BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

OPTIMIZATION IN TECHNICAL SCIENCES: ALGORITHMS AND APPLICATIONS

OPTIMALIZACE V TECHNICKÝCH VĚDÁCH: ALGORITMY A APLIKACE

HABILITATION THESIS
HABILITAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE

Ing. JAKUB KŮDELA, Ph.D.

BRNO 2023

ABSTRACT

Optimization, seen as a process of perpetual improvement, is firmly rooted in the fabric of human society. From engineering design to the decisions on financial markets, from our daily activity planning to organizing our next vacation, and from computer science to industrial applications, optimization techniques, models, and algorithms have an extremely broad appeal.

This habilitation thesis summarizes the contributions of the author to optimization related fields through a detailed discussion of fourteen papers (co-authored by the author of the thesis). In particular, in relation to optimization algorithms we present eleven papers that cover heuristic algorithms (mainly for the field of evolutionary computation), surrogate-assisted techniques, mathematical programming algorithms, and methods used for benchmarking and comparison of various optimization algorithms. On the modelling and application side of optimization, we present three papers from diverse technically oriented disciplines (waste management, structural design, and social distancing). Each of these either propose new models for the problem at hand, or they include algorithmic techniques that take advantage of the problem-specific structure. Reprints of the papers are enclosed in appendix.

KEYWORDS

Optimization, Optimization Algorithms, Optimization Models, Mathematical Programming, Evolutionary Algorithms, Benchmarking

CITATION

KŮDELA, Jakub. *Optimization in Technical Sciences: Algorithms and Applications*: habilitation thesis. Brno: Brno University of Technology, 2023. 275 p.

Contents

CHAPTER 1

Optimization in Technical Sciences

1.1 | Introduction

Optimization is a process that is intrinsic to all living beings. Human societies are in a never ending cycle of seeking to improve their capabilities, while individuals aim to improve their lives [105]. In the animal kingdom, optimization can be found as one of the guiding principles in the evolution of species - the fitter organisms are more likely to survive. Optimization principles are also a fundamental part of physics, as systems are driven to their lowest energy state subject to physical laws. To quote Leonhard Euler, "For since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

Optimization has a broad appeal as it is utilized in various domains and because it aids the human desire to make things better [105] - from engineering design to financial markets, from our daily activity to planning our holidays, and computer sciences to industrial applications. Any problem where a decision needs to be made can be cast as an optimization problem. We always intend to maximize or minimize something. Although the term "optimization" is frequently used to mean "improvement", we will use a more precise definition: finding the best possible (i.e., optimal) solution by changing variables that can be controlled, subject to given constraints.

We can write a general optimization problem as:

$$\begin{aligned} &\text{minimize (or maximize)} \;\; f(x) \\ &\text{subject to} \;\; x \in \mathcal{X}, \end{aligned} \tag{1.1.1}$$

where x is an n-dimensional optimization variable (or vector, or "design point"), which can be written as $x = [x_1, x_2, \ldots, x_n]^T$ (i.e., a column vector). The elements in this vector can be adjusted to minimize the objective function f (sometimes also called optimization criterion or goal function). Any value of x from among all points in the feasible set $\mathcal{X}$ that minimizes the objective function is called an optimal solution or minimizer, such solution is denoted as x^*.

Although there are some simple optimization problems that can be reasoned about and solved analytically, the majority of practical problems of interest is too complex to be approached in this way. Fortunately, the advances of numerical computing, together with the development of specialized optimization algorithms, gave us the ability to deal with problems of increasing complexity [105].

The constraints of the optimization problem (1.1.1) are not typically specified directly through a known feasible set $\mathcal{X}$. Instead, the feasible set is typically formed from two types of constraints:

- equality constraints, $h(x) = 0$
- inequality constraints, $g(x) \leq 0$

Any optimization problem can be rewritten using these constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & h_i(x) = 0, \quad \text{for all } i \in \{1, \ldots, l\}, \\ & g_j(x) \leq 0, \quad \text{for all } j \in \{1, \ldots, m\}, \end{aligned} \tag{1.1.2}$$

where l and m are the number of equality and inequality constraints that describe the feasible set, respectively. Constraints can also be constructed from a feasible set $\mathcal{X}$ as

$$h(x) = (x \notin \mathcal{X}),$$

where Boolean expressions evaluate to 0 or 1.

1.2 | Design Optimization Process

In this text, we will mainly focus on the process of optimization in technical sciences and engineering. The engineering design can be seen as an iterative process that engineers follow to develop a product that achieves a given goal. This design process can be divided into the sequence of phases which are (in a simplified form) shown in Figure 1.1 (top). The role of the designer is to delimit the specifications of the problem that give a detailed account of its parameters, constants, objectives, and constraints. They are also responsible for crafting the problem and quantifying the merits of potential designs [85].

In order to accelerate the design cycle (or fined better designs), the iterative design process can be replaced (or enhanced) by design optimization. The design optimization process, shown (in a simplified form) in Figure 1.1 (bottom), shares several steps with the conventional design process. However, to perform design optimization one needs a formal formulation of the optimization problem that describes the variables that can be modified, the objective (or objectives) to be minimized (or maximized), and the constraints that should be satisfied. The assessment of the

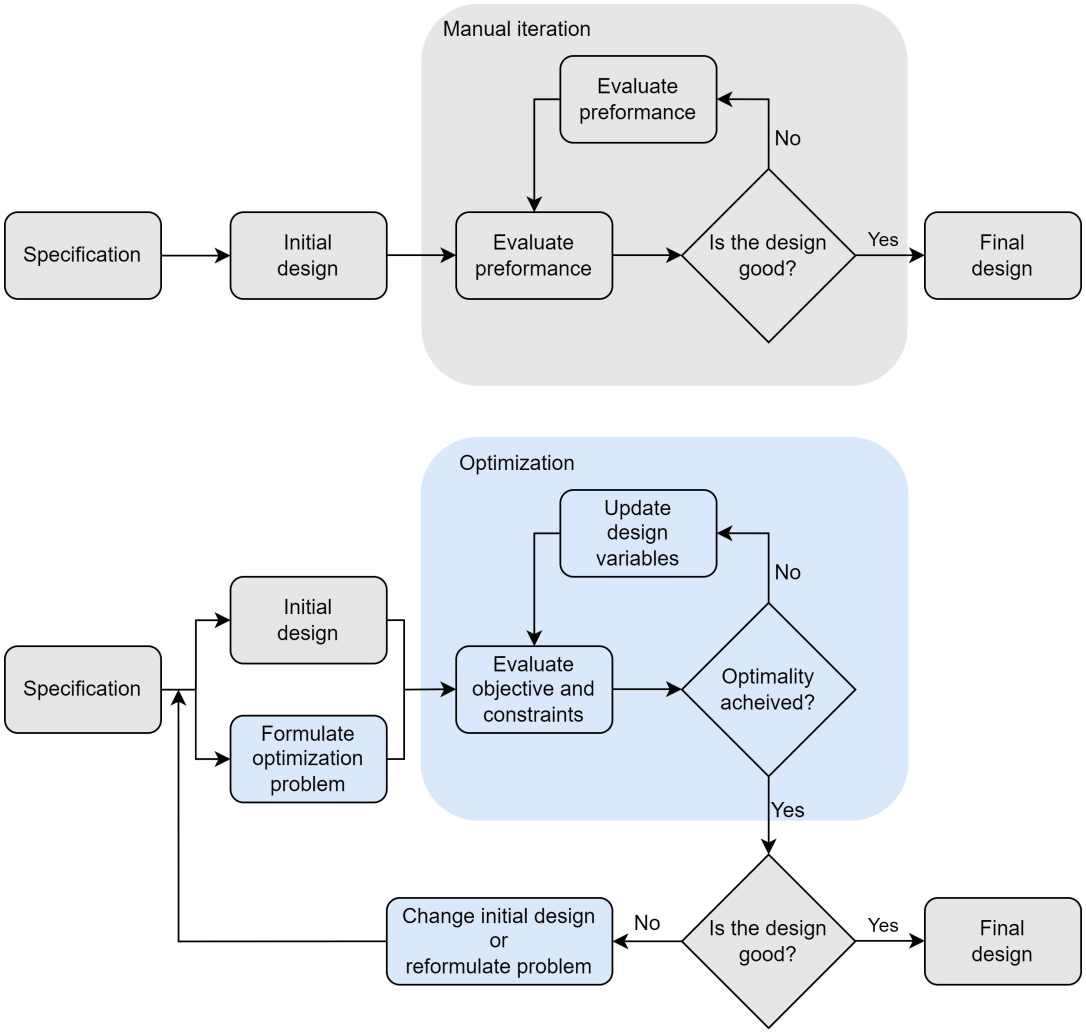


Fig. 1.1: Conventional (top) versus design optimization process (bottom), adapted from [105].

resulting design is then strictly based on numerical values of the constraints and the objective.

Even after the optimization is done, this process is still not finished. The engineers must still assess the resulting design, as it is not very likely that the baseline formulation yielded valid (or practical) solutions. But, based on this solution, the engineers might choose to modify the optimization problem (i.e., by expanding the set of design variables, changing the objective function, adding or removing some constraints, or increase/decreasing the models' fidelity).

1.3 | Optimization Problem Classification

There is an extreme amount of variety of problems that fit into the formulation (1.1.1). This first important distinction we can make is with respect to the information about the objective function f and the feasible set $\mathcal{X}$ that is available. In some problems, an analytic description of the functions involved might be available. This information can be utilized to compute values that could be useful for optimization (such as gradient or Hessian values), or it can help us to reason about interesting characteristics of the functions (such as linearity, unimodality, convexity, etc.). In other problems, we might be faced with a simulator (or a "black-box"), that given inputs (variable values) produces only outputs in the form of the values of the objective function and feasibility of the design. Such a simulator might not give us any other information, and running it could take a non-negligible amount of time.

To successfully select the most appropriate optimization algorithm for solving a given optimization problem, one should correctly classify the optimization problem. The availability of knowledge and values of the different attributes affect the efficacy and suitability of different optimization algorithms. This is important because there is no optimization algorithm that is efficient (or even appropriate) for all types of problems [8, 52].

We can classify optimization problems based on two features: the problem formulation and the characteristics of the objective and constraint functions [105], as depicted in Figure 1.2. In the problem formulation, the variables can be either continuous, discrete, or a mix of the two. The problem may only be concerned with a single objective function, or it have have multiple ones. There are problems without any constraints (such as least-square regression), with only box-type constraints on the variables (i.e., each variable should lie in a given range), or with general-type constraints that describe the limits on how the variables should be set.

Based on the characteristics of the objective and constraint functions we can infer how much "exploitable structure" the optimization problem has. Generally speaking, the more structure we find, the easier the problem is to solve. Or, put alternatively, we can solve much bigger problems (in terms of number of variables) that have some "exploitable structure" than those without any. For instance, one of most "well-structured" problems are the ones where the functions are linear (hence convex and unimodal) and deterministic. With such structure, even problems with millions of variables can be solved using modern techniques and hardware. On the other end of the spectrum, we have problems that are nonlinear, nonconvex, multimodal, and stochastic. For such problems, even having a dozen of variables can be enough for the problem to be practically unsolvable in a reasonable amount of time (in the sense of finding a provably optimal global solution).

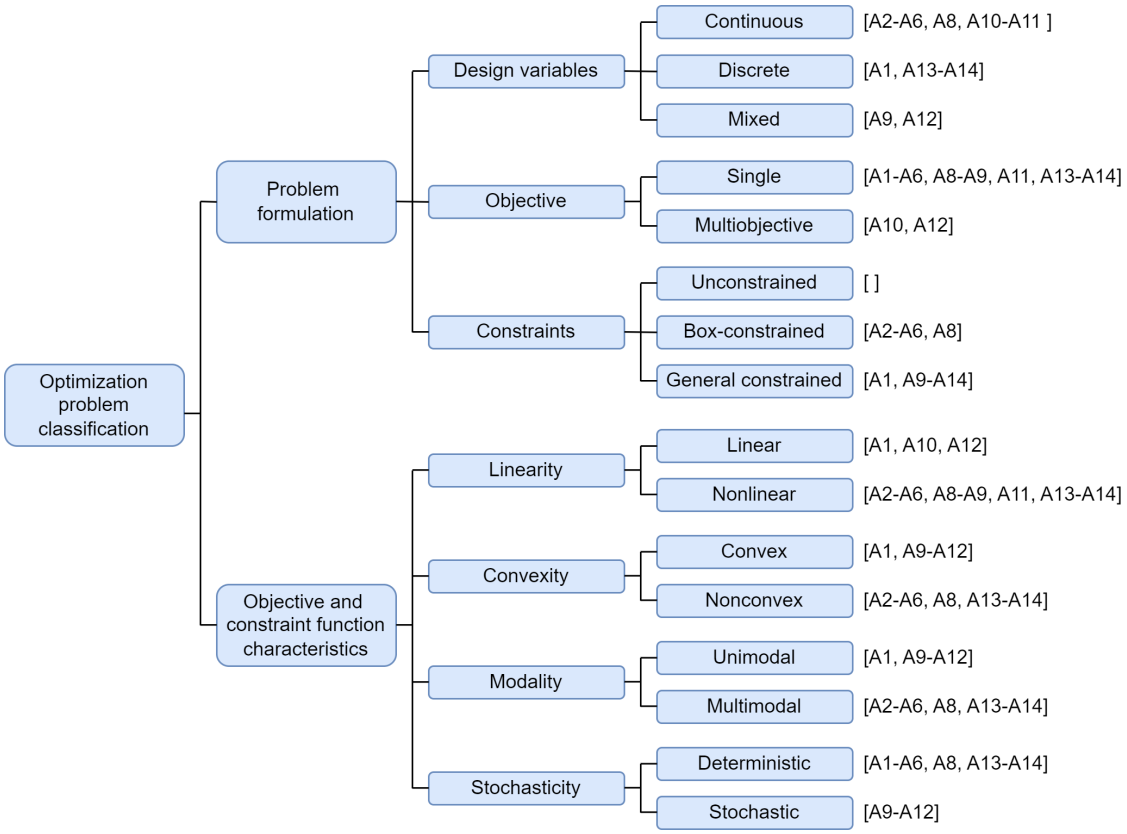


Fig. 1.2: Classification of optimization problems by attributes associated with the different aspects of the problem, adapted from [105].

1.4 | Optimization Algorithms

There is not a single optimization algorithm that can be effective (and even suitable) for all optimization problems. Good understanding of the ("exploitable") structure of the problem is always pivotal in choosing which optimization algorithm to use. Figure 1.3 shows the basic classification of optimization algorithms based on certain attributes. Most optimization algorithms have several of these attributes, but they are independent, and any combination is possible. However, this classification still does not cover the whole variety of specialized methods designed to solve specific problems where a particular structure can be exploited (such as separability, sparsity, etc.).

The minimum amount of information an optimization algorithm can require are the values of the objective and constraint functions at queried points. The methods that use the function value information are generally known as derivative-free (or gradient-free, or zeroth-order) algorithms. If the information about the derivative of the functions (called first-order information) is available (or computable in a reasonable amount of time), one might use some of the gradient-based methods.

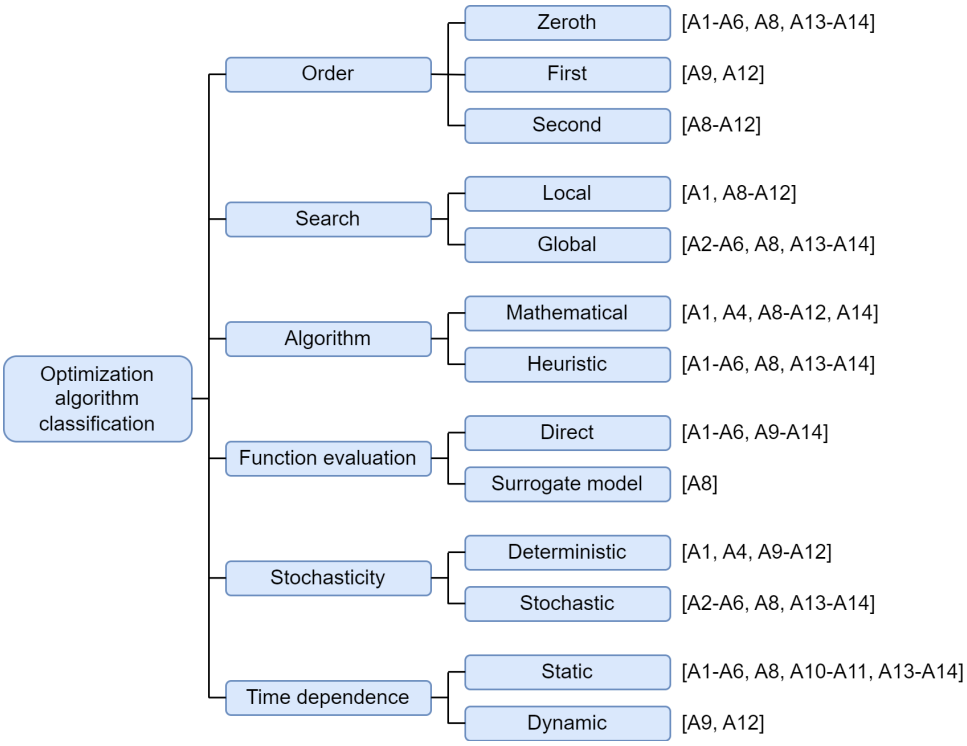


Fig. 1.3: Classification of optimization algorithms by different attributes (rightmost column), adapted from [105].

The gradient information can also be used in deciding if a given design can be (at least locally) optimal. In some problems (most notably convex ones with lost of "exploitable" structure), second-order information (i.e., Hessian values) are utilized by second-order (Newton-type) methods.

There are various techniques that are used for searching the design space, with the most basic classification being into the local and global methods. In local methods, the search typically converges towards a local minimum. Global search techniques try to span (or sometimes decompose) the whole design space in the hope of finding the global minimum. The choice of the local or global-type method is closely linked to the modality of the design space [105]. In unimodal spaces, local searches are sufficient as they will converge to the global minimum. In multimodal spaces, global search methods increase the likelihood of finding the global minimum (although finding one ca be rarely guaranteed).

Most algorithms can be divided by the mechanisms and concepts they are structured upon. Mathematical optimization algorithms are described by an iterative process, that determines the sequence of evaluated points when searching for an optimum. These algorithms also use optimality criteria, that determine whether this iterative process should end. On the other hand, heuristics are generally described as commonsense arguments, or rules of thumb. They do not have to be based on a

strict mathematical rationale, but can come from experience with the given problem, or take inspiration in naturally-occurring phenomena. Many algorithms work by combining mathematical arguments and heuristics. Mathematical algorithms also often include hyperparameters whose values are generally tuned based on experience. For instance, one of the state-of-the-art solvers for mixed-integer problems, GUROBI [61], makes use of various heuristics[1].

The above described setup of the optimization problem assumes that the function evaluations are acquired by solving numerical models of the system (so-called direct function evaluations). However, these evaluations might entail time-consuming numerical simulations or running costly experiments. In such situations it is often possible to construct surrogate models (also called metamodels) of these expensive functions and use them in the optimization process instead.

The static time dependence means that the problem was formulated as a single optimization and the complete numerical model is solved at each optimization iteration. On the other hand, in dynamic optimization problems (also known as dynamic programming) we need to solve a sequence of optimization problems to make decisions at different time stages. These decisions should be based on information that becomes accessible as time progresses. The decision at a certain time stage is also influenced by the decisions and system states from previous stages (and may also depend on a prediction of the states a few steps into the future) [16].

The stochasticity attribute is independent of the stochasticity of the model that was discussed above, and describes whether the optimization algorithm itself uses steps that are determined at random or not. In a deterministic method, the same points are evaluated and the method converges to the same result (given identical initial conditions). In contrast, stochastic methods evaluate a different points if run multiple times from identical initial conditions, even if the objective and constraints functions are deterministic.

In some problems, the dynamic nature of the decision-making process is best captured in the use of models with several decision stages [91]. Whereas single-stage models are generally well-suited for situations, in which the corresponding model data are static (do not change over time), multi-stage models are more appropriate for environments that change dynamically. The difficulty with using the multi-stage approaches is that they lead to an increase in modelling complexity and needed computational resources. An intermediate step between the single- and multi-stage approaches is the two-stage model. In this setting, typically the "important decisions" (such as facility locations in network design), are made in the first stage of the model. The second stage is then used to assess the impact that the "important

[1] https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/

decisions" will have in time.

A very similar dichotomy, as the one between the single- and multistage models, can be found in the approaches to the considered data of the model. Modelling a static situation with very little data variation usually means that the deterministic approaches are the most sensible ones. However, when the system that should be managed is dynamically changing and there are multiple possible paths of development, uncertain models are the most reasonable choice [91].

1.5 | Aim of the Thesis

The aim of the thesis is to provide an insight into some of the techniques that are used in modelling and solving optimization problems in technical sciences. The selection of the topics is based on the work experience of the author, and the topics are split into five main sections. The first section is focused on heuristic algorithms. The second section deals with surrogate-assisted methods. The third section discusses some mathematical programming (more concretely, stochastic programming) methods. The fourth section focuses on practices in benchmarking optimization methods. Finally, in the last section are described several applications of optimization models. Each of these sections first gives a brief overview of the current development in the field, which is followed by subsections that provide a commentary to and a show the most important excerpts from the journal papers co-authored (with significant contribution) by the author of the thesis, demonstrating his contribution to the field.

Overall, the thesis contains fourteen peer-reviewed papers co-authored by the author of the thesis. Figures 1.2 and 1.3 show how each of the papers fits within the categorization of the optimization problem being solved, and optimization algorithm being used. The only exception is the review paper [A7] that is not shown in the Figures, as it would fit most of the categories. Each presented paper has been either published in a peer-review journal with an impact factor (IF) according to Web of Science (ten papers), or it has been presented at a CORE-ranked[2] conference (four papers). The papers themselves can be found in the Appendix.

Among the included ten journal papers, three papers are contained in the first decile (i.e., in the top 10% journals in a given category) of mostly engineering or computer science-related areas according to the Journal Citation Reports (JCR). The highest impact factor of the included papers is 25.898, the best CORE-ranked conference was A.

[2]https://www.core.edu.au/

CHAPTER 2

Algorithms and Applications

2.1 | Heuristics Methods

Heuristics are general algorithmic techniques that are designed to solve complex optimization problems. They aim at finding a "sufficiently good" solution to an optimization problem, and are especially well-suited for problems with limited amount of information and structure, or in settings with a limited computational capacity [17]. Most of these methods belong to the category of derivative free (or zeroth-order) algorithms.

Heuristic methods can be very roughly divided into two classes. In the first are the deterministic methods that often use general geometric patterns in their search [85]. Most of the methods in this class are older (but still used) techniques. The canonical examples include the Powell's method [128], which is based on line searches in dynamically updating directions. The Hooke-Jeeves method [70], in which the search is based on evaluations at small steps in the coordinate directions. A generalization of the Hooke-Jeeves method called the Generalized Pattern Search [49] can search in arbitrary directions (provided that they form a positive spanning set). The Nelder-Mead Simplex Method [98] uses a pattern called the simplex (a generalization of a tetrahedron to n-dimensional space) to traverse the design space.

There is one category of the derivative-free methods are the techniques based on the concept of Lipschitz continuity and design space decomposition. Such methods are not really heuristics, as can be shown to converge to the global optimum (when the Lipschitz continuity assumptions are satisfied). However, they are generally only viable in finding globally optimal points in lower dimensions, as the space-partitioning suffers from the curse of dimensionality. When the objective function evaluations are limited, the best-found solutions from these methods can be seen as a "heuristic" one, with the added bonus of also providing valid lower bounds on the global optimal function value. The first such method was called the Shubert-Piyavskii approach [127, 150]. The more recent variants of this approach are based on the divided rectangles (or DIRECT) algorithm [74].

Metaheuristics are stochastic methods which use randomness to explore the design space, and make relatively few assumptions about the optimization problem being solved [19]. The two most notable areas where metaheuristics found extensive use are combinatorial and black-box optimization problems.

There is a wide variety of metaheuristic methods. The most basic distinction can be made into three classes. The first class are the single-solution methods which work by modifying and improving a single candidate solution. In this category, we can find methods such as Mesh Adaptive Direct Search [6], which is similar to the generalized pattern search method but uses randomly generated positive spanning set. The popular Simulated Annealing [82] method took the inspiration for the design space search from metallurgical processes. Another widely-used methods are Iterated Local Search, Tabu Search, or Variable Neighborhood Search [159].

The second class are the methods that maintain and modify an explicit probability distribution (often called the proposal distribution) over the design space. At each iteration, the proposal distribution is used to generate samples and the update the proposal distribution is based on the best-found samples (in terms of the objective function value). The hope is for the proposal distribution to converge towards the global optimum. Among these methods are the Cross-entropy method [137], or Natural Evolution Strategies [139], The perhaps most popular method in this class is the covariance matrix adaptation evolutionary strategy (CMAES) [65], which is a robust and sample-efficient method that maintains a multivariate Gaussian proposal distribution with a full covariance matrix [85].

The third class are the Evolutionary computation (EC) or population-based approaches, which find inspiration in naturally occurring processes. These methods maintain multiple candidate solutions (called a population), which are evolved together. These metaheuristics include genetic algorithms [69], differential evolution (DE) [44], or particle swarm optimization (SPO) [21, 79]. A closely related category of metaheuristics are swarm-intelligence methods, which mimic a collective behavior of decentralized, self-organized agents. Examples of such techniques are ant colony optimization [50], artificial bee colony optimization [76], or the previously mentioned PSO. We will describe both DE and PSO in a little more detail, as they were used in the papers (co-written by the author of this thesis) [A8, A2, A6, A3, A5] that will be discussed in detail in further sections.

The DE method is very popular optimization framework, because of its straightforward structure and its global optimization capabilities. Several variants of DE have been developed to improve its performance [44]. The computation of DE can be divided into four stages: initialization, mutation, crossover, and selection. We assume we have a population at the current generation, $x = [x_1, \ldots, x_t]$, where each individual has dimension D, $x_i = (x_i^1, \ldots, x_i^D)$. There are several variants of DE. In

this text, we show the DE/best/1 strategy for the mutation process of DE which, can be expressed as

$$v_i = x_b + F \cdot (x_{i_1} - x_{i_2}), \tag{2.1.1}$$

where x_b is the current best solution, x_{i_1} and x_{i_2} are different randomly selected individuals from the population, and F is a scalar number typically within the interval [0.4, 1] [44]. After mutation, the crossover stage is conducted which has the following form:

$$u_i^j = \begin{cases} v_i^j, & \text{if } (U_j(0,1) \leq C_r \,|\, j = j_{rand}), \\ x_i^j, & \text{otherwise,} \end{cases} \tag{2.1.2}$$

where u_i^j the jth component of ith offspring, x_j^i and v_j^i are the jth component of ith parent individual and the mutated individual, respectively. The crossover constant C_r is between 0 and 1, $U_j(0,1)$ indicates a uniformly distributed random number, and $j_{rand} \in [1, \ldots, D]$ is a randomly chosen index that ensures u_i has at least one component of v_i.

The PSO method introduced momentum to accelerate convergence toward minima. Each individual i (or in this case a particle), in the population has a memory of its current position x_i, velocity v_i, and the best position (in terms of the objective function value) that it has seen thus far $x_{i,\text{best}}$. Momentum allows a particle to accumulate speed in a hopefully favorable direction, which could be independent from local perturbations [85]. At each iteration, the individuals are accelerated in the direction of both the their individual best positions and also in the direction of best position found thus far by the whole population x_{best}. These two accelerations are weighted by separately generated random terms. The equations describing PSO are

$$x_i = x_i + v_i$$
$$v_i = wv_i + c_1 r_1 (x_{i,\text{best}} - x_i) + c_2 r_2 (x_{\text{best}} - x_i)$$

where w, c_1, and c_2 are parameters, and r_1 and r_2 are random numbers drawn from $U(0,1)$.

There are some optimization problems in which the number of variables is unknown, such as in the optimization of computer programs or graphical structures [85]. The designs in such contexts can be represented by expressions that belong to certain grammar. The two most well-known EC techniques for such problems are Genetic programming [87], which represents individuals using tree structures, and Grammatical evolution [138] that operates on an integer array instead of a tree. Some of our work constituted using genetic programming for symbolic regression [107].

Over the past few years, the field of EC have witnessed an explosion of "novel" methods that are based on natural/evolutionary principles. The bestiary of EC[1], which tries to catalog of a portion of these nature-based methods, now contains over 250 methods that claim their inspiration in natural processes. It has also become clear that there was more creativity being spent at naming some of these "novel" methods, than in making sure they contain anything new computation-wise [5]. Many of these methods were found to be just a "rebranding" of older techniques [172, 30, 31, 165, 29], and most of them are of questionable quality [113, 47, 26, 32, 94].

The state-of-the art EC methods for box-constrained numerical optimization can generally be found at the Congress on Evolutionary Computation (CEC) competitions which started in 2005 and continue to this day [99, 47, 26]. Many of these methods are modern adaptations of the standard techniques, such as DE or CMAES. The five methods from these CEC competitions that were utilized in some of the papers included in this thesis are the following. Adaptive Gaining-Sharing Knowledge (AGSK) [112] – the runner-up of the CEC 20 competition. The algorithm improves and extends upon original GSK [111] algorithm by adding adaptive settings to two main control parameters: the knowledge factor and the knowledge ratio, that control junior and senior gaining and sharing phases among the solutions during the optimization process. Hybrid Sampling Evolution Strategy (HSES) [180] – winner of the CEC 18 competition. HSES is an evolution strategy optimization algorithm which combined CMAES and the univariate sampling method. Improved Multi-operator Differential Evolution (IMODE) [140] – winner of the CEC 20 competition. This algorithm employs multiple differential evolution operators and a sequential quadratic programming local search technique for accelerating its convergence. Linear Population Size Reduction SHADE (LSHADE) [160] – one of the most popular variants of adaptive DE, that was used as a basis for many of the best performing algorithms in the CEC competitions in past few years. Multiple Adaptation DE Strategy (MadDE) [18] – one of the best performing algorithms in the CEC 21 competition. This is another modification of the DE algorithm that uses a multiple adaptation strategy for its search mechanisms and for adapting its control parameters at the same time.

Many EC methods perform relatively well in global search, using their exploration abilities to avoid local minima and to find the best regions of the design space [85]. On the other hand, such methods often do not perform as well in local searches (at least in comparison to descent methods). Several hybrid methods (also known as memetic algorithms) have been proposed to extend population methods

[1]Campelo, F., Aranha, C. Evolutionary computation bestiary. https://github.com/fcampelo/EC-Bestiary

with descent-based features to improve their exploitation performance. The IMODE method mentioned above is one example of such an approach. Our work in [A8] can also fit into this category. Another hybridization approach are the composite schemes in [62] and in our work [97] that combine several methods at different stages of the search for the purpose of getting the respective benefits of the different methods.

Another algorithm that was utilized in the papers included in this thesis was the binary HC12 algorithm, which is a metaheuristic searching algorithm that was developed in [109]. The basic step of the algorithm is a generation of a neighborhood of the current solution, which serves as a base for the construction of a new (improved) population. The method of generating the neighborhood is the pivotal characteristic of HC12. The paradigm of the algorithm is the search of the optimal solution in the binary (Hamming) space, that encodes the solution. In this context, it is a parallel approach to genetic algorithms, where the solution is encoded as a binary vector.

The author's contribution to this field was mainly in the areas of surrogate assisted methods [A7, A8] and in benchmarking EC methods [A2, A6, A3, A5], both of which are covered in further sections. In this section, one CORE-ranked conference paper and one IF journal paper that have a common theme are discussed in detail:

[A13] R Matoušek, L Dobrovský, J Kůdela. The quadratic assignment problem: metaheuristic optimization using HC12 algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, pages 153–154, 2019. ··

Author's contribution: 33%.

Ranking (Core 2019): A - information and computing sciences.

[A14] R Matoušek, L Dobrovský, J Kůdela. How to start a heuristic? utilizing lower bounds for solving the quadratic assignment problem. International Journal of Industrial Engineering Computations, 13(2):151–164, 2022.

Author's contribution: 33%.

Metrics: $\text{IF}_{2021} = 3.271$.

Ranking (JCR 2021 WoS): Q2 - operations research & management science; Q3 - engineering, industrial.

2.1.1 | The Quadratic Assignment Problem: Metaheuristic Optimization Using HC12 Algorithm

The paper [A13] focuses on the Quadratic Assignment Problem (QAP), which is one of the classical NP-hard combinatorial optimization problems [37]. The QAP, in its Koopmans and Beckmann form [86], can be described in the following way: The problem is structured on a complete directed graph with n nodes and n^2 arcs. There is also a set of n facilities, that one needs to assign to the nodes. The indices i, j correspond to the nodes, while the indices f, g correspond to the facilities. Problem data encompass encompass a given (directed) distance from node i to node j ($b_{i,j} \geq 0$), a flow from facility f to facility g ($a_{f,g} \geq 0$), and a cost of assigning facility f to node i ($c_{i,f}$). By using binary variables $x_{i,f} = 1$ if facility f is assigned to node i, and 0 otherwise, the QAP can be stated as the following binary optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_i \sum_f \sum_j \sum_g a_{f,g} b_{i,j} x_{i,f} x_{j,g} + \sum_i \sum_f c_{i,f} x_{i,f} \\ \text{subject to} \quad & \sum_i x_{i,f} = 1, \quad \sum_f x_{i,f} = 1, \qquad \forall f, \forall i \\ & x_{i,f} \in \{0, 1\}, \qquad \forall f, \forall i \end{aligned} \tag{2.1.3}$$

In many approaches, the facility-assignment costs are neglected (i.e., $c_{i,f} = 0$). There have been several extensions of the QAP formulation have proposed over the years – among the most notable of these are the multiobjective [141] and stochastic formulations [108]. The QAP can also be equivalently formulated (neglecting $c_{i,f}$) as a problem on permutation matrices:

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(AXBX^T) \\ \text{subject to} \quad & X \in \Pi_n \end{aligned} \tag{2.1.4}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix, A and B are matrices of the data ($a_{f,g}$ and $b_{i,j}$), and Π_n is the space of permutation matrices with n rows/columns.

The paper presents the metaheuristic algorithm HC12, developed in [109], and its utilization in solving some QAP instances from the standard QAPLIB library [28]. In QAPLIB, most problems are in a range from $n = 10$ to $n = 256$. The smallest instance for which the optimal solution was still not confirmed (and not for a lack of trying) is tai25a (with $n = 25$). This illustrates how difficult of a combinatorial problem the QAP is. The implementation of HC12 searched for the best solution in multiple runs (restarts of the algorithm). Although the running times of the HC12 algorithm were extremely fast (compared to the other heuristics discussed in the paper), the robustness of the resulting solutions was still rather low and required additional research and tuning.

2.1.2 | How to Start a Heuristic? Utilizing Lower Bounds for Solving the Quadratic Assignment Problem

The core focus of the paper [A14] was also in the QAP. It showed the possible utilization of lower-bounding techniques in constructing starting points for the HC12 metaheuristic. Exact solution (derived from mathematical programming techniques) of a QAP typically requires the use of a branch-and-bound framework [3]. The key factor in the difficulty of the problem is the lack of efficiently computable and tight lower bounds. Generally, the tighter the bound is, the more difficult it is to compute.

There were various approaches proposed for computing valid lower bounds. One of the oldest methods, known as the Gilmore-Lawler bound (GLB) [60, 100], is widely used to this day. A computational comparison of the older bounds, which were based on linearization of the QAP, can be found in [78]. Another type of bounds are the so-called eigenvalue bounds that utilize the QAP formulation (2.1.4) and use the fact that the set of permutation matrices Π_n can be characterized as:

$$\Pi_n = \mathcal{Q}_n \cap \mathcal{E}_n \cap \mathcal{N}_n$$

where $\mathcal{Q}_n$ is the set of orthogonal matrices, $\mathcal{E}_n$ is the set of doubly stochastic matrices, and $\mathcal{N}_n$ is the set of matrices with positive elements of size $n \times n$. The most notable bound based on this decomposition is the so-called Hadley-Rendl-Wolkowitz (HRW) bound [63]. The first significant breakthrough in the computation of lower bounds for the QAP was the development of the convex quadratic programming bound introduced in [4], which led to great success in solving previously unsolved QAP instances.

One of the latest seminal breaking points in combinatorial optimization was the emergence of semidefinite programming (SDP) techniques [24]. For the QAP, the SDP bounds were first studied in [181]. The problem with this relaxation was that it involved a matrix variable of order n^2 and could only be efficiently solved by interior point methods for, roughly, $n \leq 20$. This limitation of the SDP bounding technique has encouraged research into exploring group symmetries of the QAP data matrices to construct smaller and more tractable SDP problems [45]. Another research direction was in the SDP relaxations of QAP where the matrix variables based on matrix-splitting, which were of order n [124]. In both these lines of research the new techniques led to finding the best-known lower bounds for some QAPLIB instances.

In the paper [A14], the construction of the different lower bounding techniques (GLB, HRW, convex quadratic-based bound, and one semidefinite-based bound) is presented in detail. These bounds are then used to construct a starting point (or in the case of the QAP, a starting permutation) for the HC12 metaheuristic. We

use a projection of the matrix obtained from the lower bounding schemes to the space of permutation matrices. Let $\hat{X}$ be a matrix obtained from the computation of the lower bounds. The "closest" permutation matrix X to $\hat{X}$ can be computed by solving the following problem:

$$\begin{aligned} \text{minimize} \quad & \sum_i \sum_j (X_{i,j} - \hat{X}_{i,j})^2 \\ \text{subject to} \quad & \sum_i X_{i,j} = 1, \qquad \forall j \\ & \sum_j X_{i,j} = 1, \qquad \forall i \\ & X_{i,j} \in \{0,1\}, \qquad \forall i, \forall j. \end{aligned} \tag{2.1.5}$$

The problem above can be reformulated into an equivalent problem using the fact that $X_{i,j}$ binary:

$$\begin{aligned} \text{minimize} \quad & \sum_i \sum_j (1 - 2\hat{X}_{i,j}) X_{i,j} \\ \text{subject to} \quad & \sum_i X_{i,j} = 1, \qquad \forall j \\ & \sum_j X_{i,j} = 1, \qquad \forall i \\ & X_{i,j} \in \{0,1\}, \qquad \forall i, \forall j, \end{aligned} \tag{2.1.6}$$

which is a simple Linear Assignment problem, that can be solved in polynomial time [27].

The computational experiments were carried out on 53 symmetrical QAP instances from the QAPLIB. The results showed that, at least in general, the more complicated formulations (convex quadratic and semidefinite) produce better (higher) lower bounds, but not necessarily better (lower) starting point values, when judged solely on the resulting projection. The trade-off was that these more complicated formulations needed significantly more computational resources (judged by the computational time) and were only feasible for instances up to $n = 100$. Also, every considered lower-bounding method produced the best lower bound and best projected value for at least one problem instance. It is important to note that the lower bounds were not only useful for the construction of possible starting solutions for the HC12 metaheuristics, but can also help to judge the closeness of the obtained solution to the true optimum.

Next, we used the projected values from the lower bounding techniques as the starting points for the HC12 metaheuristic and run it 1,000 times for each problem instance and are compared to simulations that used random permutations as the starting point (the results are reported in various Tables in the paper). The main

takeaway from the results is that even the "worst" performing lower bounding technique (HWB) was significantly better that random start, beating it in 30 of the 53 instances. The "best" performing lower bounding technique was the most complicated semidefinite formulation, which was better than random start in 41 of the 52 instances.

From the results, it was apparent that starting a heuristic from a carefully chosen points can lead to an increase in quality of the resulting solutions and in a more robust convergence. The choice of the technique for constructing the starting points depended primarily on the computational resources at ones disposal. While for the QAP the SDP-based formulation produced the best starting points, it was also the most computationally demanding method, requiring the utilization of advanced convex optimization algorithms or the use of one of the most powerful available solvers. In contrast, the GLB bound produced starting points that were almost as good as the SDP-based ones, but the computational requirements for GLB were negligible.

2.2 | Surrogate-assisted Methods

The analysis of complex systems, where obtaining an analytical solution may be either difficult or impossible, one often resorts to methods of numerical analysis such as the finite-element method (FEM), computational fluid dynamics (CFD), or structural finite-element analysis (FEA). Such analyses are becoming broadly utilized in evaluating and optimizing design, reliability, and maintenance of complex systems and structures in a vast range of industrial applications such as aerospace [176], automotive [15], sustainable architecture [171], and many others.

The problem with such computer simulations is that they tend to be very computationally demanding, because of their intrinsically detailed description of the studied systems. The problems based on FEM or CFD models also require the computation of thousands of simulations in order to construct a suitable solution, which requires a large computational budget [2].

The main purpose of using surrogate models (also called metamodels) is to approximately emulate the expensive-to evaluate high-fidelity models, employing computationally less costly statistical models. The surrogate models are build based on a relatively low number of simulations based on input and output data, that are computed by using the high-fidelity expensive computations. After the surrogate model is validated and achieves a satisfactory level of approximation of the high-fidelity model, its utilization in predicting the outputs of this high-fidelity model can be done at a fraction of the computational cost.

To mitigate the computational costs, surrogate models have been widely used in combination with evolutionary algorithms (EAs), which are known as surrogate-assisted EAs (SAEAs) [73]. SAEAs only execute a limited number of real objective function (or constraint) evaluations and use them to train the chosen surrogate models. Based on the current surrogate model, the SAEAs typically choose between two types of solutions for the real function evaluations: either promising samples which are around the current optimum of the surrogate model, or uncertain samples with a large expected approximation error. Surrogate models can guide the search of EAs to promising directions by using optima of these models, as was demonstrated [132]. Also, it has been found that evaluating the uncertain samples can strengthen the exploration capabilities of SAEAs and effectively improve the approximation accuracy of the surrogate [72], and over the years various approaches for estimating the degree of uncertainty in the function predictions have been proposed [169].

In recent years, a multitude of SAEAs has been proposed in the literature. Usually, such algorithms employ a metaheuristic method to be the primary search strategy and employ the surrogate models as additional tools in order to accelerate the convergence of the underlying metaheuristic algorithm. Because of the curse of

dimensionality, it is generally difficult for EAs to search for global optima in high-dimensional spaces. And SAEAs encounter this challenge as well when the dimension of a problem is high. Although current SAEAs can handle high-dimensional expensive problems relatively well, most of these algorithms still need many (usually more than several thousands) function evaluations to obtain "good" optimization results. The use of multiple surrogates have been shown to perform better than single ones in assisting EAs. Generally, this approach works by utilizing a global surrogate model to smooth out the local optima, and local surrogate models to capture the local details of the fitness function around the neighborhood of the best-found solutions.

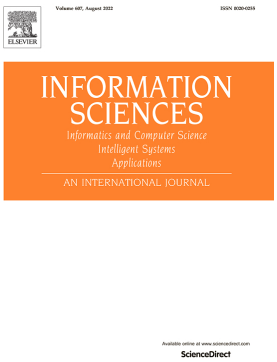
Two IF journal papers are included in the thesis in order to demonstrate the author's contribution to the field:

[A7] J Kůdela, R Matoušek. Recent advances and applications of surrogate models for finite element method computations: a review. Soft Computing, 26(24):13709–13733, 2022.

Author's contribution: 80%.

Metrics: $\text{IF}_{2021} = 3.732$.

Ranking (JCR 2021 WoS): Q2 - computer science, artificial intelligence; Q2 - computer science, interdisciplinary applications.



[A8] J Kůdela, R Matoušek. Combining lipschitz and rbf surrogate models for high-dimensional computationally expensive problems. Information Sciences, 619:457–477, 2023.

Author's contribution: 90%.

Metrics: $\text{IF}_{2021} = 8.233$.

Ranking (JCR 2021 WoS): D1 - computer science, information systems.

2.2.1 | Recent Advances and Applications of Surrogate Models for Finite Element Method Computations: A Review

The primary motivation of the paper [A7] lied in investigating recent advances and applications of surrogate models for FEM-based computations. Although the utilization of surrogate models is growing in popularity, a comprehensive text summarizing the state-of-the-art and recent developments (especially for FEM-based computations) was missing. The novelty of the paper was in encapsulating the state-of-the-art in surrogate modelling for FEM-based computations from both the theoretical and application perspectives. It is also expected that this work will function as a guide in the selection of the suitable approximation models for applications of computationally expensive high-fidelity FEM-based problems.

The review paper [A7] emphasized the differences between employing surrogates for the three problem classes. The first class encompasses the most fundamental use of surrogates—building and validation techniques for surrogate models, and their use for prediction. The second class of problems focuses on sensitivity analysis of the resulting models and the various ways one can quantify the impact of uncertain parameters, which could influence the behaviour of the modelled systems. Lastly, the third class is commonly referred to as surrogate-assisted optimization, where the objective function or constraints used for optimization are prohibitively expensive to compute and the information about the corresponding derivatives is not available.

The paper [A7] gives a detailed account of the state-of-the-art in surrogate modelling techniques. First, different sampling strategies are presented. Both stationary sampling methods (that are based upon geometry or patterns, such as Latin Hypercube Sampling) and adaptive sampling methods (which are popular in the surrogate-assisted optimization) are discussed. Afterwards, we focused on model validation methods, which encompass resampling techniques such as bootstrapping or cross-validation [66]. Next, we describe the various model choices for surrogates and the underlying mathematical formulas: Response surfaces and linear regression [23], Kriging (which is sometimes called Gaussian process regression) [173], Radial basis functions (RBF) [25], Support vector regression [163], Artificial neural networks [157], Polynomial chaos expansion [174], Boosted trees and random forests [55], and other less-used techniques. We also discuss the use of ensembles of surrogates, that can be used to mitigate the drawbacks of using a single surrogate model, and multifidelity models, which are constructed by a combination of different fidelity models that depend on the specifics of a given problem, with the goal of reducing the high computing cost while giving accurate solutions.

Another section of the paper [A7] is devoted to Sensitivity analysis, which is a techniques that is frequently employed to determine the effect of the input parameters on a given outcome variable [177]. It is also often used in a preliminary step before an early design, analysis of uncertainty, or optimization in an effort to reduce the complexity of the studied problem. Different values of the model parameters, along with the initial (input) values of variables, are often subjected to different sources of uncertainty. A solid comprehension of the sensitivity of the outputs of the model to uncertainty in the values of the parameters and input variables is important for strengthening the confidence in the model and in the resulting predictions [2]. The paper discusses the state-of-the-art advances in two complementary approaches for Sensitivity analysis: the local and global methods. The local methods perturb the inputs of one chosen design in order to approximate its partial derivatives, which give an insight into the sensitivities of inputs around the chosen design. On the other hand, the global methods aim to determine the effect of the parameters over the entire design space. Apart from the methods for fast parameter prescreening, the global methods are generally computationally more demanding than the local ones. As both local and global methods are based on simulations, faster evaluations of surrogates can be utilized to speed up the process of sample generation, which is especially useful for variance-based techniques that require numerous samples.

The state-of-the-art in optimization of complex and costly problems that appear in real-world applications involves utilizing surrogate models during optimization [155]. The problems based on FEM simulations belong to a category of the black-box problems, in which the available problem information (i.e., mathematical equations and/or other exploitable knowledge) is very limited, and the only way of extracting any information is the costly evaluation of the candidate designs. The main purpose of surrogate-assisted optimization (SAO) is in the reduction of the computational time, resources, and the related costs by utilizing all available information efficiently in order to lower the number of needed objective function evaluations. A frequently used approached is based on performing only a few of the expensive true function evaluations for a construction of a "rough" surrogate model and running an optimization algorithm on such surrogate model. The optimal solution resulting from this computation is subsequently used as the next point for another (expensive) true function evaluation and for the refinement of the chosen surrogate model. Such process is generally repeated until some stopping criterion (such as finding a "good enough solution", long computational time, large number of iteration, achieved precision of the surrogate, etc.) is met.

Most of the algorithms used in SAO come from the class of derivative-free methods [134]. These algorithms can be further categorized into global and local search methods. Local search algorithms work by refining a given solution or to reach

a local optimum. On the other hand, global search methods utilize a mechanism that (hopefully) lets them to escape from local minima. Among the local search methods are the algorithms which sequentially evaluate candidate points that are generated based on a particular strategy (often utilizing geometric patterns), such as the Hooke and Jeeve's algorithm and the Nelder-Mead simplex method. The trust-region methods are also in this category. They use a surrogate model in a close neighbourhood of a given location. Another extensively utilized local search method is sequential quadratic programming, that constructs a quadratic approximation of the problem at each iteration and finds the corresponding solution [117]. Among the global search methods, we can find the design space-partitioning methods such as the DIRECT algorithm [74] and stochastic algorithms. In recent years, especially the stochastic algorithms have become popular for SAO [73], with methods such as evolutionary algorithms, simulated annealing, particle swarm optimization, genetic algorithms, differential evolution, and many others.

In many cases, the optimization of highly challenging problems needs to consider more than a single objective function. These multiple objectives are often aggregated into a single objective by using a weighted sum (or similar aggregation procedure) that makes the problem approachable with ordinary (single objective) optimization methods. The converse approach is to consider all objectives in parallel, which becomes especially important in situations where these objectives are conflicting, such as quality and price in production or lift and drag in airfoil design [155]. Although there have been developed several algorithms for multi-objective SAO, the field still lacks a common repository in which the different methods could be collected and compared. This is because the development of such methods is generally application-oriented and such methods have often been used to solve a particular real-word or industrial optimization problem. Some of the algorithm that are most widely used are the multiobjective genetic algorithm NSGA-II [46], the RASM method [102], and the ParEGO algorithm [83].

Another problem characteristic that plays a significant role in optimal design is uncertainty. Incorporating various ways of dealing with uncertainty into the optimization process is very often a crucial step can guarantee that the resulting design is able to handle variations in the input parameters. One of the possibilities is to apply sensitivity analysis to the result of the optimization process, and (if found inadequate) modify the objective function and/or the constraints. Another possibility lies in utilizing the robust optimization approach, which is mathematical framework for optimization that is designed to minimize the propagation of the input uncertainties to the output responses [39]. The paper [A7] gives a detailed account of the most-used surrogate model types in SAO, the optimization models and algorithms, and the various application areas.

The next section of the paper [A7] deals with the available software tools for surrogate modelling, sensitivity analysis, uncertainty quantification, and surrogate-assisted optimization. As the employment and utilization of surrogate models for analyzing and optimizing computationally expensive problems have become more prevalent, we have seen a significant increase in the availability of new software tools which provide an easy access to the needed technologies. Among these tools are ALAMO [42], ARGONAUT [22], Agros Suite and Artap [77], Eureqa [143], FReET [118], RBFOpt [41], and many other tools, mainly developed in MATLAB and Python.

The last section of the review paper [A7] presents the current trends, research gaps, and practical recommendations in the utilization surrogate models for the different tasks. What is anticipated is that future research in surrogate models for FEM-based computations will focus more on the development of automated tools for the selection and construction of surrogate models, as well as on the efficient use of the ensambles of surrogates and multi-fidelity models, based on where in the three discussed classes the particular application is located. Another expected trend is that future analyses will concentrate further on decreasing computational cost related to deriving surrogate models and on improving their interpretability.

2.2.2 | Combining Lipschitz and RBF surrogate models for high-dimensional computationally expensive problems

In the paper [A8], we proposed a surrogate model based on a Lipschitz underestimation and use it to develop a differential evolution-based algorithm. The algorithm, called Lipschitz Surrogate-assisted Differential Evolution (LSADE), utilizes the Lipschitz-based surrogate model, along with a standard radial basis function surrogate model and a local search procedure.

In the paper [A8], we used a combination of two surrogate models. The first one was built by the radial basis function (RBF) methodology. RBFs compute a weighted sum of prespecified simple functions to approximate complex design landscape. Given t different sample points $X_1, \ldots, X_t$, the RBF surrogates are written as

$$f_{\text{RBF}}(x) = \sum_{i=1}^{t} w_i \psi(||x - X_i||_2),$$

where w_i denotes the weight which is computed using the method of least squares, and ψ is the chosen basis function. There are several (symmetric) radial functions that can serve as a basis function, such as Gaussian function, thin-plate splines, linear splines, cubic splines, and multiquadrics splines.

The second surrogate model was based on the concept of Lipschitz continuity. The use of a the Lipschitz continuity concept in optimization was first proposed in the Shubert-Piyavskii method (or Sawtooth method) [127, 150] and initiated a line of research within global optimization that is active to this day [103]. We assume that the unknown (and expensive to compute) objective function f has a finite Lipschitz constant k, i.e.

$$\exists k \geq 0 \text{ s.t. } \; |f(x) - f(x')| \leq k||x - x'||_2 \; \forall (x, x') \in \mathcal{X}^2,$$

which is one of the weakest regularity assumptions one can ask for. Based on a sample of t evaluations of the function f at points $X_1, \ldots, X_t$, a global underestimator f_L of f can be constructed by using the following expression [103]

$$f_L(x) = \max_{i=1,\ldots,t} f(X_i) - k||x - X_i||_2. \tag{2.2.1}$$

The visual representation of the Lipschitz-based surrogate function in 1D is depicted in Figure 2.1. In the figure, each already evaluated point has two lines (one to the left and the other to the right) emanating from it under an angle that depends on the Lipschitz constant k. The surrogate is constructed as the pointwise maximum of these individual lines. A 2D visualization is shown in Figure 2.2. The Lipschitz-based surrogate has two important properties. Firstly, it assigns low values to points that are far from previously evaluated points. It also combines it with

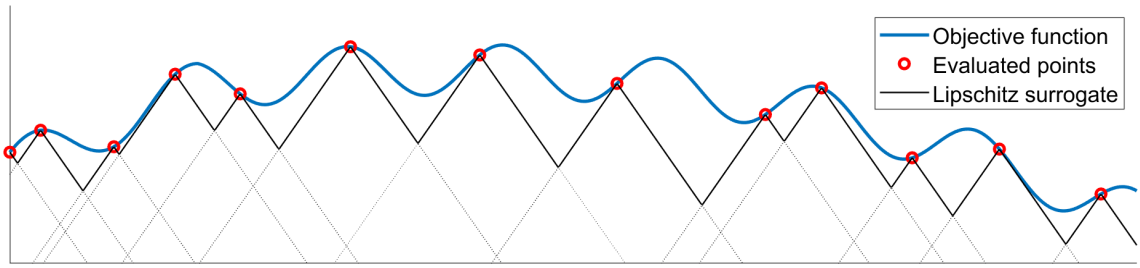


Fig. 2.1: Visual representation of the Lipschitz-based surrogate in 1D [A8].

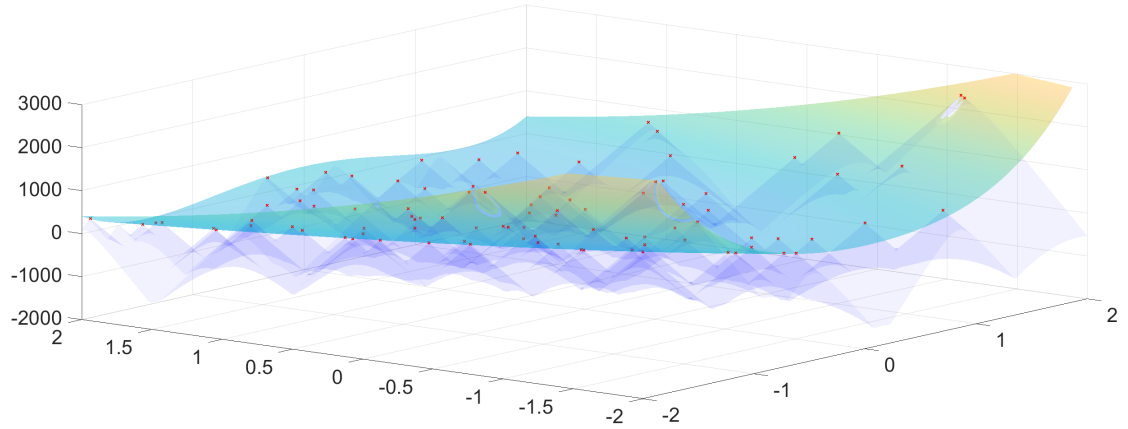


Fig. 2.2: Visual representation of the Lipschitz-based surrogate on the Rosenbrock function in 2D. Sampled points are highlighted in red and the Lipschitz-based surrogate in light blue [A8].

the information (objective value and "global" Lipschitz constant) from the closest evaluated point. This means that it can serve as a good "uncertainty measure" of prospective points for evaluation, as points with low values of f_L are either far from any other evaluated solution, or relatively close to a good one.

The proposed LSADE method consists of four distinct parts: 1) the DE-based generation of prospective points, 2) the global RBF evaluation of the prospective points, 3) the Lipschitz surrogate evaluation of the prospective points, and 4) the local optimization within a close range of the best solution found so far. The execution of parts 2) – 4) of the algorithm is controlled based on chosen conditions, i.e., one may sometimes skip RBF surrogate evaluation, Lipschitz surrogate evaluation, or local optimization, if either of them deemed to be advantageous for the search.

First, Latin hypercube sampling is employed to generate the initial population of t individuals, whose objective function is then evaluated. The best individual is found, a parent population of size p is randomly selected from the evaluated points and a new population is constructed based on the DE rules. If the RBF evaluation condition is true, the new population is evaluated based on the RBF surrogate model. Then the best individual based on the RBF model has its objective function evaluated and is added to the whole population. This step constitutes the global

Algorithm 1 Pseudocode of the LSADE [A8].

1: Generate an initial population of t points $X_1, \ldots, X_t$ and evaluate their objective function values. Denode the best solution as X_b.
2: Set $iter = 0$ (iteration counter), $NFE = t$ (number of function evaluations).
3: Use the evaluated points so far to estimate k and to construct the RBF surrogate.
4: Sample p points from the population as parents for DE.
5: Based on the DE rules, generate children.
6: Increase *iter* by 1.
7: **if** *RBF condition* **then**
8: Evaluate the children on the RBF surrogate.
9: Find the child with the minimum RFB surrogate value, and add it to the population and evaluate its objective function value. Increase NFE by 1.
10: **if** *Lipschitz condition* **then**
11: Evaluate the children on the Lipschitz surrogate.
12: Find the child with the minimum Lipschitz surrogate value, and add it to the population and evaluate its objective function value. Increase NFE by 1.
13: **if** *Local Optimization condition* **then**
14: Construct a RBF local surrogate model using the best c solutions found so far.
15: Find the bounds in each dimension for the local optimization.
16: Minimize the local RBF surrogate model within the bounds. Denote the minimum as $\hat{X}_m$ and, if it is not already in the population, add it to the population and evaluate its objective function value. Increase NFE by 1.
17: Find the best solution so far and denote it as X_b.
18: **if** $NFE < NFE_{max}$ **then**
19: **goto** *3*.
20: **else**
21: **terminate**.

search strategy.

If the Lipschitz evaluation condition is true, the Lipschitz constant k is estimated and the new population is evaluated on the Lipschitz surrogate model. The best individual based on the Lipschitz surrogate model has its objective function evaluated and is added to the whole population.

If the Local optimization condition is true, a local RBF surrogate model is constructed using the best c solutions found so far, which we denote by $\hat{X}_1, \ldots, \hat{X}_c$. Additionally, we find the bounds for the local optimization procedure within those

c points:

$$lb(i) = \min_{j=1,\ldots,c} \hat{X}_j(i), \quad i = 1, \ldots, D,$$
$$ub(i) = \max_{j=1,\ldots,c} \hat{X}_j(i), \quad i = 1, \ldots, D,$$

and perform a local optimization of the local RBF model within the bounds $[lb, ub]$. For local optimization we adapt a sequential quadratic programming strategy, which was also used by the winner of the 2020 CEC Single Objective Bound Constrained Competition [140]. We find the local optimum and check, if it is not already in the population, before evaluating it and adding it to the population.

The evaluation of points based on the Lipschitz-based surrogate model can be thought of as an exploration step of the method (and is expected to increase our ability to find the regions where "good" solutions might be found). On the other hand, the evaluation of points based on the local optimization procedure can be thought of as an exploitation step of the algorithm (and is expected to give us the means to improve the best solutions we have found so far).

The cycle of generating new population, evaluating it on the RBF and Lipschitz surrogate models and conducting the local optimization is carried out until a maximum number of objective function evaluations is reached. The pseudocode1 for the LSADE method is described in Algorithm 1.

In the experimental evaluation of the proposed method, LSADE was compared with six other state-of-the-art algorithms on a testbed of standard benchmark functions in dimensions $D = [30, 50, 100, 200]$. We also investigated the advantages of the individual components of the method, the choice of the conditions for using these components, the choice of basis functions for the RBF surrogates, and the computational complexity of the individual parts of the method. The computational results showed its effectiveness and competitiveness with other state-of-the-art algorithms, especially for complicated and high-dimensional problems. The LSADE method was also investigated on the benchmark set of the ICSI'2022 competition [95].

2.3 | Mathematical Programming Methods

Mathematical programming methods are specialized techniques that are extremely effective for problems with a known (exploitable) structure. They are designed to utilize function attributes such as convexity, linearity, or unimodality. Knowing that such a structure is present in a given problem opens the door for an extremely useful analysis. Theorems about the model structure, including properties pertaining to feasibility, or redundancy and theorems about the form of a solution (including whether one exists) can be proved. Convergence of algorithms can be analyzed. Various approximation arising from imperfections of model forms, levels of aggregation, computational error, and other deviations can be meaningfully investigated. With such structure, even extremely large problems (having billions of variables [179]) then can be successfully tackled.

The field of mathematical programming methods is incredibly rich - every bit of an exploitable structure has a corresponding method that tries to take advantage of it. An extremely comprehensive source of the most used methods is the book [117] - it covers large-scale mathematical programming techniques (such as interior-point or limited-memory methods), and the role of partially separable functions and automatic differentiation. It also includes a thorough discussion of topics such as Newton (and quasi-Newton) methods, constrained optimization theory, nonlinear least squares and nonlinear equations, penalty and barrier methods for nonlinear programming, the simplex method, trust-region methods, and sequential quadratic programming.

The author's area of research (in the mathematical programming methods) was in algorithms for the so-called stochastic programming problems [149], in which one has to deal with uncertain parameters in the problem data. We can model the decision making as specifying an objective function $F(x, \xi)$, that depends on a decision vector $x \in \mathcal{R}^{n_x}$ and vector $\xi \in \mathcal{R}^{n_\xi}$ of uncertain parameters, and minimizing $F(x, \xi)$ over x, which is restricted to the feasible set $x \subseteq \mathcal{R}^{n_x}$. However, this optimization problem is not well defined because our objective depends on an unknown and uncertain value of ξ. One possibility is to optimize the expected value of the objective function value. We assume that ξ is a random vector, with known probability distribution $\mathcal{P}$ with a support $\Xi \subseteq \mathcal{R}^{n_\xi}$ and formulate the following optimization problem:

$$\begin{aligned} &\text{minimize} \;\; f(x) = E_{\mathcal{P}}[F(x, \xi)] \\ &\text{subject to} \;\; x \in \mathcal{X}, \end{aligned} \tag{2.3.1}$$

where it is assumed that the considered expectations are well defined [93]. Another optimize a weighted sum of the expected value and a term representing variability

of the second-stage objective function. For example, one can instead minimize

$$f(x) = E[F(x, \xi)] + c\text{Var}[F(x, \xi)],$$

where $c \geq 0$ is a chosen constant, which was used in the Markowitz portfolio selection [136]. The additional (variance) term in the expression above can be viewed as a risk measure of the second-stage (optimal) outcome - adding the variance term may destroy convexity of the function f even if $F(x, \xi)$ is convex for all realizations of ξ [158].

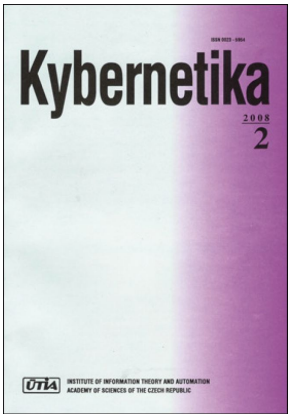
The formulation (2.3.1) can be applied to the so-called two-stage stochastic programming problem with recourse [11], where this optimization problem is divided into two decision stages - the first stage (also called the planning stage) where a decision has to be made on the basis of some available information, and the second stage (also called the operational stage) in which the particular realization of the uncertain data becomes known. This stochastic programming problem can be written in the form (2.3.1) where the optimal value of the second-stage problem is included in $F(x, \xi)$. These two-stage (or even multi-stage) can be approached by approximating the distribution of ξ and reformulating them into "standard" optimization problems. The caveat is that the problem size (measured in the number of variables and constraints) increases (sometimes quite dramatically) as the approximation of the distribution becomes more refined. The currently most-used approaches for such problems use variants of Bender's decomposition [182], progressive hedging [135], and stochastic dual dynamic programming [148].

One think to emphasize is that in the formulation (2.3.1) the uncertainty is concentrated in the objective function while the feasible set Ξ is assumed deterministic. However, in problems the feasible set itself can be delimited by constraints which depend on uncertain data. One approach is to formulate such problems in the form (2.3.1) by introducing penalty for any possible infeasibility. Another approach is to enforce satisfying constraints for all values of the unknown parameters in a chosen (uncertain) region. This approach is known as robust optimization [13]. This field experienced a significant increase in interest, building upon the advances in convex, conic, and semidefinite programming [14].

Requiring the satisfaction of the constraints for all possible values of the uncertain parameter can result in a solution that is too conservative. An alternative is try to satisfy the constraints with a high probability instead. This leads to the chance, or probabilistic, constraints approaches [38]. The currently most-used approaches for chance constrained problems utilize convex reformulations (also called the Bernstein approximation) [115], mixed-integer reformulations [1, 121], and the theory of probabilistic robust design [34, 35].

Stochastic programming is now a mature field with vast array of applications [168] spanning energy [167], transportation/logistics [129], nurse scheduling [9], portfolio selection [106], or disaster management [119]. The author of this thesis also has substantial experience of applying the stochastic programming framework to real-world problems, mainly in the area of network design and strategic decision-making for waste management problems [71, 154, 91]. Two of these applications [A10] and [A12] will be discussed in detail in the last section of the thesis.

In this section, two IF journal papers which are concerned with the algorithms for certain stochastic programming problems are included in the thesis in order to demonstrate the author's contribution to the field:



[A9] J Kůdela, P Popela. Warm-start cuts for generalized benders decomposition. Kybernetika, 53(6):1012–1025, 2017.

Author's contribution: 70%.

Metrics: $\text{IF}_{2017} = 0.632$.

Ranking (JCR 2017 WoS): Q4 - computer science, cybernetics.

[A11] J Kůdela. Pool & discard algorithm for chance constrained optimization problems. IEEE Access, 8:79397–79407, 2020.

Author's contribution: 100%.

Metrics: $\text{IF}_{2020} = 3.367$.

Ranking (JCR 2020 WoS): Q2 - computer science, information systems; Q2 - engineering, electrical & electronic; Q2 - telecommunications.

2.3.1 | Warm-start Cuts for Generalized Benders Decomposition

The paper [A9] focuses on a variation of the Benders decomposition [20] that was originally developed for solving mixed-variables programming problems. Into was further generalized for nonlinear convex problems and named the Generalized Benders Decomposition (GBD) [58]. The GBD method found its main use as a solution technique for mixed-integer nonlinear problems. In stochastic programming linear separability of the objective function and constraints is a common property - the two-stage problems can be often separated into the functions concerning only the first- and the second-stage variables. In the paper [A9], we assumed that in the problem under consideration the uncertainty was in the form of S scenarios $\xi^k \in \Xi$ with probabilities $\mathcal{P}\{\xi = \xi^k\} = p(\xi^k) > 0$, and that the problem had the following form:

$$\begin{aligned} \text{minimize} \quad & f_1(x) + \sum_{k=1}^{S} p(\xi^k) f_2(y_k, \xi^k) \\ \text{subject to} \quad & g_{1i}(x) \leq 0, \quad i = 1, \ldots, m_1 \\ & a_j(\xi^k)^T x + g_{2j}(y_k, \xi^k) \leq 0, \quad \xi^k \in \Xi, \quad j = 1, \ldots, m_2, \end{aligned} \tag{2.3.2}$$

where $f_1 : \mathcal{R}^{n_x} \to \mathcal{R}$ is a convex function, all m_1 constraint functions $g_{1i} : \mathcal{R}^{n_x} \to \mathcal{R}$ are convex, and for all $\xi^k \in \Xi$ (with $|\Xi| = S$), all m_1 constraint functions $g_{2j}(\cdot, \xi^k) : \mathcal{R}^{n_y} \to \mathcal{R}$ are convex, $a_j(\xi^k)$ is a vector for $j = 1, \ldots, m_2$, and the function $f_2(\cdot, \xi^k) : \mathcal{R}^{n_y} \to \mathcal{R}$ is convex. For this problem, we derived the corresponding master problem and subproblems (from the GBD framework).

We introduced a reformulation of the master problem that included bounds obtained from two different problems - the wait-and-see solution (where all scenarios are treated and optimized separately), and the expected value solution (where all random variables are replaced by their expected values). The idea was to include such a valid lower bound to the algorithmic procedure to "jumpstart" it and by doing so save on iterations (and the overall computational effort and time).

Another extensions in the form of bunching and multicut strategies were also investigated. Bunching is a technique in which we use "bunches" of scenarios and decompose the original problem alongside these bunches. The multicut formulation develops one cut for every second-stage problem (i.e., for every scenario) instead of the aggregated cut that the original method uses. The usefulness of the developed theoretical concepts (and the different extensions and variations) was numerically tested on a pair of two convex two-stage problems. The results showed that proposed warm-start formulation for GBD perform well for solving medium-sized convex two-stage stochastic problems and that especially the bunching ideas and modifications produce fruitful results.

2.3.2 | Pool & Discard Algorithm for Chance Constrained Optimization Problems

The article [A11] described a new method for handling chance constrained optimization problems [130]. The problem structure was the following. Let $\mathcal{X} \subseteq \mathcal{R}^{n_x}$ be a convex and closed domain of optimization and consider a family of constraints $x \in \mathcal{X}_\xi$ parameterized in $\xi \in \Xi$. The uncertain parameter ξ again describes different instances of an uncertain optimization scenario. We also have a probability measure $\mathcal{P}$ that describes the probability with which the uncertain parameter ξ takes value in Ξ. Then, a chance constrained optimization program can be written as:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } \mathcal{P}\{\xi : x \in \mathcal{X}_\xi\} \geq 1 - \varepsilon, \\ &\qquad\qquad x \in \mathcal{X}, \end{aligned} \tag{2.3.3}$$

where the linearity of the objective function can be assumed without loss of generality. The prototype optimization problem consists in minimizing a linear objective $c^T x$, subject to that x satisfies the constraints $g(x, \xi) \leq 0, \forall \xi \in \Xi$, where $g(x, \xi) : \mathcal{X} \times \Xi \to [-\infty, \infty]$ is a scalar-valued function that specifies the constraints. Considering only scalar-valued constraint functions can be assumed without loss of generality, as multiple constraints $g_1(x, \xi) \leq 0, \ldots, g_m(x, \xi) \leq 0$ can be expressed by a single scalar-valued constraint by aggregation $g(x, \xi) = \max_{i=1,\ldots,m} g_i(x, \xi)$. Although convexity is preserved by this operation, other valuable properties (such as linearity or differentiability) can be lost. In most situations, Ξ has infinite cardinality (i.e., contains an infinite number of possible realization of ξ). We assume that for each $\xi \in \Xi$, the sets $\mathcal{X}_\xi$ are convex and closed.

In this formulation (2.3.3), constraint violation is tolerated, but the violated constraint set must be no larger than ε. The parameter ε gives us the ability to trade robustness (i.e., the probability of constraint violation) for performance (i.e., improved optimal objective function value). The the optimal objective function value of (2.3.3) is a decreasing function of ε and provides the quantification of this trade-off.

In this setting, an important concept is the probability of violation of x which is defined as

$$\mathcal{V}(x) = \mathcal{P}\{\xi \in \Xi : g(x, \xi) > 0\}.$$

A solution x with small associated $\mathcal{V}(x)$ is feasible for most of the problem instances. We also say that a solution x is ε-level robustly feasible (or approximately feasible) if $\mathcal{V}(x) \leq \varepsilon$. The main goal is to devise an algorithm that returns a ε-level solution (with any fixed small reliability level ε). The approach that was utilized in the paper

[A11] was based upon a surrogate model called the "Scenario Design Problem". In the "scenario design" one optimizes the objective subject to a finite number of randomly selected scenarios. It is assumed that S independent identically distributed samples $\xi^1, \ldots, \xi^S$ are drawn according to probability $\mathcal{P}$. A scenario design problem can then be written as the convex program

$$\begin{aligned} \text{minimize } & c^T x \\ \text{subject to } & g(x, \xi^i) \leq 0, \quad i = 1, \ldots, S \\ & x \in \mathcal{X}, \end{aligned} \tag{2.3.4}$$

with the assumption that for all possible extractions $\xi^1, \ldots, \xi^S$, the optimization problem (2.3.4) is either infeasible, or, if feasible, it attains a unique optimal solution. The scenario problem (2.3.4) is a "standard" convex optimization problem with a finite number of constraints and, hence, its optimal solution should (usually) be efficiently computable by the means of mathematical programming algorithms [24].

The relationship between the number of sampled scenarios S and the probability of violation of the optimal solution to corresponding Scenario Design Problem was derived in [33]. It was found that for a chosen ε we can always find S that is large enough such that the solution to (2.3.4) is ε-level feasible for the original problem (2.3.3) with arbitrarily high confidence. However, there is no guarantee, that the resulting optimal objective value of (2.3.4) will be anywhere close to the true optimal value of (2.3.3).

A pivotal concept in the development of the proposed Pool & Discard (P&D) algorithm was the so-called "support scenario". A scenario $\xi^i, i \in \{1, \ldots, S\}$ is a support scenario for the scenario problem (2.3.4) if its removal changes the optimal solution of (2.3.4). It was shown in [33] that the number of support scenarios for (2.3.4) is at most n_x (i.e., the size of x), and it does not depend on S The first main contribution of the paper [A11] is an efficient way of solving (2.3.4) using this result.

The main idea behind the Pooling part of the algorithm is the following. If one were to verbally describe the problem (2.3.4), the word that probably first comes up is "long", as there are much more constraints than decision variables. Moreover, the number of support constraints (or support scenarios), that the optimal solution of (2.3.4) depends upon is very small, at least when compared to the overall number of constraints.

The method consists of solving (2.3.4) by the following procedure. First, we start by completely neglecting the constraints in (2) that correspond to the different scenarios and solve this relaxed optimization problem. Then we find the most violated constraints (by computing the slacks), add them to the relaxed problem and find a new optimal solution.

The Pooling part can be summarized as follows:

Step 0. Set $\mathcal{I} = \emptyset$.

Step 1. Solve the following problem:

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & g(x, \xi^i) \leq 0, \quad i \in \mathcal{I}, \end{aligned} \tag{2.3.5}$$

and obtain a solution $\hat{x}$.

Step 2. Check feasibility of the solution by computing the slacks s^i:

$$s^i = g(\hat{x}, \xi^i), \quad i \in \{1, \ldots, S\}. \tag{2.3.6}$$

Step 3. If $\max\limits_{i \in \{1,\ldots,S\}} s^i > 0$, find the associated index of the maximum value $\hat{i} = \operatorname{argmax}\limits_{i \in \{1,\ldots,S\}} s^i$, add it to the set $\mathcal{I}$ and return to Step 1. Otherwise, set $x^* = \hat{x}, \mathcal{I}^* = \mathcal{I}$ and terminate.

Once this procedure end, we get the optimal solution of (2.3.4), and also an index set $\mathcal{I}$ that contains the support scenarios. This index set is very significant for the success of the Discarding part of the P&D algorithm.

However, if we were to enforce all the S constraints we cannot expect to obtain good approximations of chance constrained solutions. To get a less conservative solution we utilized the framework introduced in [34], that allowed us to remove k constraints out of the S scenario constraints. A general removal procedure can be formalized by the following definition: Let $k < S$. An algorithm $\mathcal{A}$ for constraints removal is any rule by which k constraints out of a set of S constraints are selected and removed. The output of $\mathcal{A}$ is the set $\mathcal{A}\{\xi^1, \ldots, \xi^S\} = \{i_1, \ldots, i_k\}$ of the indexes of the k removed constraints. The sample-based optimization program where k constraints are removed as indicated by $\mathcal{A}$ can be written as

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & g(x, \xi^i) \leq 0, i \in \{1, \ldots, S\} \setminus \mathcal{A}\{\xi^1, \ldots, \xi^S\} \\ & x \in \mathcal{X}, \end{aligned} \tag{2.3.7}$$

There is an additional assumption which requires that the algorithm $\mathcal{A}$ chooses constraints whose removal improves the solution by violating the removed constraints, and it rules out for example algorithms that remove inactive constraints only, or algorithms that remove constraints at random. Such assumption is very natural and reflects the fact that we want to remove the constraints that improve the optimal objective value.

The Discarding part of the proposed P&D algorithm consists of utilizing the index set $\mathcal{I}$, finding the support scenarios among this set and finding the one scenario, whose removal decreases the optimal objective value the most. This procedure is repeated k times, where k is either set a priori, or is terminated once an estimate

of the probability of violation of obtained solution $\mathcal{V}(x)$ reaches certain threshold. Although this approach is in principal similar to the one discussed in [122] (called greedy constraint removal), our algorithm utilizes the Pooling step and uses warm-starts (primarily utilizing $\mathcal{I}$) throughout the iterations and as such can be rather effective (which is demonstrated in the numerical/experimental sections of the paper). The P&D algorithm can be summarized as follows:

Step 0. Solve the pooling part described above to obtain $\mathcal{I}^*$ and x^*. Set $\gamma > 0, k > 0, \mathcal{I}_p = \emptyset$.

Repeat k times, or terminate once an estimate of $\mathcal{V}(x^*)$ reaches a threshold:

Step 1. Find the set of support scenarios $\mathcal{I}_r \subset \mathcal{I}^*$ – either by examining the slacks ($s^i > -\gamma$) or the associated dual variables ($\mu^i > \gamma$).

Step 2. For each of the support scenarios $i_r \in \mathcal{I}_r$, solve the following problem:

$$\begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} && c^T x \\ &\text{subject to} && g(x, \xi^i) \leq 0, \quad i \in \{1, \ldots, S\} \setminus \{i_r \cup \mathcal{I}_p\}, \end{aligned} \tag{2.3.8}$$

using the Pooling part, warm-started by using $\mathcal{I} = \mathcal{I}^* \setminus \{i_r\}$ and $x = x^*$. Denote the solution to (2.3.8) as $x^*_{i_r}$, its optimal objective function value $v^*_{i_r}$ and its final set of scenarios $\mathcal{I}^*_{i_r}$.

Step 3. Find the index with the best optimal objective value: $i^* = \underset{i_r}{\operatorname{argmin}}\, v^*_{i_r}$. Set $x^* = x^*_{i^*}$, $\mathcal{I}^* = \mathcal{I}^*_{i^*}$ and add the corresponding scenario to the set of permanently discarded ones $\mathcal{I}_p$.

The parameter γ can be, in theory, set to 0 – what discourages us from doing so are the implementation issues of numerical computing that are further discussed in the paper.

The numerical examinations show that the proposed P&D algorithm provides a powerful framework for handling certain types of chance constrained optimization problems. The exploitation of the problem structure and efficient implementation allows us to considerably speed up the computations, especially for large instances, when compared with conventional methods. When compared with conventional approaches [122] on a linear example, it was several hundred times more efficient in the largest instances. On the nonlinear examples, it was on par with the specialized state-of-the-art methods [147] and [1]. The pooling techniques developed in [A11] and the batching techniques used in [A9], were also utilized in another paper of the author of this thesis [88], that improved the efficiency of the Wolfe-Atwood algorithm for the minimum-volume covering ellipsoid problem [24].

2.4 | Benchmarking Optimization Methods

Over the years, the number of different optimization methods (and the various implementations of these methods) have increased substantially [12]. Comparative studies of different methods can provide a great value to the practitioners by helping them to choose the most fitting optimization method for their specific problem. This comparison is generally called optimization benchmarking. In its general sense, benchmarking constitutes a comparison of one (or possibly more) products to an industrial standard product over a collection of performance metrics [12]. In the context of benchmarking optimization methods, the "products" are the different implementations of selected methods, and the "performance metrics" are computed by running these implementations on collections of test problems.

Such a framework can give a bit of clarity into benchmarking optimization methods, because there is (at least to an extent) an agreement on what it means for one method to be "better" than another. If one method gives a better final objective function value, uses less memory, or runs significantly faster, on all considered problems, then it can be seen as better than the alternative. Naturally, in practice such a clear conclusion can be found only very rarely [12].

When done properly, benchmarking optimization methods can provide immense practical value, as it can uncover certain strengths and weaknesses of different methods. On the other hand, when performed poorly, benchmarking optimization methods can also lead to misleading conclusions. It can hide various weaknesses (or strengths) of different methods, report improvements that do not actually exist, and even suggest objectively incorrect algorithmic choice for certain problems.

A crucial part in benchmarking is the selection of test sets (also called benchmark sets, or benchmark suits). A test set is a collection of test problems, which contain a test (or benchmark) function, possibly with some further criteria such as the feasible domain, or constraint set. Benchmarking can yield meaningful insights only when the different methods are evaluated on the same benchmark set with using same performance measures. There are three main sources for the test problems: randomly-generated problems, artificially-generated problems, and real-world problems. The real-world problems are found through instances of specific applications, while pre-generated problems exist in common test set libraries[12]. A great representative of a benchmark suite that uses both real-world and pre-generated problems is the MIPLIB set [84] that is extensively used by the integer programming community. For the QAP mentioned in earlier sections, QAPLIB [28] is also the standard (and expected) choice for comparing different methods.

Apart from benchmarking, the comparison of mathematical programming methods can also rely on theoretical techniques such as convergence analysis. However,

the metaheuristic methods, that are often used for black-box problems without any exploitable structure, such theoretical performance results are hard (or impossible) to develop. Especially the evolutionary computation (EC) methods rely heavily on benchmarking for the development of novel search algorithms as well as in the assessment and comparison of contemporary algorithmic ideas.

There are two main lines of development in benchmarking for EC methods, the IEEE Congress on Evolutionary Computation (CEC) competitions and the Genetic and Evolutionary Computation Conference (GECCO), where the Black-Box Optimization Benchmarking (BBOB) workshop is held. The BBOB functions are a part of the COCO platform for comparing optimization [64]. One advantage of the COCO platform is that a large number of algorithm results are publicly available for comparison. Currently, more than 230 distinct (classical, contemporary, mathematical programming based, and EC) algorithms have been tested on the COCO suite. On the other hand, the competitions that are organized every year during the CEC aim to compare state-of-the-art stochastic search algorithms. The CEC competitions provide a specific test environment for algorithm assessment and comparison. As was shown in [56], the characteristics of the functions used in these two benchmark suites are very different. Interestingly, the CEC benchmark functions are mainly composed of similar subfunctions, which possibly gives an advantage to methods that perform well on these fewer subfunctions. Also, it was found that the CEC functions share more similarities among themselves than with those found in the BBOB [56]. An extremely useful feature of both the COCO platform and the repository of the CEC competitions is the inclusion of methods and corresponding data. In the repository of the CEC competitions, the best-performing algorithms are shared along with their results. On the COCO platform, the results of different methods can be found. However, some authors voiced their critique about the artificial nature of these benchmark sets [125], as a vast majority of the test problems included in them are artificially-generated, and advocated for testing EC methods on real-world problems instead [162].

In the global (deterministic) optimization community, one of the most popular benchmark sets is the one produced by the GKLS generator [57]. The GKLS generator constructs classes of test functions (either non-differentiable, differentiable, or twice-differentiable) for multi-modal, multi-dimensional box-constrained global optimization. The advantage of the GKLS generator is that for each generated problem, the location and function value of its local and global minima are known. Although the GKLS generator can be used to create various types of problems (based on input parameters), there are 8 classes of problems (2 for dimensions 2, 3, 4, and 5) each containing 100 functions that are generally used [145, 144, 123, 156]. The GKLS generator was also recently used for the construction of nonlinear model predictive

control [175] and general-constrained [146] test problems.

In benchmarking EC methods on single-objective optimization problems, the performance measure is usually the mean error obtained from the different runs [99]. For multi-objective optimization problems the more adequate measures differ [67]. Other performance indicators, such as running time or memory footprint, can also give us some interesting insights on the behavior of the compared methods. However, these should be carefully considered as such measures have hidden biases that depend on external factors and can hinder the comparison, such as the chosen programming language, or a particular implementation.

For a comparison of several algorithms to be fair, they all should conduct a similar effort in finding the best possible solution [99]. This is usually done by setting a common stopping criterion, such as the maximum number of objective function evaluations. The choice and analysis of different stopping criteria is also an important part of benchmarking, as different algorithms generally have different convergence rates, and the results of the comparison will often substantially differ depending on the "checkpoint" at which they were evaluated.

In benchmarking, a principled validation procedure is just as important as the selection of the benchmark set [99]. Two different tools are widely used for this purpose: statistical analysis and comparative visual analysis. Once the appropriate benchmark set has been selected and the algorithms were run, the statistical comparison can be carried out. Usually, the ranking of each algorithm over the benchmark set is computed, and the significance of the differences in the ranking are tested. The most used method for this analysis is the Friedman rank-sum test [43], which is a non-parametric method that does not have many restrictive assumptions (such as normality).

Visualization techniques are another useful tool for reporting results when comparing optimization algorithms. Their main advantage (over reporting raw data in tables) is in their much easier interpretation and their ability to summarize complex data and relationships. A typical visual representation is that of the convergence of an algorithm. In this case, the variable being discussed is the convergence speed (typically measured in the function value or error found in a given number of function evaluations) of the compared methods. The previously mentioned COCO platform has an integrated procedures for generating visualization of the results, such as expected running time or empirical cumulative distribution of the number of objective function evaluations for different error targets [64]. Another great profiling tool for optimization heuristics of the results that we used in several papers is the IOHprofiler [48], which takes as input the benchmark data and provides very detailed analysis. Apart from the standard statistical tests and visualizations, IOHprofiler includes new approaches such as the Deep Statistical Comparison [53], or elo-based

Glicko-2 rating [164]. IOHprofiler is available as an R package, as well as a free online service.

In order to quantify the low-level properties of optimization problems, various features of the landscape can be computed [101]. Such analysis falls under the field of Exploratory Landscape Analysis (ELA) [110]. The landscape features try to approximate different aspects of the optimization problem, such as its modality, separability, or whether or not the problem has plateaus. The most notable uses of ELA are in the visualization of the problem space of various optimization benchmark problem sets [151], and in automated algorithm selection [80]. It was recently shown that the ELA features are sensitive to sampling strategy [133] and function transformations [153, 152]. The most used tool for computing the ELA features is the flacco library in R [81].

Three CORE-ranked conference papers and two IF journal papers are included in the thesis in order to demonstrate the author's contribution to the field:

[A2] J Kůdela. Novel zigzag-based benchmark functions for bound constrained single objective optimization. In 2021 IEEE Congress on Evolutionary Computation (CEC), pages 857–862. IEEE, 2021.

Author's contribution: 100%.

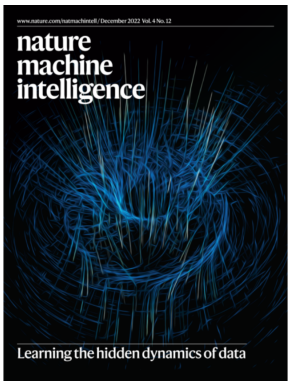
Ranking (Core 2021): B - Artificial intelligence.

[A6] J Kůdela, R Matoušek. New benchmark functions for single-objective optimization based on a zigzag pattern. IEEE Access, 10:8262–8278, 2022.

Author's contribution: 70%.

Metrics: $\text{IF}_{2021} = 3.476$.

Ranking (JCR 2021 WoS): Q2 - computer science, information systems; Q2 - engineering, electrical & electronic; Q2 - telecommunications.



[A3] J Kůdela. A critical problem in benchmarking and analysis of evolutionary computation methods. Nature Machine Intelligence, 4:1238–1245, 2022.

Author's contribution: 100%.

Metrics: $\text{IF}_{2021} = 25.898$.

Ranking (JCR 2021 WoS): D1 - computer science, artificial intelligence; D1 - computer science, interdisciplinary applications.

[A5] J Kůdela, M Juříček, R Parák. A collection of robotics problems for benchmarking evolutionary computation methods. In: Correia, J., Smith, S., Qaddoura, R. (eds) Applications of Evolutionary Computation. EvoApplications 2023. Lecture Notes in Computer Science, vol 13989. Springer.

Author's contribution: 75%.

Ranking (Core 2021): B - Artificial intelligence; Machine learning; Applied computing.

[A4] J Kůdela, M Juříček. Computational and Exploratory Landscape Analysis of the GKLS Generator. In 2023 GECCO.

Author's contribution: 90%.

Ranking (Core 2021): A - Artificial intelligence.

2.4.1 | Novel Zigzag-based Benchmark Functions for Bound Constrained Single Objective Optimization

In the paper [A2], we proposed novel zigzag-based benchmark functions for bound constrained single objective optimization, that are non-differentiable and highly multimodal. These new benchmark functions were constructed as follows. First, we devised a so-called "zigzag" function $z(x)$. For given parameters $k > 0, m > 0$ the zigzag function $z(x)$ at a point $x \in \mathcal{R}$ is computed as:

$$z(x) = \begin{cases} m\Big(\frac{1}{2} + (-1)^{\lceil kx \rceil}\big(\frac{\lceil kx \rceil + \lfloor kx \rfloor}{2} - kx\big)\Big), \text{if } (kx) \notin \mathcal{Z} \\ 0, \quad \text{if } \frac{kx}{2} \in \mathcal{Z} \\ m, \quad \text{otherwise,} \end{cases}$$

where $\frac{2}{k}$ is the period and m is the amplitude of the zigzag function. This function is depicted in Fig 2.3.

The next step was the construction of a multimodal function $f(x)$, which was devised as a sum of an absolute value of a high degree polynomial with one root in zero and an absolute value function. Finally, the two proposed benchmark functions $F_1(x)$ and $F_2(x)$, for $x = [x_1, \ldots, x_D]^T$ and $x \in [-100, 100]^D$, were the following:

$$\begin{aligned} f_1(x) &= \sum_{i=1}^{D} f(x_i) \\ F_1(x) &= f_1(M_1(x - s_1)) \\ f_2(x) &= \sum_{i=1}^{D} f(f(x_i)) \\ F_2(x) &= f_2(M_2(x - s_2)) \end{aligned}$$

where $s_1, s_2 \in [-100, 100]^D$ were random shifts of the optimal solution and M_1, M_2 were random rotation/scaling matrices, with eigenvalues in the range $[0.5, 1]$. The rotation/scaling matrices were constructed in the following way: for a given dimension D, a random square matrix A was generated, and a matrix $B = A'A$ was computed. Then, by using the singular value decomposition of B, we computed matrices P, R, Q, i.e. $B = PQR'$. Lastly, we generated a D dimensional vector v whose individual components are uniformly distributed random values on the interval [0.5,1], and we construct the matrix M as $M = P \cdot \text{diag}(v) \cdot R'$, where $\text{diag}(\cdot)$ transformed a vector into a diagonal matrix. This ensured that the eigenvalues of M lied on the interval [0.5,1]. The rotation/scaling matrix is an integral part of the benchmark function [120], as it creates additional difficulty for the optimization algorithms.

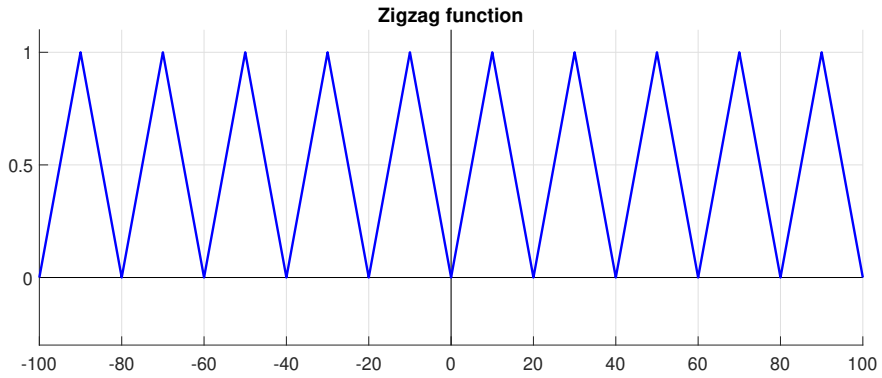


Fig. 2.3: Zigzag function $z(x)$ with $k = 0.1, m = 1$ [A2].

Next, we showed that different parametrizations of the proposed benchmark functions can be successfully used in benchmarking. For this purpose, we chose three algorithms: the canonical particle swarm optimization (PSO) [79], the Improved Multi-operator Differential Evolution (IMODE) algorithm [140] (which was the winner of the CEC'20 competition), the Adaptive Gaining-Sharing Knowledge (AGSK) [112] based algorithm (the runner-up of CEC'20). Our benchmark rules followed the ones from the CEC'20 competition. Overall, the results of the comparisons on both of the newly proposed benchmark functions showed that neither of the two best algorithms from the CEC'20 competition performed significantly better than a standard PSO.

2.4.2 | New Benchmark Functions for Single-Objective Optimization Based on a Zigzag Pattern

The paper [A6], we expanded on the work presented in [A2] and introduced four new benchmark functions for bound-constrained single-objective optimization that are based on a zigzag pattern, and are non-differentiable and highly multimodal. We took inspiration from the recently proposed tunable benchmark functions for combinatorial problems [170]. The newly proposed functions included three parameters that could change their behaviour and difficulty. We conducted extensive numerical experiments to investigate how the different parametrizations work as benchmarks and showed their successful utilization in algorithmic comparisons.

The proposed functions are constructed in the following way. First, a so-called "zigzag" $z(x, k, m, \lambda)$ (or triangular wave) function is constructed based on the following formula, where $\lfloor \cdot \rfloor$ is the floor operator:

$$z(x,k,m,\lambda) = \begin{cases} 1 - m + \frac{m}{\lambda}(|x|/k - \lfloor |x|/k \rfloor), & \text{if } |x|/k - \lfloor |x|/k \rfloor \leq \lambda, \\ 1 - m + \frac{m}{1-\lambda}(1 - |x|/k + \lfloor |x|/k \rfloor), & \text{otherwise,} \end{cases}$$

This extended zigzag function also contains three parameters: $k > 0$ that controls its period, $m \in [0, 1]$ that controls its amplitude, and $\lambda \in (0, 1)$ that controls the location of its local minima. The effect of the individual parameters on the shape of the zigzag function can be seen in Figure 2.4. For $m = 0$ the zigzag function is identically equal to 1 for any point x (regardless of the values of the other two parameters).

Next, we constructed four basic benchmark functions (F1-F4) that combined the zigzag function with different multimodal functions. Their construction starts with four 1-D functions $\phi_1, \ldots, \phi_4$, which are formulated in as:

$$\begin{aligned} \phi_1(x,k,m,\lambda) &= 3{\cdot}10^{-9}|(x-40)(x-185)x(x+50)(x+180)|z(x,k,m,\lambda) + 10|\sin(0.1x)| \\ \phi_2(x,k,m,\lambda) &= \phi_1(\phi_1(x,k,m,\lambda),k,m,\lambda) \\ \phi_3(x,k,m,\lambda) &= 3|\ln(1000|x|+1)|z(x,k,m,\lambda) + 30 - 30|\cos(\frac{x}{10\pi})| \\ \phi_4(x,k,m,\lambda) &= \phi_3(\phi_3(x,k,m,\lambda),k,m,\lambda) \end{aligned}$$

The common feature of the functions is that they are bounded on the interval [-200, 200] by a maximum value of 200. This allows us to compose the functions with themselves without running to numerical difficulties. There is also a single global minimum located at 0, with function value 0. Although the optimization will take place only on the interval [-100, 100] we will use a shift vector to change the placement of the optimal point (hence, the need for the functions to be "well-behaved" on [-200, 200]).

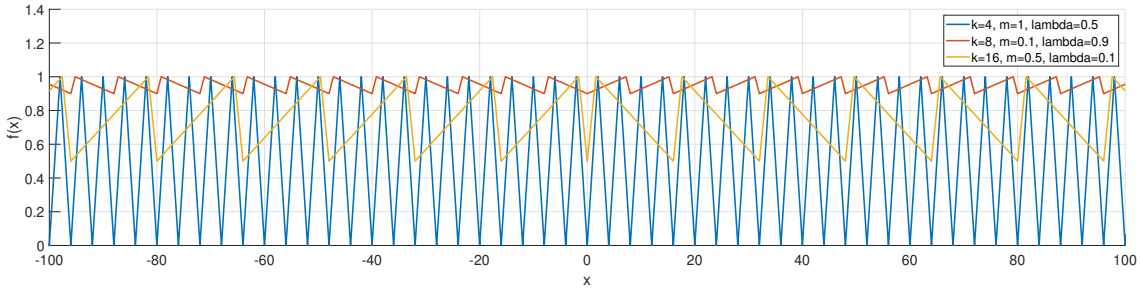


Fig. 2.4: Zigzag function for different values of the parameters [A6].

To obtain the benchmark functions for a dimension D, we used a simple sum of the functions ϕ for the individual components, but modified the inputs by a shift vector $s \in [-100, 100]^D$ and a rotation/scaling matrix M, which was constructed in the exact same manner as in [A2]:

$$f_j(x, k, m, \lambda) = \sum_{i=1}^{D} \phi_j(x_i, k, m, \lambda) \qquad j = 1, \ldots, 4$$

$$\mathrm{F}j(x, k, m, \lambda) = f_j(M_j(x - s_j), k, m, \lambda) \qquad j = 1, \ldots, 4$$

The shapes of the functions F1-F4 in one dimension for selected values of the parameters can be seen in Figure 2.5. The individual parameters of the zigzag function also serve secondary roles. The parameter k can be though of as a "ruggedness" parameter, that makes the zigzag more frequent. The parameter λ can be seen as a "deception" parameter, as low values of λ skew the function in such a way that its derivatives (if existing) point predominantly away from the global optimum. Higher values of the parameter m result in a "deeper" local optima. By setting these parameters to different values, we should be able to increase/decrease the difficulty of the resulting optimization problems, and to bring out the advantages and the disadvantages of different optimization methods.

Next, we investigated the behaviour of selected EC methods on the newly proposed benchmark functions. The selected methods were: CMAES [7], DE [44], PSO [79], AGKS algorithm [112], HSES [180], IMODE [140], LSHADE [160], and MadDE [18]. To investigate the impact of the different values of the parameters of the benchmark functions, we chose 100 parameter settings (for all functions F1-F4) for computational evaluations. For the individual computations, we used rules similar to the CEC'21 competition: the eight algorithms were evaluated on the four benchmark functions with 100 different parameter settings with $D = \{5, 10, 15\}$ dimensions.

Furthermore, we selected a few parametrizations of the proposed benchmark functions to create a benchmark set. This selection was done by carefully examining the results of the comparison of the algorithms and choosing, for each

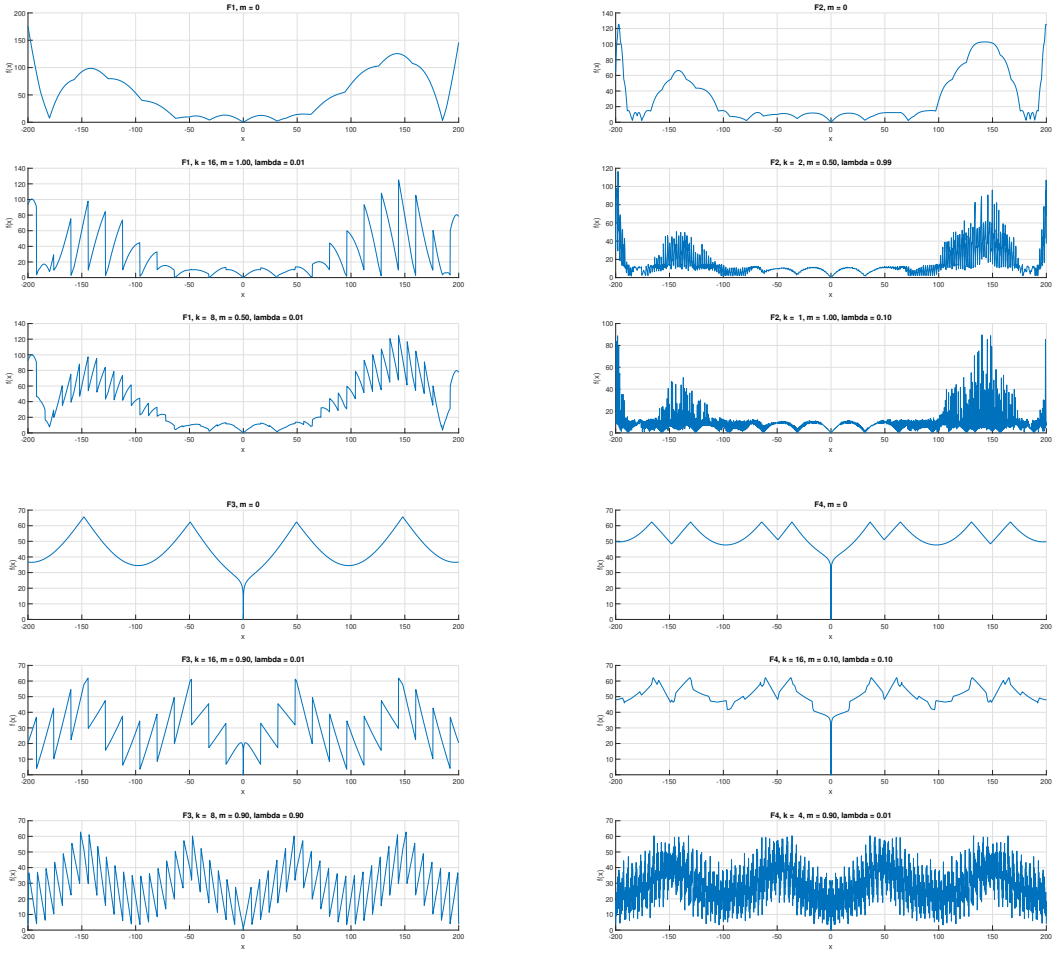


Fig. 2.5: Shape of the benchmark functions F1-F4 for different values of the parameters [A6].

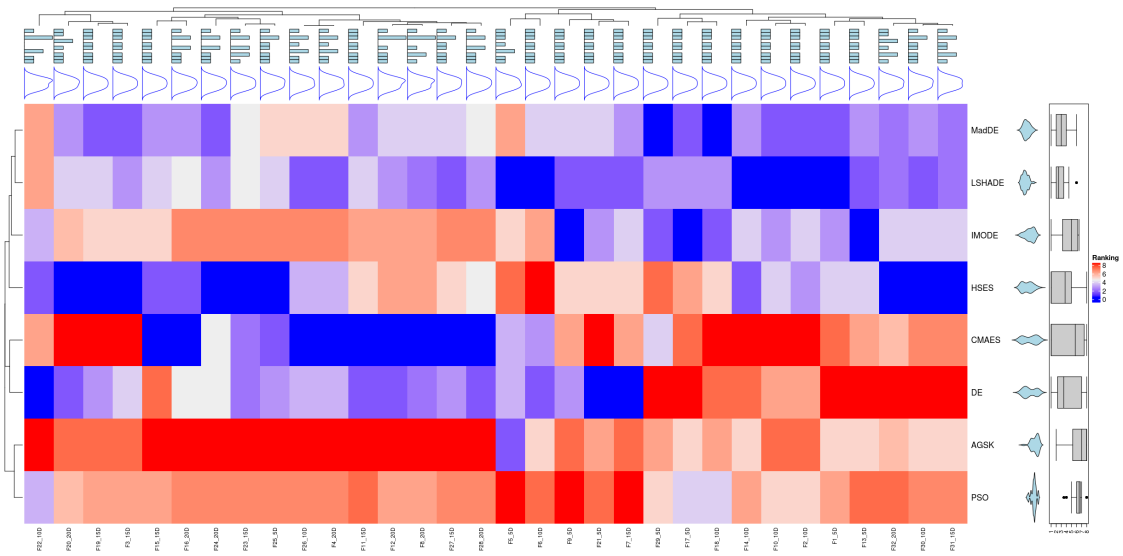


Fig. 2.6: Comparison of the algorithms on the ambiguous benchmark set [A6].

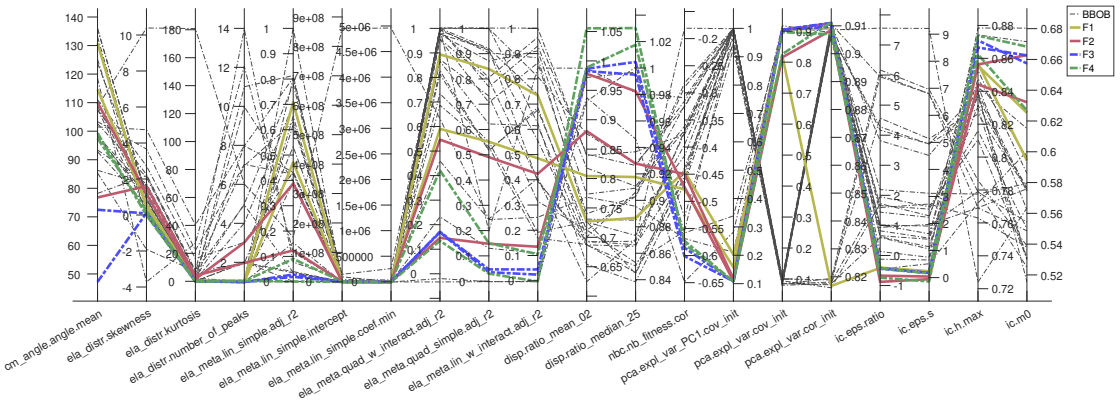


Fig. 2.7: Comparison of ELA measures between different parametrizations of F1-F4 and BBOB, $D = 10$ [A6].

benchmark function, two parametrization, that together resulted in the ambiguous benchmark set. We also decided to include additional dimension $D = 20$. This resulted in the creation of a benchmark set with 32 instances (four functions F1-F4, two parametrizations each, four different dimension), whose 1-D plots are shown in Figure 2.5. The results of the comparison of the eight chosen algorithms on the ambiguous benchmark set are presented in Figure 2.6 using the IOHprofiler [48] - for a particular problem, if a rectangle, that represents a certain problem (horizontal dimension) and a certain algorithm (vertical dimension), is bluer then the algorithm is better for the given problem (and, the redder it is, the worse is the algorithm). In the horizontal dimension, if the color of the rectangles remains the same, it signals that the ranking of the given algorithm remains the same across different problems.

From this comparison, it can be seen that most of the algorithms, apart from PSO and AGSK, were best-ranked for at least one of the problems in the set. Also, for at least one problem, all algorithms placed in the bottom half of the rankings. This signals that the chosen parametrizations cover a wide range of the single-objective optimization problem space (measured by the performance of different algorithms). However, it is clear from the results that some of the selected algorithms performed consistently better than others.

To better explore the problem space that is covered by the different parametrizations we computed different ELA features. The results of the ELA are summarized in Figure 2.7, where the parametrizations chosen for the ambiguous benchmark set with the benchmark functions from the BBOB (COCO) suite are shown. When looking at these results, we can see that particularly the ELA measures of the chosen F3 and F4 parametrizations fall into places where there are only a few BBOB counterparts.

2.4.3 | A critical problem in benchmarking and analysis of evolutionary computation methods

In the paper [A3], we showed that some of the frequently used benchmark functions have their respective optima in the centre of the feasible set and that this poses a critical problem for the analysis of evolutionary computation methods. As written in the introductory part of this section, two of the most popular benchmark sets for comparing EC methods are the BBOB set and the CEC competition sets. There is, however, another standard benchmark set that is extensively used for comparing optimization methods consists of some of the most well-known functions such as Ackley, Griewank, Rosenbrock, Rastrigin, or Schwefel. This set contains a small design flaw - a large portion of the functions found in this set have the optimum at a zero vector (or in the centre of the feasible set). This fact, on its own, does not pose a serious problem, but comparing methods that incorporate a "check-the-middle" procedure or a centre-bias (or zero-bias) operator on these benchmarks to other methods is essentially meaningless. This issue was also identified during the CEC 2021 competition.

The inclusion of the centre-bias operator is hardly ever plainly visible from the description of the method. This is both a problem of the often-used metaphor-rich jargon and the difficulty of the theoretical analysis of stochastic systems. Possible mechanisms leading to a centre-bias operator in metaheuristics are the contraction operator that was found in the Grey Wolf Optimization algorithm [116] and a search bias that was found in the Salp Swarm Optimization algorithm [36]. The centre-bias was also found in the Sooty Tern Optimization Algorithm and the Tunicate Swarm Algorithm [90].

In the paper [A3], we took a closer look at seven recently published methods in journals such as Applied Soft Computing, Information Sciences and Future Generation Computer Systems. We showed that all of the considered methods contained the centre-bias operator. We will carried out a fair comparison between these relatively new methods and two older ones — Differential Evolution (DE) and Particle Swarm Optimization (PSO). Additionally, we included in the comparisons two methods that performed well in the CEC competitions.

Out of the seven problematic methods, six performed the computational comparison on a very similar benchmark set, with thirteen of the considered problems summarized in Table 2.1. Of these thirteen problems, eight have their respective optimum at the zero vector. Four other problems have the optimum quite close to the zero vector and give very good results when evaluated at the zero vector. Only a single problem (F08) has the optimum located far from zero.

We use the following methodology to analyse the behaviour of the methods.

Tab. 2.1: First 13 benchmark functions dimension 30. U -– unimodal, M — multimodal, S — separable, N — non-separable, f^* – the optimal function value, $f(0)$ – function value at the zero vector, x^* – optimal solution.

ID	name	type	range	f^*	$f(0)$	$x*$
F01	Sphere	U, S	[-100,100]	0	0	[0,0,...]
F02	Schwefel 2.22	U, N	[-100,100]	0	0	[0,0,...]
F03	Schwefel 1.2	U, N	[-100,100]	0	0	[0,0,...]
F04	Schwefel 2.21	U, S	[-100,100]	0	0	[0,0,...]
F05	Rosenbrock	U, N	[-30,30]	0	2.90E+01	[1,1,...]
F06	Step	U, S	[-100,100]	0	7.50E+00	[-0.5,-0.5,...]
F07	Quartic with noise	U, S	[-1.28,1.28]	0	0	[0,0,...]
F08	Schwefel 2.26	M, S	[-500,500]	-1.25E+04	0	[420.9, 420.9,...]
F09	Rastrigin	M, S	[-5.12,5.12]	0	0	[0,0,...]
F10	Ackley	M, N	[-32,32]	0	0	[0,0,...]
F11	Griewank	M, N	[-600,600]	0	0	[0,0,...]
F12	Penalized1	M, N	[-50,50]	0	1.67E+00	[-1,-1,...]
F13	Penalized2	M, S	[-50,50]	0	3.00E+00	[1,1,...]

First, the different methods were evaluated on the thirteen benchmark functions without any additional modifications. The dimension of the problems was set to 30 and the maximum number of allowed function evaluations was set to 50,000. As the performance measure, we chose the mean error (difference between optimal function value and best function value found after the max iteration count) over 30 independent runs. The results showed that all the problematic methods (apart from PSO and DE) perform extremely well on the problems that have optimum at the zero vector (F01–F04, F07, F09–F11). Additionally, on the four problems that produce good values when evaluated at the zero vector (F05, F06, F12, F13) the methods performed very well. On F08, however, the performance of the problematic methods was poor - five of the seven performed worse than PSO.

Next, we examined the effects of modifying these benchmark problems. We introduce a shift operator, which moves the evaluated point by a predetermined vector s, that is, instead of the benchmark function being $f(x)$, it is modified to $f(x+s)$. Since the functions stay basically the same, one would expect the algorithms to perform consistently. The shift vector was chosen as 10% of the range in each dimension. The results showed that the supreme performance of the problematic methods vanished, while the performance of PSO and DE stayed roughly the same.

We also evaluated the different methods on a often-used collection of real-world problems [10]. These problems are in relatively low dimensions (between 2 and 11). We set the maximum number of function evaluations to 50,000 (which is usually done). The most striking insight from the computational experiments was that all these real-world problems could be consistently solved by DE. Additionally, for 8

of the 13 problems, more than half of the 11 methods achieved the exact same results. This made the setting of this real-world benchmarking set and performance metric (best result after a given number of function evaluations) not well suited for comparing advanced algorithms.

To study the performance of the seven problematic methods on difficult problems for which they were not tuned, we decided to test them on the ambiguous benchmark set [A6]. The results showed that only a single one of the problematic methods was significantly better than either PSO or DE. Three methods were comparable the to PSO, two performed very badly and the worst one performed barely better than a random search. We also found that even the best-performing algorithm of the problematic methods was dominated by the two algorithms from the CEC competitions.

Based on the described computational comparisons it was clear that the seven studied methods contain a centre-bias (or a zero-bias) operator that helped them perform extremely well on functions that have their respective optima in the centre of the feasible set. Some of these methods were analysed on and tuned to benchmark sets where such functions constitute the majority of the problems. When compared with other methods, they seemed to offer superior performance. However, when evaluated on shifted problems, most of their superior performance all but vanishes. And when evaluated on a proper benchmark set, some of them turn out to perform just barely better than a random search.

Further, we provide several suggestions that could help to improve analysis and benchmarking in evolutionary computation. Similar to other authors [125, 162], we believe that benchmark sets should also include instances of real-world problems. This seems particularly important since it was recently found that a large portion (near 30%) of nature-inspired algorithms studied previously [161] had no application associated to them. But, as we have shown, there are only a handful of frequently used real-world engineering problems, and these are too easy to serve as benchmarks. The EC community should initiate the construction of a large set of challenging real-world benchmark problems.

One possible reason for the ubiquity of the troublesome benchmark problems is their ease of use. They are relatively straightforward to program, and all of the methods considered in this paper included them in their respective source code. We advocated for the construction of an easy-to-use cross-platform repository, that would collect: (1) several heterogeneous benchmark sets (BBOB, CEC competitions, ambiguous benchmark set, real-world benchmarks, and similar ones) with a unified way of calling the test problems; (2) trusted implementations (source codes) of both standard EC methods and up-to-date state-of-the-art techniques; (3) data obtained from running the algorithms (from (2)) on the benchmarks (from (1)).

2.4.4 | A collection of robotics problems for benchmarking evolutionary computation methods

In the paper [A5], we presented a collection of real-world robotics problems that can be used for benchmarking evolutionary computation methods. The proposed benchmark problems were a combination of inverse kinematics and path planning in robotics that can be parameterized. As the framework for the benchmark problems, we chose the 6-DOF collaborative robotic arm, which is controlled by changing the angles θ_i in its joints. The [x,y,z]-coordinate position of the last link of the robot can be found as

$$[x, y, z]^T = FK(\theta), \tag{2.4.1}$$

where FK is the forward kinematics solution, and $\theta = [\theta_1, \ldots, \theta_6]^T$,$\theta_i \in [-2\pi, 2\pi]$, $i = 1, \ldots, 6$. In the proposed benchmark problems, we were interested in the trajectories of the robot's last link (end-effector), which corresponds to the way θ changes in time τ, and can be expressed as

$$[x(\tau), y(\tau), z(\tau)]^T = FK(\theta(\tau)). \tag{2.4.2}$$

The first quality of the trajectory we used was its length L, which (starting in $\tau = 0$ and ending in $\tau = 1$) can be expressed as

$$L = \int_0^1 \sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2}\mathrm{d}\tau. \tag{2.4.3}$$

The second quality of the trajectory was its closeness to a predefined point $[x_p, y_p, z_p]$, which can be written as

$$\min_{\tau \in [0,1]} || \, [x(\tau) - x_p, y(\tau) - y_p, z(\tau) - z_p] \, ||_2, \tag{2.4.4}$$

and, in the case of multiple predefined points $[x_p^j, y_p^j, z_p^j], j = 1, \ldots, P$, the closeness (C) to the farthest one

$$C = \max_{j=1,\ldots,P} \min_{\tau \in [0,1]} || \, [x(\tau) - x_p^j, y(\tau) - y_p^j, z(\tau) - z_p^j] \, ||_2. \tag{2.4.5}$$

As continuous control would pose too complex of a problem, we restricted our attention to a situation where the angles θ change linearly from one setting θ^a to the next θ^b, i.e.

$$\theta(\tau) = \theta^a + \tau(\theta^b - \theta^a). \tag{2.4.6}$$

In the case where we want to have multiple points of change $\theta^0, \ldots, \theta^M$ one of the possibilities is to model it as M time intervals of length 1, i.e.:

$$\hat{\theta}(\tau) = \theta^\iota + (\tau - \iota)(\theta^\iota - \theta^{\iota+1}), \quad \text{for } \tau \in [\iota, \iota + 1], \iota = 0, \ldots, M - 1. \tag{2.4.7}$$

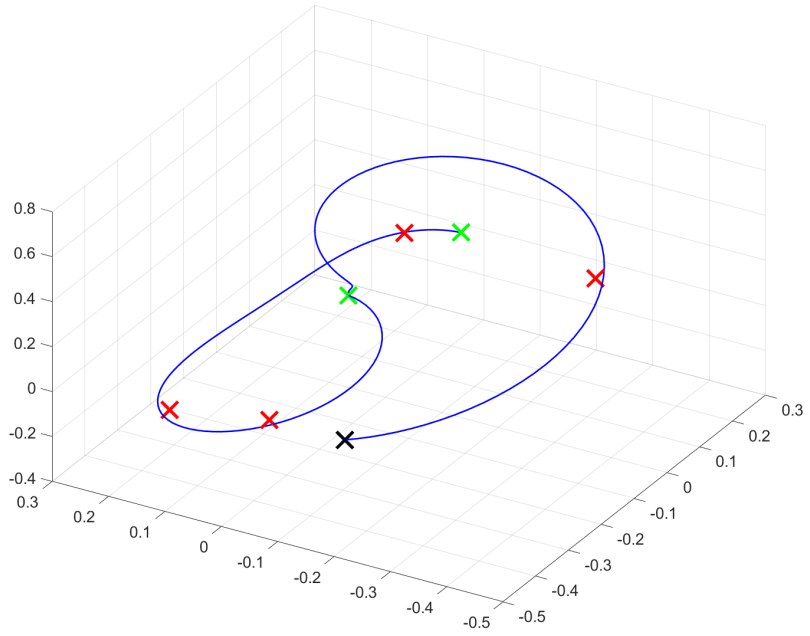


Fig. 2.8: Trajectory (blue line) of one solution starting of the black cross with $M = 2$ points of change (green crosses) and $P = 4$ predefined points (red crosses). Using $N = 100$, $\gamma = 100$ the objective value $f(\theta^1, \theta^2) = 4.8641$ [A5].

Even with the restriction on linear change in θ, the expressions (2.4.3) and (2.4.5) would be hard to compute analytically, which is why we resorted to a discretization of τ into $M \cdot N$ evenly spaced values $[\tau_1 = 0, \ldots, \tau_{M \cdot N} = M]$ and computed:

$$[x(\tau_i), y(\tau_i), z(\tau_i)] = FK(\hat{\theta}(\tau_i)), \quad i = 1, \ldots, M \cdot N \tag{2.4.8}$$

$$\hat{L} = \sum_{i=1}^{M \cdot N - 1} || \, [x(\tau_{i+1}) - x(\tau_i), y(\tau_{i+1}) - y(\tau_i), z(\tau_{i+1}) - z(\tau_i)] \, ||_2 \tag{2.4.9}$$

$$\hat{C} = \max_{j=1,\ldots,P} \min_{\tau_i, i=1,\ldots,M \cdot N} || \, [x(\tau_i) - x_p^j, y(\tau_i) - y_p^j, z(\tau_i) - z_p^j] \, ||_2. \tag{2.4.10}$$

For a given starting position θ^0, the objective function for all the considered benchmark problems has the form:

$$f(\theta^1, \ldots, \theta^M) = \gamma \cdot \hat{L} + \hat{C}, \tag{2.4.11}$$

where the parameter $\gamma \geq 0$ lets us control the degree to which we prefer trajectories with shorter length (higher γ), or higher precision in reaching the predefined points (lower γ). The resulting optimization problem is a "simple" box-constrained one:

$$\begin{aligned} &\text{minimize } f(\theta^1, \ldots, \theta^M) \\ &\text{subject to } \theta^\iota \in [-2\pi, 2\pi]^6, \iota = 1, \ldots, M \end{aligned}$$

The resulting benchmark function can be parametrized by:

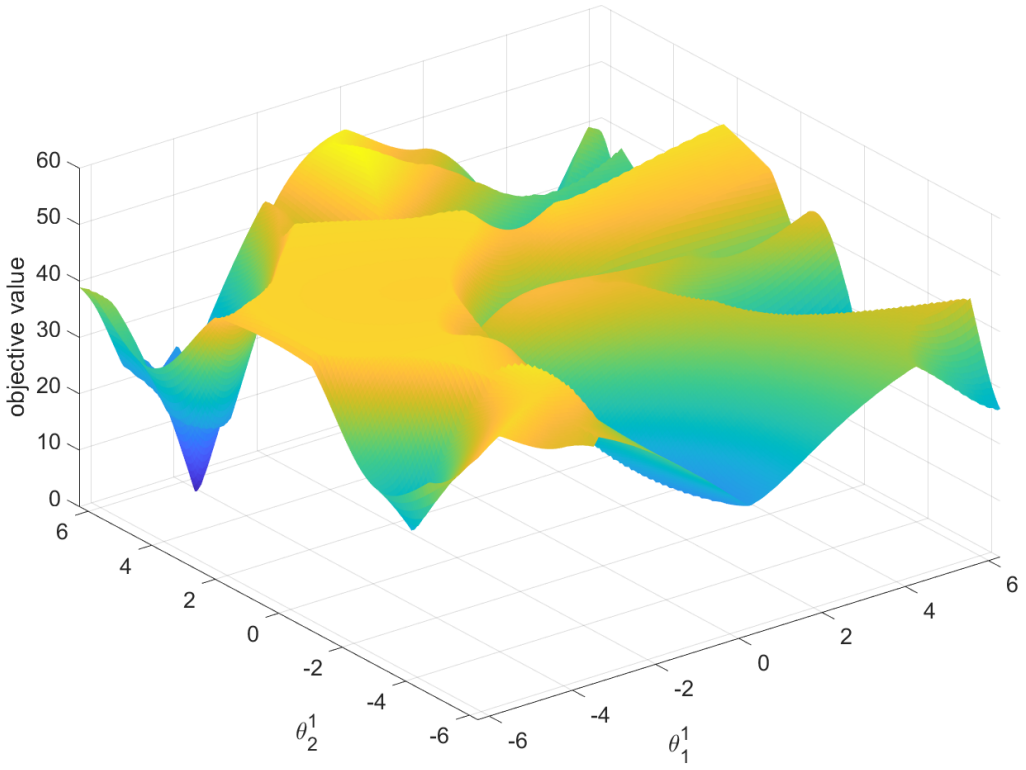


Fig. 2.9: Sensitivity of the objective value of the solution shown in Figure 2.8 on the first two components of θ^1 [A5].

(i) the number of points of change M, which determine the dimension of the optimization problem $D = 6 \cdot M$
(ii) the number of predefined points P to which the trajectory should get close to
(iii) the coefficient γ that scales the two objectives (trajectory length and closeness to the farthest predefined point)

Figure 2.8 shows a solution to one problem instance with $M = 2$, $P = 4$, and $\gamma = 100$ (with $N = 100$ as the discretization constant). Figure 2.9 shows the sensitivity of the objective function value on the first two components of θ_1. It can readily be seen that the objective is multimodal and nonseparable, which are both desirable characteristics in benchmark functions.

In order to showcase the capabilities of the proposed benchmark functions in differentiating various metaheuristics, we chose seven representative methods: Particle swarm optimization (PSO) [79], Differential evolution (DE) [44], Covariance matrix adaptation evolution strategy (CMA-ES) [65], Artificial bee colony (ABC) [75], Hybrid Sampling Evolution Strategy (HSES) [180], Adaptive Gaining-Sharing Knowledge (AGSK) [112], and Success-history based adaptive differential evolution with linear population size reduction (LSHADE) [160]. We conducted a thorough numerical investigation of the proposed benchmark problems - each of the seven

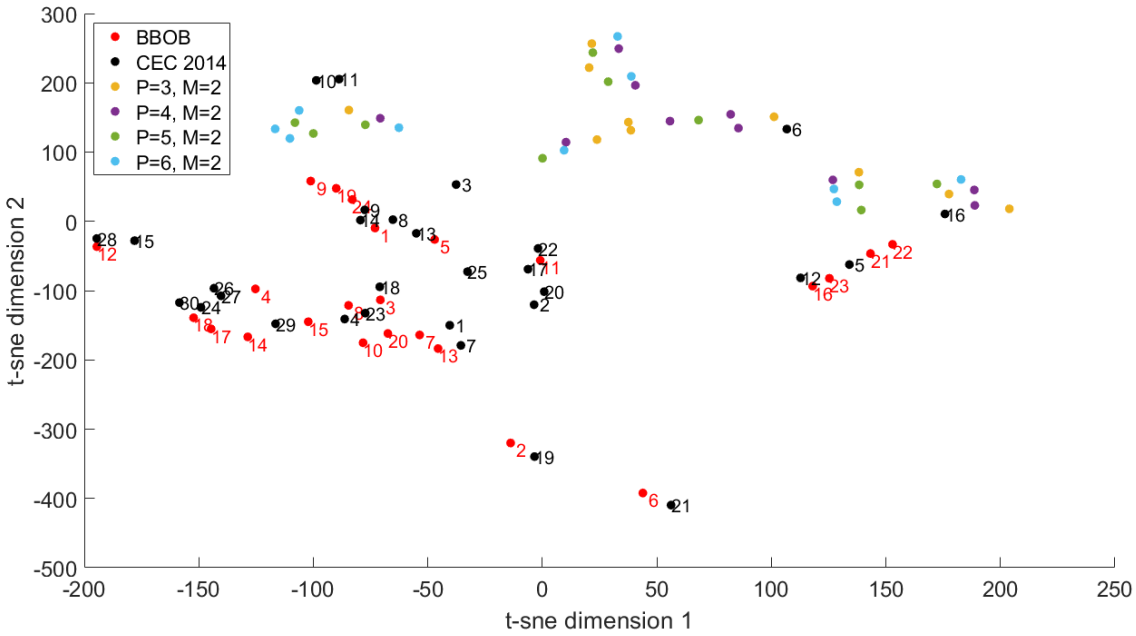


Fig. 2.10: The t-sne visualization of the ELA features (after normalization and using the first components from the PCA) of the three benchmark sets [A5].

selected methods for numerical comparison was used to solve 200 benchmark problems. The results of this investigation are that the proposed benchmark problems are quite difficult and that they can be successfully used for differentiating and ranking the selected metaheuristics.

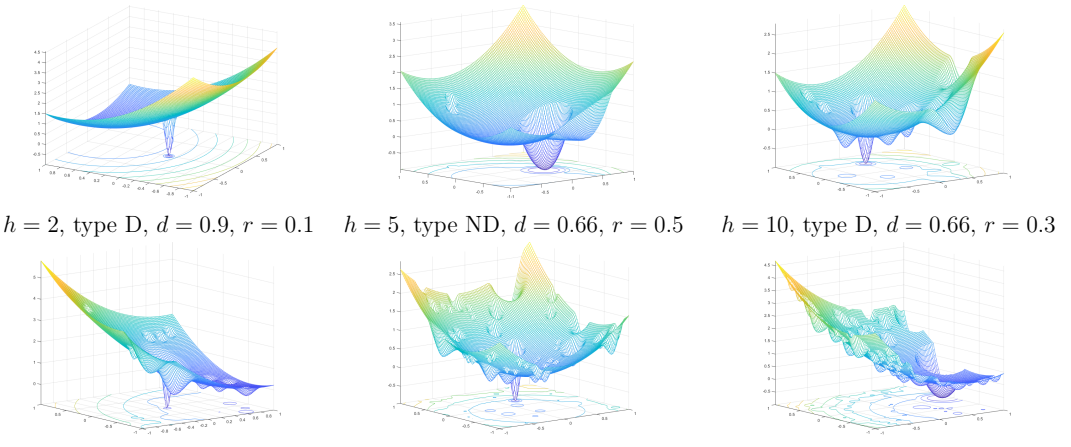
We also computed the ELA features of the proposed robotics problems to see how they compare to the BBOB and CEC 2014 benchmark suits. We followed the methodology described in [151] for the selection and visualization of the relevant ELA features. The features that produced constant results on every problem and those that produced invalid values were removed. Another set of removed features were the ones that were sensitive to scaling and shifting. The last batch of features that got removed were the highly correlated ones. For further analysis, the values of the ELA features on the three benchmark sets were normalized, and we used Principal Component Analysis (PCA) to reduce the number of features even further. Using the first 8 components explained 99.85% of the variance. For visualizing the results, we used the t-Distributed Stochastic Neighbor Embedding (t-sne). In the this visualization, which is shown in Figure 2.10, benchmark problems that have similar ELA features should be shown close to each other. We can see that the proposed robotics benchmarks are not very similar to functions in either BBOB or CEC 2014 sets. They are also not very similar to each other, at least in the sense of the performed analysis.

2.4.5 | Computational and Exploratory Landscape Analysis of the GKLS Generator

In the paper [A4] (an expanded version of which can be found in [96]) we analyzed the problems constructed by the GKLS generator. In the GKLS generator, a prespecified number of test problems (a class of problems) is constructed by defining a convex quadratic function (a paraboloid) which is systematically distorted by polynomials in order to produce local (and one global) minima. The input parameters for this construction are the following: type of the problem (ND: non-differentiable, D: differentiable, D2: twice-differentiable) problem dimension (D), number of local minima (h), the value of the global minimum (f^*), radius (r) of the attraction region of the global minimizer, and the distance (d) from the global minimizer to the vertex of the quadratic function. All problems are constructed on $[-1, 1]^D$.

A visualization of different functions that can be generated by the GKLS and the effect of different parameter choices is shown in Figure 2.11. Several interesting observations can be made regarding the generated functions. Firstly, they are all relatively well-conditioned. The local minimum of the "big" paraboloid is always in the domain and has a function value of 0. The "attraction regions" of the different local minima do not overlap (this is by design) - this also means that they become more shallow when their number increases.

To study the effect of the different parameters on the constructed problems, we set up the following computational analysis. We run three state-of-the-art methods both from the evolutionary computation (EC) and deterministic optimization communities on the "canonical" GKLS-generated problems in dimensions 5 and 10. We also construct a new class ("mod") 50 of GKLS-generated problems (again, in



$h = 2$, type D, $d = 0.9$, $r = 0.1$ $h = 5$, type ND, $d = 0.66$, $r = 0.5$ $h = 10$, type D, $d = 0.66$, $r = 0.3$

$h = 20$, type D, $d = 0.9$, $r = 0.1$ $h = 50$, type D, $d = 0.66$, $r = 0.2$ $h = 100$, type ND, $d = 0.66$, $r = 0.5$

Fig. 2.11: Functions generated by the GKLS in dimension $D = 2$ ($f^* = -1$ for all functions) [96].

$D = 5$, and 10) by the following procedure:

- Each problem $i = 1, \ldots, 50$ in this class will have different values of the parameters type_i, d_i, r_i, and h_i, but the same $f_i^* = -1$.
- type_i is decided by a coin flip between types D and ND (with the same probability).
- d_i is a uniformly distributed random number on [0,1].
- $r_i = d_i/u_i$, where u_i is a uniformly distributed random integer on [2,10], i.e. $r_i \in [d_i/2, d_i/10]$.
- $h_i = \text{round}(10^{c_i})$, where c_i is a uniformly distributed random number on [1,3], i.e. $h \in [10, 10^3]$.

From the EC side, we chose two methods to run on the GKLS-generated problems. The first selected method was Adaptive Gaining-Sharing Knowledge (AGSK), which was the runner-up of the CEC'20 competition [112]. The second method is L-SHADE or Success-history based adaptive differential evolution with linear population size reduction [160]. From the deterministic methods, we selected BIRMIN [123] as one of the best-performing methods from a recent extensive numerical study [156].

For the numerical comparison, we run each method once on every problem from each of the three classes (100 problems in both "simple" and "hard" classes, and 50 problems in the "mod" class) in dimensions $D = [5, 10]$, with a budget of $5{\cdot}10^4{\cdot}D$ available function evaluations. For every run, if the objective function value of the resulting solution was less than or equal to 1E–8, it was considered as zero.

For dimension $D = 5$, the Empirical cumulative distributions (ECDs) of simulated runtimes, measured in the number of function evaluations for 51 targets $10^{[-8..2]}$ (similar analysis which is done in the COCO platform) are shown in Figure 2.12. There is a noticeable difference in the behavior of the three algorithms on the "simple" and "hard" classes. On the "simple" class all three algorithms were able to find either a good or the optimal solutions faster than on the "hard" class. There is also a quite large difference between the performance of the three different methods - BIRMIN clearly dominated the two EC methods, and AGSK turned out to be better at finding good solutions at the later stages of the search than LSHADE.

The results change quite dramatically when looking at the "mod" class. Although the performance of BIRMIN is still superior to that of the two EC methods, the margin narrowed substantially. What is more, the relative performance (against the "hard" class) of AGSK decreased, while for LSHADE it increased. For all three classes, the local minimum of the "big" paraboloid (error value 1) was found by every method within a few hundred function evaluations for BIRMIN and a few thousand function evaluations for the EC methods.

The results for dimension $D = 10$ show another substantial change - we can see

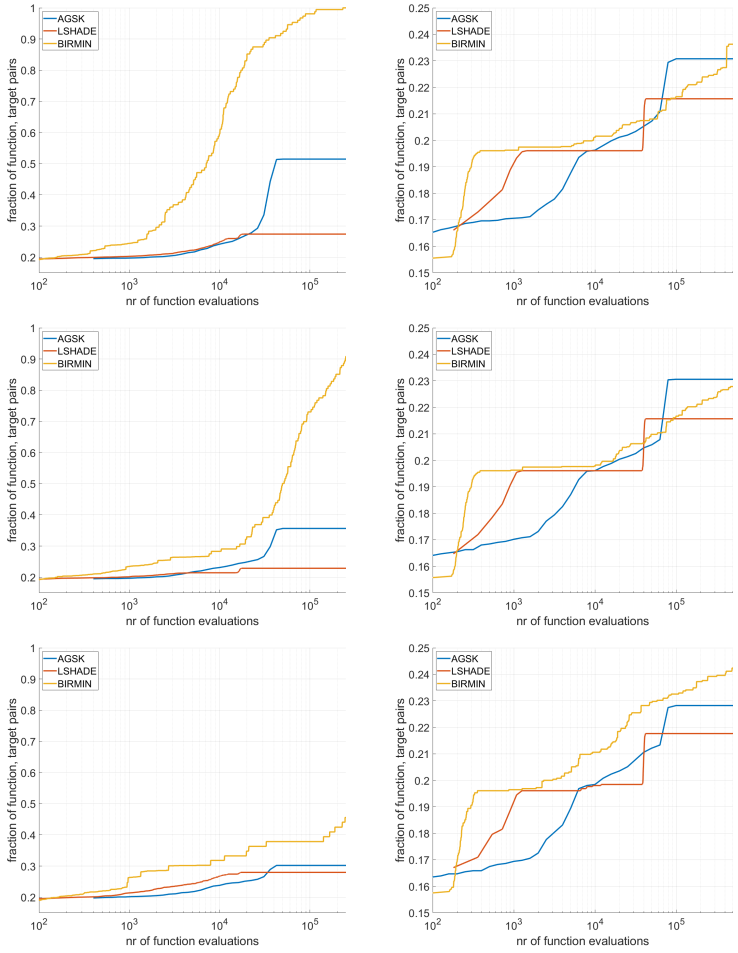


Fig. 2.12: ECD of simulated runtimes, measured in number of f-evaluations for the 51 targets $10^{[-8..2]}$ in dimension a) $D = 5$, b) $D = 10$ [96].

that all three classes are basically equivalent. The plateaus in the ECD plots between the 0.19 and 0.20 values on the y-axis indicate the points where the methods found the local minimum of the "big" paraboloid. Although this happened relatively early for all methods, finding better local optima proved to be extremely challenging.

Next, we use ELA features [81] to show how the GKLS-generated problems compare to the BBOB and CEC 2014 benchmark suits. We chose ELA feature sets which only require samples of input and function value pairs and dimension $D = 10$ for all considered suits. We chose to ignore the features that were sensitive to function transformations [153] and used uniform sampling with $250 \cdot D$ samples their computation [133].

We then followed the methodology described in [151] for the selection and visualization of the relevant ELA features. The features that produced constant results on every problem and those that produced invalid values were removed. Another batch of features that got removed were the highly correlated ones.

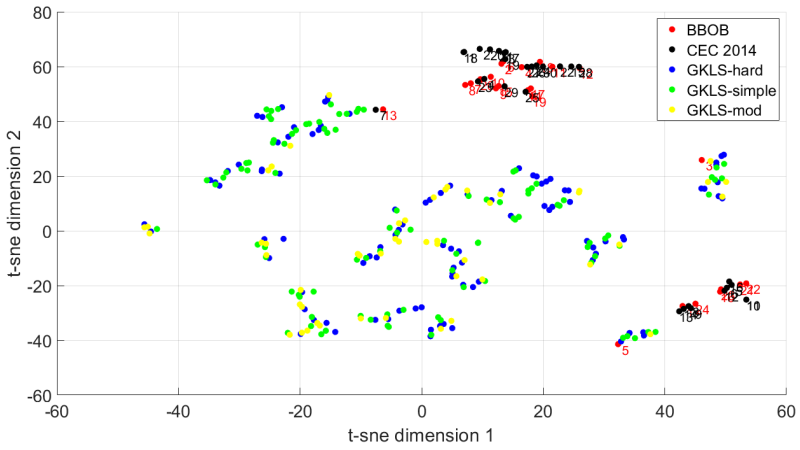


Fig. 2.13: The t-sne visualization of the ELA features (after normalization and using the first seven components from the PCA) of the benchmark sets [96].

The values of the ELA features on the different benchmark sets were then normalized, and we used Principal Component Analysis (PCA) to reduce the number of features even further. For visualizing the results, we used the t-Distributed Stochastic Neighbor Embedding (t-sne). In this visualization, which is shown in Figure 2.13, benchmark problems that have similar ELA features should be shown close to each other. We can see that the t-sne visualization grouped most of the functions from the BBOB and CEC 2014 suits together (in a few groups), while the "similar" problems generated in the three GKLS suites take up most of the space.

In the analysis, we have shown that the GKLS generator produces "needle in a haystack " type problems which get extremely difficult to optimize as the problem dimension grows. The GKLS generator could be successfully used for benchmarking state-of-the-art methods in lower dimensions ($D = 5$) on some of the simpler instances. However, in the higher dimension ($D = 10$), the performance of the three considered methods was hard to differentiate as the problems became extremely difficult (given the computational budget). This difficulty of the generated instances was also largely unaffected by the choice of parameters that the generator has.

It is possible that the GKLS generator could be modified to have a much "deeper" local minima. As the task of finding the global minimum is practically impossible in higher dimensions, having problems with lots of "good" local minima (i.e., better ones than the local minimum of the "big" paraboloid) could be useful for analyzing the exploration capabilities of optimization methods.

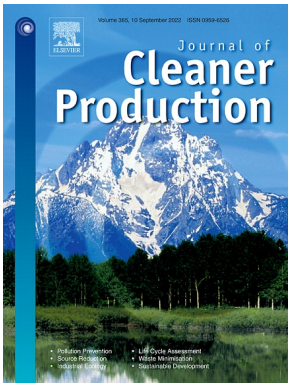
It is not very clear how one could meaningfully use the results of the computations of the ELA features or the t-sne plots on such "needle in a haystack" problems. It is probably impossible to have any sample-based features that would both uncover that the problem is a "needle in a haystack" and be computationally tractable (as it would amount to finding the "needle" in a reasonable amount of function evaluations).

2.5 | Applications

As was written in the introduction, we can find the utilization of optimization models in all aspects of our daily lives. Hence, applications of optimization can be found in all technical and engineering fields.

The author of this thesis has the most experience in the applications of optimization in waste management, mainly through collaborations with colleagues from the Institute of Process Engineering (Faculty of Mechanical Engineering, Brno University of Technology). A majority of this collaboration was on strategic network design problems that involved waste production and disposal. In [71], we developed a robust two-stage integer non-linear program for the support the strategic decision-making in this area applying modern circular economy principles. In [154], we presented a new model for integration of pricing and advertising strategies in conceptual waste management planning. We introduced a multistage stochastic model for legislation-induced planning of waste processing infrastructure in [91]. We also developed a new quadratic-optimization based model for waste flow quantification [166]. Another application-focused areas of interest of the author of this theses are mixed-integer models for university rankings [89], outlier identification methods [68], or dynamic programming approaches for the modelling of fuel mix changes in district heating networks [131].

The application papers selected for the inclusion in this thesis have one characteristic in common. They are not "just applications" of known models or techniques. They either propose new models for the problem at hand, or they include algorithmic techniques that take advantage of the problem-specific structure. The three IF journal papers that included in the thesis are:

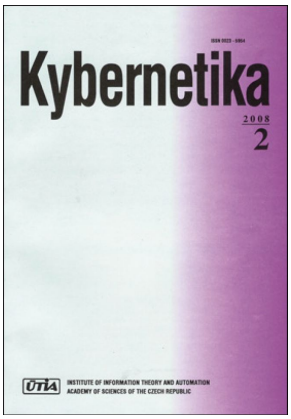


[A12] J Kůdela, R Šomplák, V Nevrlý, T Lipovský, V Smejkalová, L Dobrovský. Multi-objective strategic waste transfer station planning. Journal of Cleaner Production, 230:1294–1304, 2019.

Author's contribution: 55%.

Metrics: $\text{IF}_{2019} = 7.426$.

Ranking (JCR 2019 WoS): D1 - environmental sciences; Q1 - engineering, environmental; Q1 - green & sustainable science & technology.



[A10] J Kůdela, P Popela. Chance constrained optimal beam design: Convex reformulation and probabilistic robust design. Kybernetika, 54(6):1201–1217, 2018.

Author's contribution: 90%.

Metrics: $\text{IF}_{2018} = 0.560$.

Ranking (JCR 2018 WoS): Q4 - computer science, cybernetics.

[A1] J Kůdela. Social distancing as p-dispersion problem. IEEE Access, 8:149402–149411, 2020.

Author's contribution: 100%.

Metrics: $\text{IF}_{2020} = 3.367$.

Ranking (JCR 2020 WoS): Q2 - computer science, information systems; Q2 - engineering, electrical & electronic; Q2 - telecommunications.

2.5.1 | Multi-objective Strategic Waste Transfer Station Planning

The paper [A12] presents an approach utilizing an optimization model for the design of a transfer station grid for municipal solid waste collection. The presented approach was based on two-stage stochastic programming, where the uncertainties were projected through the model parameters. The result comprised of the suggestions of transfer stations placement, which were robust with respect to future realization of unknown parameters.

We used a multi-objective formulation of the model, which stemmed from the desire to design systems that are both economical and with as small environmental impact as possible. In this case, the environmental impact of a solution is measured in terms of the total distance travelled by all the vehicles used in the transportation of waste. The modelling technique employed to tackle the multi-objectivity was the standard scalarization one [24], where the two objectives are given different weights, which were used to construct a new single objective function that was minimized. By appropriately changing the weights, one obtains the desired trade-off curve (or a Pareto frontier) between the two objectives, as well as the corresponding optimal decisions. The uncertainty in data was modelled as different possible scenarios, with the objective computed as an average (trade-off between costs and distance) over these scenarios.

The derived optimization model was utilized in a case study that involved the transfer station planning in the Czech Republic for the mixed municipal waste. In terms of scale, it dealt with the most detailed description of the road networks and municipality structure available. In total, 6,258 nodes (municipalities producing waste), 44 waste processing plants (15 of which were foreign, allowing a potential export of the waste to Germany or Austria) and 116 possible places for the transfer stations were considered (these sets are not mutually exclusive). For every possible transfer station 6 options for its capacity were considered.

The first road network (connecting the municipalities) had 24,770 arcs and is depicted in Figure 2.14. In order to differentiate between the transportation of waste that does or does not use the transfer stations, a separate road network was computed - for each possible transfer station was found the shortest path to each waste-processing plant. In this pre-processing step, 5,075 shortest path optimization problems were solved, resulting in the second network.

The first-stage of the optimization problem consisted only of the planning decisions (on where to build the transfer stations) and is described by 696 binary variables. The second-stage of the optimization problem used 29,889 continuous decision variables.

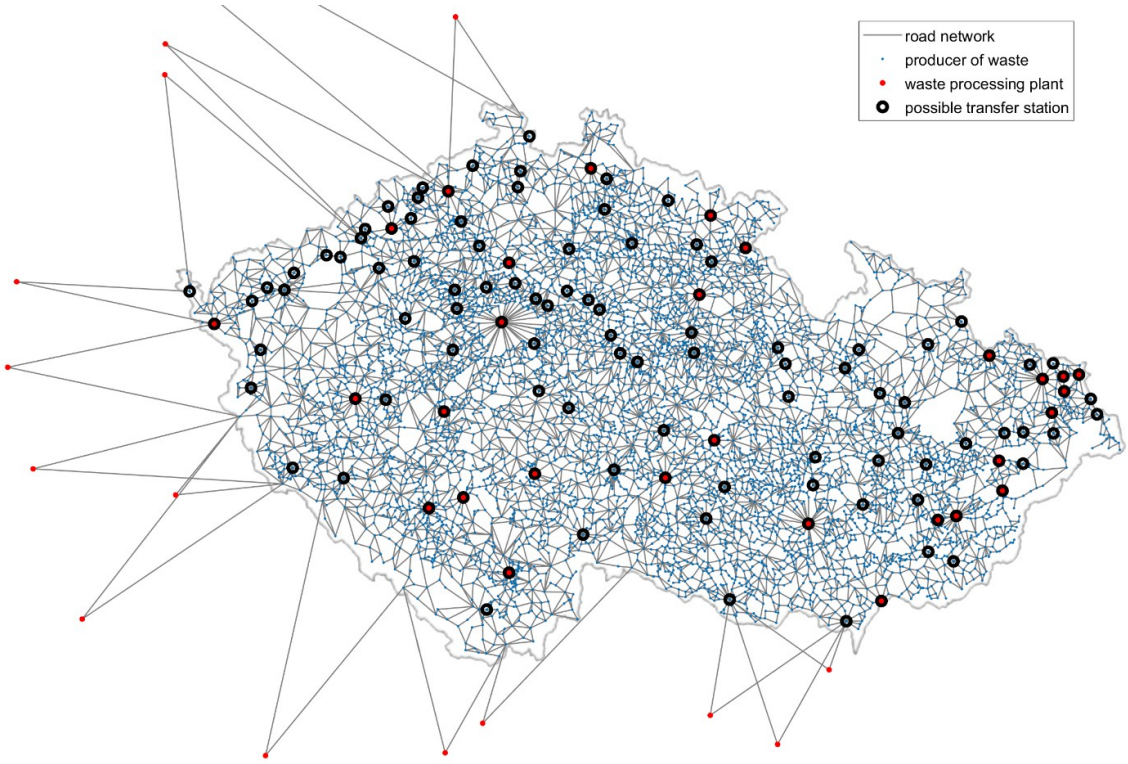


Fig. 2.14: A map showing the layout of the case study [A12].

The uncertain parameter that was considered in the model was the cost for processing the waste at the 44 different plants. To appropriately capture the nature of the inherent uncertainty, 1,000 possible scenarios for the waste treatment costs were constructed to be used within the optimization. The resulting optimization model had almost 30 million variables. The total number of constraints that depend on scenarios was 36,307, meaning that the optimization model had over 36 million constraints.

Because of the enormous number of variables of the considered optimization model, we utilized the GBD with warm starts that we developed in our previous work [A9]. The solution of the mixed-integer master problem was obtained using a branch-and-cut method (with lazy cuts), calling the CPLEX 12.6.3 solver [104], while the individual subproblems in the second stage were solved by the primal-dual simplex method, calling the GUROBI 7.5 solver [61]. This combination of solvers and algorithms achieved the best overall performance e the scheme reached the 1.5% optimality gap for the problem formulation with 1,000 scenarios within 24 h (for one setting of the scalarization weights).

The results showed that from the macro-level perspective, the mathematical model could be used for the assessment of the optimal strategies (both tactical and operational). It also provided means for the micro-level analysis of the impacts of the selected strategies on the individual municipalities and waste processing plants.

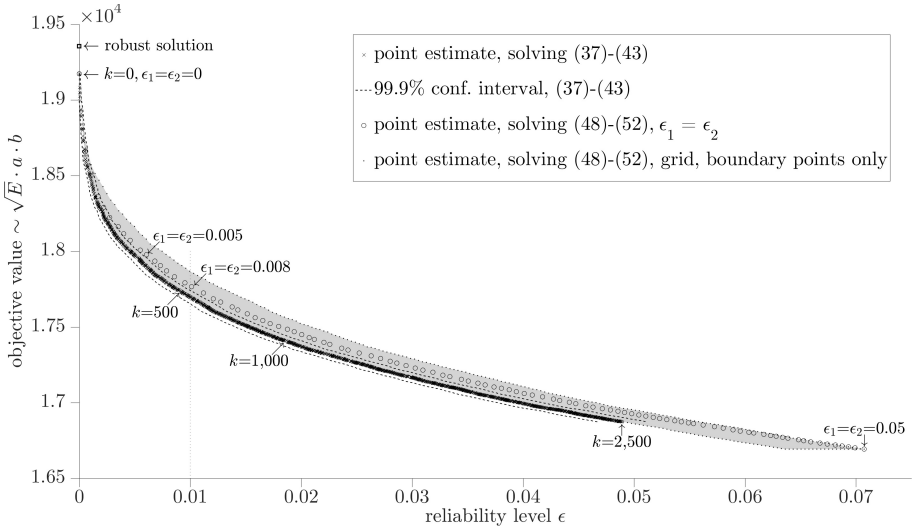


Fig. 2.15: The trade-off between reliability and optimal objective value [A10].

2.5.2 | Chance Constrained Optimal Beam Design: Convex Reformulation and Probabilistic Robust Design

In paper [A10], we are dealt with a civil engineering application of optimization, namely the optimal design of a loaded beam. We considered a fixed beam of a given length with a rectangular cross-section that was subjected to a given load. The task was to find the optimal design, in terms of the cross-section dimensions that minimizes the weight of the beam. The requirement for the design was that the maximum stress in the beam was less then a material-specific constant, that ensured that the design was safe.

We used the FEM approximation of the ordinary differential equations that describe the behaviour of the loaded beam [178]. We showed that this beam design problem formulation can be formulated as a geometric programming problem [24]. For this formulation, we even derived an analytic solution.

Further, the structure of the problem allowed us to consider on of the material constants (quality of the material) as a variable. We also made an additional restriction on the solution that involved the maximum absolute deflection of the beam. The resulting formulation was a geometric program in which every function both in the objective and the constraints was a monomial. This allowed us to the transform it into a linear program.

Next we investigated the robust and chance constrained variants of the problem, utilizing what can be though of as the prototype of the P&D algorithm that we presented in [A11]. In Figure 2.15 is depicted the resulting trade-off between the reliability level and the optimal objective value.

2.5.3 | Social Distancing as p-Dispersion Problem

In the paper [A1], we tackled the problem of positioning people in a given area, such as in a restaurant, school, office, etc., in the context of social distancing measures. We will posed the problem of using the available space to its full extent in the following way: Given a fixed number p of people, fit them into a predefined space in such a way, that the minimum distance between any two persons is maximized.

On way to model this problem is through the p-dispersion problem [114](pDP), where we are given a set of n points, a dissimilarity (or distance) matrix $D = \{D(i,j) : 1 \leq i,j \leq n\}$ satisfying $D(i,j) \geq 0$ for every $1 \leq i,j \leq n$ and $D(i,i) = 0$ for every $1 \leq i \leq n$, and an integer $p \geq 2$. The goal is to choose p points from the set of n points in such a way, that the minimum pairwise dissimilarity (the distance between any two points) within the selected points is maximized. The pDP is one of the classical combinatorial optimization problems and is known to be NP-hard [54].

There are different formulations and approaches for the pDP [126, 92]. In the paper, we utilized the pure binary compact formulation [142]. Let (I, E) be the complete graph in which points $I = \{1, \ldots, n\}$ are the vertices and $E = \{(i,j) \in I \times I : i < j\}$ are the edges. Given any distance d, we define subsets of edges as

$$E(d) = \{(i,j) \in E : D(i,j) < d\} \subseteq E.$$

The compact pure binary formulation exploits the fact that the optimal distance is identical to at least one of the entries in the dissimilarity matrix. Let $D^0 < D^1 < \cdots < D^{k_{max}}$ be the different non-zero values in D. The associated index sets are $K = \{1, 2, \ldots, k_{max}\}$ and $K_0 = \{0\} \cup K$. This formulation uses two types of binary variables: The binary location variable x_i indicates if the point $i \in I$ is selected. For $k \in K$, the binary variable z_k indicates if the location decisions (the particular selection of p points) satisfy a minimum distance of at least D^k. The pure binary program is the following:

$$\begin{aligned}
\max\ & D^0 + \sum_{k \in K} (D^k - D^{k-1}) z_k \\
\text{s.t.}\ & \sum_{i \in I} x_i = p \\
& z_k \leq z_{k-1},\ k \in K, k > 1 \\
& x_i + x_j + z_k \leq 2,\ k \in K, (i,j) \in E(D^k) \backslash E(D^{k-1}) \\
& x_i \in \{0,1\},\ i \in I \\
& z_k \in \{0,1\},\ k \in K
\end{aligned}$$

This formulation can be further strengthen using clique-like inequalities and computation can be sped-up by exploiting valid lower and upper bounds [142]. To solve this

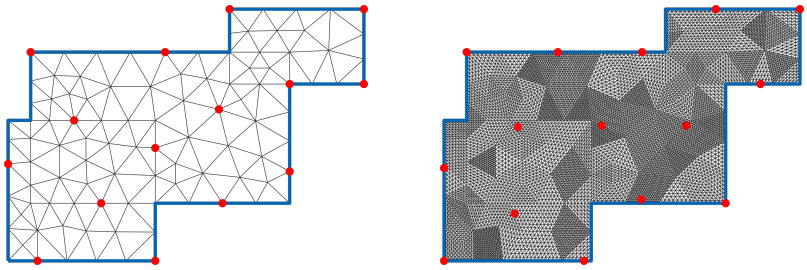


Fig. 2.16: Optimal placements, $p = 15$. Initial mesh with $n = 102$, $z^r = 552.63$ (left). Final mesh with $n = 8{,}745$, $z^r = 622.75$ (right) [A1].

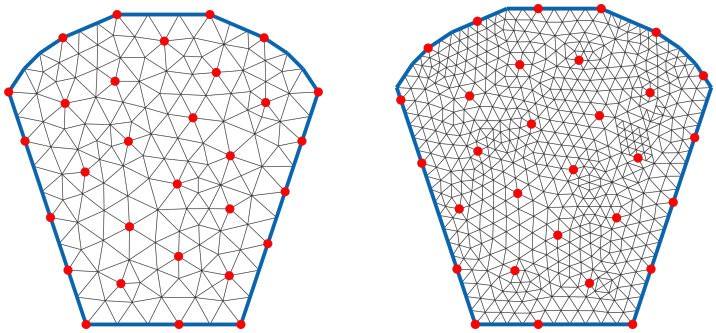


Fig. 2.17: Optimal placements, $p = 30$, $r = 1$. Initial mesh with $n = 150$, $z^r = 323.69$ (left). Final mesh with $n = 575$, $z^r = 352.96$ (right) [A1].

formulation, we used a decremental clustering scheme [40] that provides guarantees for optimality.

However, our goal was to devise a method for the continuous variant of the pDP is extremely difficult to solve and the techniques for approaching it are usually bound to convex feasible spaces [51]. In order to apply the pDP framework for general spaces, the feasible space of the problem was discretized. This discretization was carried out by triangulation of the two dimensional feasible area, with the vertices of the triangles being the possible feasible points [59].

A natural question arises about the relationship between the granularity of the triangulation and the objective value of the optimal solution of the associated pDP – the finer the mesh, the better the solution (with higher smallest distance between any two selected points). We addressed this issue by devising a mesh refinement scheme and solving a series of pDPs on a progressively finer mesh.

Another "discretization" can be devised in the space of possible values of the dissimilarity matrix D. As the problem gets more difficult with more unique values in D, with each unique value having a separate binary variable in the optimization model, we can greatly reduce the number of these added variables by rounding the elements of D. The trade-off between the optimal objective value and computational efforts of these two discretization schemes were investigated in computational experiments. Two representative results are shown in Figures 2.16 and 2.17.

Bibliography

[1] Lukáš Adam and Martin Branda. Nonlinear chance constrained problems: Optimality conditions, regularization and solvers. *Journal of Optimization Theory and Applications*, 170:419–436, 2016.

[2] Reza Alizadeh, Janet K Allen, and Farrokh Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31:275–298, 2020.

[3] Kurt M Anstreicher. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97:27–42, 2003.

[4] Kurt M Anstreicher and Nathan W Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical programming*, 89:341–357, 2001.

[5] Claus Aranha, Christian L Camacho Villalón, Felipe Campelo, Marco Dorigo, Rubén Ruiz, Marc Sevaux, Kenneth Sörensen, and Thomas Stützle. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence*, 16(1):1–6, 2022.

[6] Charles Audet and John E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization*, 17(1):188–217, 2006.

[7] Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *2005 IEEE congress on evolutionary computation*, volume 2, pages 1769–1776. IEEE, 2005.

[8] Anne Auger and Olivier Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.

[9] Mohsen Bagheri, Ali Gholinejad Devin, and Azra Izanloo. An application of stochastic programming method for nurse scheduling problem in real word hospital. *Computers & industrial engineering*, 96:192–200, 2016.

[10] Hadi Bayzidi, Siamak Talatahari, Meysam Saraee, and Charles-Philippe Lamarche. Social network search for solving engineering optimization problems. *Computational Intelligence and Neuroscience*, 2021:1–32, 2021.

[11] Evelyn ML Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):173–184, 1955.

[12] Vahid Beiranvand, Warren Hare, and Yves Lucet. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18:815–848, 2017.

[13] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.

[14] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[15] PR Berthelson, Payam Ghassemi, JW Wood, GG Stubblefield, AJ Al-Graitti, MD Jones, Mark F Horstemeyer, Souma Chowdhury, and RK Prabhu. A finite element–guided mathematical surrogate modeling approach for assessing occupant injury trends across variations in simplified vehicular impact conditions. *Medical & Biological Engineering & Computing*, 59:1065–1079, 2021.

[16] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[17] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009.

[18] Subhodip Biswas, Debanjan Saha, Shuvodeep De, Adam D Cobb, Swagatam Das, and Brian A Jalaian. Improving differential evolution through bayesian hyperparameter optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 832–840. IEEE, 2021.

[19] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308, 2003.

[20] J BnnoBRs. Partitioning procedures for solving mixed-variables programming problems ‘. *Numerische mathematik*, 4(1):238–252, 1962.

[21] Mohammad Reza Bonyadi and Zbigniew Michalewicz. Particle swarm optimization for single objective continuous space problems: a review. *Evolutionary computation*, 25(1):1–54, 2017.

[22] Fani Boukouvala and Christodoulos A Floudas. Argonaut: Algorithms for global optimization of constrained grey-box computational problems. *Optimization Letters*, 11:895–913, 2017.

[23] George EP Box and Norman R Draper. *Empirical model-building and response surfaces.* John Wiley & Sons, 1987.

[24] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[26] Petr Bujok, Josef Tvrdík, and Radka Poláková. Comparison of nature-inspired population-based algorithms on continuous optimisation problems. *Swarm and Evolutionary Computation*, 50:100490, 2019.

[27] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems: revised reprint*. SIAM, 2012.

[28] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. Qaplib–a quadratic assignment problem library. *Journal of Global optimization*, 10:391–403, 1997.

[29] Christian L Camacho-Villalón, Marco Dorigo, and Thomas Stützle. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. *International Transactions in Operational Research*, 2022.

[30] Christian Leonardo Camacho-Villalón, Marco Dorigo, and Thomas Stützle. The intelligent water drops algorithm: why it cannot be considered a novel algorithm: A brief discussion on the use of metaphors in optimization. *Swarm Intelligence*, 13:173–192, 2019.

[31] Christian Leonardo Camacho Villalón, Thomas Stützle, and Marco Dorigo. Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In *Swarm Intelligence: 12th International Conference, ANTS 2020, Barcelona, Spain, October 26–28, 2020, Proceedings 12*, pages 121–133. Springer, 2020.

[32] Felipe Campelo and Claus de Castro ARANHA. Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary. In *CEUR Workshop Proceedings*, volume 3007, page 6. CEUR Workshop Proceedings, 2021.

[33] Marco C Campi and Simone Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[34] Marco C Campi and Simone Garatti. A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *Journal of optimization theory and applications*, 148(2):257–280, 2011.

[35] Algo Care, Simone Garatti, and Marco C Campi. Scenario min-max optimization and the risk of empirical costs. *SIAM Journal on Optimization*, 25(4):2061–2080, 2015.

[36] Mauro Castelli, Luca Manzoni, Luca Mariot, Marco S Nobile, and Andrea Tangherloni. Salp swarm optimization: a critical review. *Expert Systems with Applications*, 189:116029, 2022.

[37] Eranda Cela. *The quadratic assignment problem: theory and algorithms*, volume 1. Springer Science & Business Media, 2013.

[38] Abraham Charnes, William W Cooper, and Gifford H Symonds. Cost horizons and certainty equivalents: an approach to stochastic programming of heating oil. *Management science*, 4(3):235–263, 1958.

[39] Tanmoy Chatterjee, Souvik Chakraborty, and Rajib Chowdhury. A critical review of surrogate assisted robust design optimization. *Archives of Computational Methods in Engineering*, 26:245–274, 2019.

[40] Claudio Contardo. Decremental clustering for the solution of p-dispersion problems to proven optimality. *INFORMS Journal on Optimization*, 2(2):134–144, 2020.

[41] Alberto Costa and Giacomo Nannicini. Rbfopt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10:597–629, 2018.

[42] Alison Cozad, Nikolaos V Sahinidis, and David C Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014.

[43] Wayne W Daniel et al. *Applied nonparametric statistics*. Duxbury Thompson Learning, 1990.

[44] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.

[45] Etienne de Klerk and Renata Sotirov. Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical programming*, 133:75–91, 2012.

[46] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[47] Javier Del Ser, Eneko Osaba, Aritz D Martinez, Miren Nekane Bilbao, Javier Poyatos, Daniel Molina, and Francisco Herrera. More is not always better: insights from a massive comparison of meta-heuristic algorithms over real-parameter optimization problems. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2021.

[48] Carola Doerr, Hao Wang, Furong Ye, Sander Van Rijn, and Thomas Bäck. Ioh-profiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281*, 2018.

[49] Elizabeth D Dolan, Robert Michael Lewis, and Virginia Torczon. On the local convergence of pattern search. *SIAM Journal on Optimization*, 14(2):567–583, 2003.

[50] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[51] Zvi Drezner and Erhan Erkut. Solving the continuous p-dispersion problem using non-linear programming. *Journal of the Operational Research Society*, pages 516–520, 1995.

[52] Stefan Droste, Thomas Jansen, and Ingo Wegener. Optimization with randomized search heuristics—the (a) nfl theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science*, 287(1):131–144, 2002.

[53] Tome Eftimov, Gašper Petelin, and Peter Korošec. Dsctool: A web-service-based framework for statistical comparison of stochastic optimization algorithms. *Applied Soft Computing*, 87:105977, 2020.

[54] Erhan Erkut. The discrete p-dispersion problem. *European Journal of Operational Research*, 46(1):48–60, 1990.

[55] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[56] Robert W Garden and Andries P Engelbrecht. Analysis and classification of optimisation benchmark functions and benchmark suites. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1641–1649. IEEE, 2014.

[57] Marco Gaviano, Dmitri E Kvasov, Daniela Lera, and Yaroslav D Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software (TOMS)*, 29(4):469–480, 2003.

[58] Arthur M Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10:237–260, 1972.

[59] PL George. Automatic mesh generation and finite element computation. *Handbook of numerical analysis*, 4:69–190, 1996.

[60] Paul C Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the society for industrial and applied mathematics*, 10(2):305–313, 1962.

[61] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[62] Anas A Hadi, Ali W Mohamed, and Kamal M Jambi. Single-objective real-parameter optimization: Enhanced lshade-spacma algorithm. *Heuristics for optimization and learning*, pages 103–121, 2021.

[63] Scott W Hadley, Franz Rendl, and Henry Wolkowicz. A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17(3):727–739, 1992.

[64] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

[65] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[66] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[67] Mardé Helbig and Andries P Engelbrecht. Performance measures for dynamic multi-objective optimisation algorithms. *Information Sciences*, 250:61–81, 2013.

[68] J Holesovsky and J Kudela. Outlier identification based on local extreme quantile estimation. In *Mendel*, volume 22, pages 255–260, 2016.

[69] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[70] Robert Hooke and Terry A Jeeves. "direct search"solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.

[71] Dušan Hrabec, Jakub Kůdela, Radovan Šomplák, Vlastimír Nevrlý, and Pavel Popela. Circular economy implementation in waste management network design problem: a case study. *Central European Journal of Operations Research*, 28(4):1441–1458, 2020.

[72] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[73] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 23(3):442–458, 2018.

[74] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79:157–181, 1993.

[75] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, 2005.

[76] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39:459–471, 2007.

[77] Pavel Karban, David Pánek, Tamás Orosz, Iveta Petrášová, and Ivo Doležel. Fem based robust design optimization with agros and ārtap. *Computers & Mathematics with Applications*, 81:618–633, 2021.

[78] Stefan E Karisch, Eranda Cela, Jens Clausen, and Torben Espersen. A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 63(4):351–403, 1999.

[79] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[80] Pascal Kerschke and Heike Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation*, 27(1):99–127, 2019.

[81] Pascal Kerschke and Heike Trautmann. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. *Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement*, pages 93–123, 2019.

[82] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[83] Joshua Knowles. Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[84] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E Bixby, Emilie Danna, Gerald Gamrath, Ambros M Gleixner, Stefan Heinz, et al. Miplib 2010: mixed integer programming library version 5. *Mathematical Programming Computation*, 3:103–163, 2011.

[85] Mykel J Kochenderfer and Tim A Wheeler. *Algorithms for optimization*. Mit Press, 2019.

[86] Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[87] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4:87–112, 1994.

[88] Jakub Kůdela. Minimum-volume covering ellipsoids: Improving the efficiency of the wolfe-atwood algorithm for large-scale instances by pooling and batching. *MENDEL*, 25(2):19–26, 2019.

[89] Jakub Kůdela. Mixed-integer programming model for ranking universities: Letting universities choose the weights. *MENDEL*, 27(1):41–48, 2021.

[90] Jakub Kůdela. Commentary on:"stoa: A bio-inspired based optimization algorithm for industrial engineering problems"[eaai, 82 (2019), 148–174] and "tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization"[eaai, 90 (2020), no. 103541]. *Engineering Applications of Artificial Intelligence*, 113:104930, 2022.

[91] Jakub Kůdela, Veronika Smejkalová, Radovan Šomplák, and Vlastimír Nevrlý. Legislation-induced planning of waste processing infrastructure: A case study of the czech republic. *Renewable and Sustainable Energy Reviews*, 132:110058, 2020.

[92] Michael J Kuby. Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. *Geographical Analysis*, 19(4):315–329, 1987.

[93] J Kudela. *Advanced decomposition methods in stochastic convex optimization*. PhD thesis, Doctoral thesis, Brno University of Technology, Brno, Czech Republic, 2019.

[94] Jakub Kudela. The evolutionary computation methods no one should use. *arXiv preprint arXiv:2301.01984*, 2023.

[95] Jakub Kudela, Tomas Holoubek, and Tomas Nevoral. Surrogate-assisted differential evolution-based method for the icsi'2022 competition. In *Advances in Swarm Intelligence: 13th International Conference, ICSI 2022, Xi'an, China, July 15–19, 2022, Proceedings, Part II*, pages 440–449. Springer, 2022.

[96] Jakub Kudela and Martin Juricek. Computational and exploratory landscape analysis of the gkls generator. *arXiv preprint arXiv:2304.08913*, 2023.

[97] Jakub Kudela, Tomas Nevoral, and Tomas Holoubek. Composite evolutionary strategy and differential evolution method for the icsi'2022 competition. In *Advances in Swarm Intelligence: 13th International Conference, ICSI 2022, Xi'an, China, July 15–19, 2022, Proceedings, Part II*, pages 432–439. Springer, 2022.

[98] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.

[99] Antonio LaTorre, Daniel Molina, Eneko Osaba, Javier Poyatos, Javier Del Ser, and Francisco Herrera. A prescription of methodological guidelines for comparing bio-inspired optimization algorithms. *Swarm and Evolutionary Computation*, 67:100973, 2021.

[100] Eugene L Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.

[101] Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V Kononova. Bbob instance analysis: Landscape properties and algorithm performance across problem instances. *arXiv preprint arXiv:2211.16318*, 2022.

[102] Ilya Loshchilov, Marc Schoenauer, and Michele Sebag. Dominance-based pareto-surrogate for multi-objective optimization. In *Simulated Evolution and Learning: 8th International Conference, SEAL 2010, Kanpur, India, December 1-4, 2010. Proceedings 8*, pages 230–239. Springer, 2010.

[103] Cédric Malherbe and Nicolas Vayatis. Global optimization of lipschitz functions. In *International Conference on Machine Learning*, pages 2314–2323. PMLR, 2017.

[104] CPLEX User's Manual. Ibm ilog cplex optimization studio. *Version*, 12(1987-2018):1, 1987.

[105] Joaquim RRA Martins and Andrew Ning. *Engineering design optimization*. Cambridge University Press, 2021.

[106] Meryem Masmoudi and Fouad Ben Abdelaziz. Portfolio selection problem: A review of deterministic and stochastic multiple objective programming models. *Annals of Operations Research*, 267:335–352, 2018.

[107] Radomil Matousek, Tomas Hulka, Ladislav Dobrovsky, and Jakub Kudela. Sum epsilon-tube error fitness function design for gp symbolic regression: preliminary study. In *2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, pages 78–83. IEEE, 2019.

[108] Radomil Matousek, Pavel Popela, and Jakub Kudela. Heuristic approaches to stochastic quadratic assignment problem: Var and cvar cases. In *Mendel*, volume 23, pages 73–78, 2017.

[109] Radomil Matousek and Eva Zampachova. Promising gahc and hc12 algorithms in global optimization tasks. *Optimization methods and software*, 26(3):405–419, 2011.

[110] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*, pages 73–82. Springer, 2010.

[111] Ali Wagdy Mohamed, Anas A Hadi, and Ali Khater Mohamed. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. *International Journal of Machine Learning and Cybernetics*, 11(7):1501–1529, 2020.

[112] Ali Wagdy Mohamed, Anas A Hadi, Ali Khater Mohamed, and Noor H Awad. Evaluating the performance of adaptive gaining-sharing knowledge based algorithm on cec 2020 benchmark problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[113] Daniel Molina, Javier Poyatos, Javier Del Ser, Salvador García, Amir Hussain, and Francisco Herrera. Comprehensive taxonomies of nature-and bio-inspired optimization: Inspiration versus algorithmic behavior, critical analysis recommendations. *Cognitive Computation*, 12:897–939, 2020.

[114] I Douglas Moon and Sohail S Chaudhry. An analysis of network location problems with distance constraints. *Management Science*, 30(3):290–307, 1984.

[115] Arkadi Nemirovski. On safe tractable approximations of chance constraints. *European Journal of Operational Research*, 219(3):707–718, 2012.

[116] Peifeng Niu, Songpeng Niu, Lingfang Chang, et al. The defect of the grey wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, 171:37–43, 2019.

[117] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[118] Lukas Novak and Drahomir Novak. Polynomial chaos expansion for surrogate modelling: Theory and software. *Beton-und Stahlbetonbau*, 113:27–32, 2018.

[119] Nilay Noyan. Risk-averse two-stage stochastic programming with an application to disaster management. *Computers & Operations Research*, 39(3):541–559, 2012.

[120] Karol R Opara, Anas A Hadi, and Ali W Mohamed. Parametrized benchmarking: an outline of the idea and a feasibility study. In *Proceedings of the 2020 genetic and evolutionary computation conference companion*, pages 197–198, 2020.

[121] Bernardo K Pagnoncelli, Shabbir Ahmed, and Alexander Shapiro. Sample average approximation method for chance constrained programming: theory and applications. *Journal of optimization theory and applications*, 142(2):399–416, 2009.

[122] Bernardo K Pagnoncelli, Daniel Reich, and Marco C Campi. Risk-return trade-off with the scenario approach in practice: a case study in portfolio selection. *Journal of Optimization Theory and Applications*, 155:707–722, 2012.

[123] Remigijus Paulavičius, Yaroslav D Sergeyev, Dmitri E Kvasov, and Julius Žilinskas. Globally-biased birect algorithm with local accelerators for expensive global optimization. *Expert Systems with Applications*, 144:113052, 2020.

[124] Jiming Peng, Tao Zhu, Hezhi Luo, and Kim-Chuan Toh. Semi-definite programming relaxation of quadratic assignment problems based on nonredundant matrix splitting. *Computational Optimization and Applications*, 60:171–198, 2015.

[125] Adam P Piotrowski. Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. *Information Sciences*, 297:191–201, 2015.

[126] David Pisinger. Upper bounds and exact algorithms for p-dispersion problems. *Computers & operations research*, 33(5):1380–1398, 2006.

[127] SA Piyavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972.

[128] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964.

[129] Warren B Powell and Huseyin Topaloglu. Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635, 2003.

[130] András Prékopa. *Stochastic programming*, volume 324. Springer Science & Business Media, 2013.

[131] Ondřej Putna, Jakub Kůdela, Martin Krňávek, Martin Pavlas, and Kamil Ondra. Modelling of change in fuel mix within a district heating network. *Energies*, 15(8):2879, 2022.

[132] Rommel G Regis. Particle swarm with radial basis function surrogates for expensive black-box optimization. *Journal of Computational Science*, 5(1):12–23, 2014.

[133] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II 16*, pages 139–153. Springer, 2020.

[134] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[135] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[136] Mark Rubinstein. Markowitz's" portfolio selection": A fifty-year retrospective. *The Journal of finance*, 57(3):1041–1045, 2002.

[137] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

[138] Conor Ryan, John James Collins, and Michael O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming: First European Workshop, EuroGP'98 Paris, France, April 14–15, 1998 Proceedings 1*, pages 83–96. Springer, 1998.

[139] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[140] Karam M Sallam, Saber M Elsayed, Ripon K Chakrabortty, and Michael J Ryan. Improved multi-operator differential evolution algorithm for solving unconstrained problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[141] Claudio Sanhueza, Francia Jiménez, Regina Berretta, and Pablo Moscato. Pasmoqap: a parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem. In *2017 IEEE congress on evolutionary computation (CEC)*, pages 1103–1110. IEEE, 2017.

[142] David Sayah and Stefan Irnich. A new compact formulation for the discrete p-dispersion problem. *European Journal of Operational Research*, 256(1):62–67, 2017.

[143] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[144] Ya D Sergeyev, DE Kvasov, and MS Mukhametzhanov. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Scientific reports*, 8(1):1–9, 2018.

[145] Yaroslav D Sergeyev and Dmitri E Kvasov. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization*, 16(3):910–937, 2006.

[146] Yaroslav D Sergeyev, Dmitri E Kvasov, and Marat S Mukhametzhanov. A generator of multiextremal test classes with known solutions for black-box-constrained global optimization. *IEEE Transactions on Evolutionary Computation*, 26(6):1261–1270, 2021.

[147] Feng Shan, Liwei Zhang, and Xiantao Xiao. A smoothing function approach to joint chance-constrained programs. *Journal of Optimization Theory and Applications*, 163:181–199, 2014.

[148] Alexander Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, 2011.

[149] Alexander Shapiro and Arkadi Nemirovski. On complexity of stochastic programming problems. *Continuous optimization: Current trends and modern applications*, pages 111–146, 2005.

[150] Bruno O Shubert. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 1972.

[151] Urban Škvorc, Tome Eftimov, and Peter Korošec. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing*, 90:106138, 2020.

[152] Urban Škvorc, Tome Eftimov, and Peter Korošec. The effect of sampling methods on the invariance to function transformations when using exploratory landscape analysis. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1139–1146. IEEE, 2021.

[153] Urban Škvorc, Tome Eftimov, and Peter Korošec. A comprehensive analysis of the invariance of exploratory landscape analysis features to function transformations. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2022.

[154] Radovan Šomplák, Jakub Kůdela, Veronika Smejkalová, Vlastimír Nevrlý, Martin Pavlas, and Dušan Hrabec. Pricing and advertising strategies in conceptual waste management planning. *Journal of Cleaner Production*, 239:118068, 2019.

[155] Jörg Stork, Martina Friese, Martin Zaefferer, Thomas Bartz-Beielstein, Andreas Fischbach, Beate Breiderhoff, Boris Naujoks, and Tea Tušar. Open issues in surrogate-assisted optimization. *High-performance simulation-based optimization*, pages 225–244, 2020.

[156] Linas Stripinis and Remigijus Paulavičius. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms. *arXiv preprint arXiv:2209.05759*, 2022.

[157] Gang Sun and Shuyue Wang. A review of the artificial neural network surrogate modeling in aerodynamic design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16):5863–5872, 2019.

[158] Samer Takriti and Shabbir Ahmed. On robust optimization of two-stage systems. *Mathematical Programming*, 99(1):109–126, 2004.

[159] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[160] Ryoji Tanabe and Alex S Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*, pages 1658–1665. IEEE, 2014.

[161] Alexandros Tzanetos and Georgios Dounias. A comprehensive survey on the applications of swarm intelligence and bio-inspired evolutionary strategies. *Machine learning paradigms: advances in deep learning-based technological applications*, pages 337–378, 2020.

[162] Alexandros Tzanetos and Georgios Dounias. Nature inspired optimization algorithms or simply variations of metaheuristics? *Artificial Intelligence Review*, 54:1841–1862, 2021.

[163] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[164] Niki Veček, Matej Črepinšek, Marjan Mernik, and Dejan Hrnčič. A comparison between different chess rating systems for ranking evolutionary algorithms. In *2014 Federated Conference on Computer Science and Information Systems*, pages 511–518. IEEE, 2014.

[165] CC Villalón, T Stützle, and M Dorigo. Cuckoo search≡(μ+ λ)–evolution strategy. In *IRIDIA–Technical Report Series*. 2021.

[166] Radovan Šomplák, Veronika Smejkalová, and Jakub Kůdela. Mixed-integer quadratic optimization for waste flow quantification. *Optimization and Engineering*, 23:2177–2201, 2022.

[167] Stein W Wallace and Stein-Erik Fleten. Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637–677, 2003.

[168] Stein W Wallace and William T Ziemba. *Applications of stochastic programming*. SIAM, 2005.

[169] Handing Wang. Uncertainty in surrogate models. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1279–1279, 2016.

[170] Thomas Weise and Zijun Wu. Difficult features of combinatorial optimization problems and the tunable w-model benchmark problem for simulating them. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1769–1776, 2018.

[171] Paul Westermann and Ralph Evins. Surrogate modelling for sustainable building design–a review. *Energy and Buildings*, 198:170–186, 2019.

[172] Dennis Weyland. A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing (IJAMC)*, 1(2):50–60, 2010.

[173] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[174] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[175] JiaHong Xu and LiHong Xu. N-gkls: An nmpc problem generator and a test platform for nmpc solvers. *Expert Systems with Applications*, 214:119029, 2023.

[176] Cheng Yan, Zeyong Yin, Xiuli Shen, Dong Mi, Fushui Guo, and Dan Long. Surrogate-based optimization with improved support vector regression for non-circular vent hole on aero-engine turbine disk. *Aerospace Science and Technology*, 96:105332, 2020.

[177] Song Yang, Wei Tian, Eduard Cubi, QingXin Meng, YunLiang Liu, and Lai Wei. Comparison of sensitivity analysis methods in building energy assessment. *Procedia Engineering*, 146:174–181, 2016.

[178] Eva Žampachová, Pavel Popela, and Michal Mrázek. Optimum beam design via stochastic programming. *Kybernetika*, 46(3):571–582, 2010.

[179] Filippo Zanetti and Jacek Gondzio. An interior-point-inspired algorithm for linear programs arising in discrete optimal transport. *arXiv, paper 2206.11009*, 2022.

[180] Geng Zhang and Yuhui Shi. Hybrid sampling evolution strategy for solving single objective bound constrained problems. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7. IEEE, 2018.

[181] Qing Zhao, Stefan E. Karisch, Franz Rendl, Henry Wolkowicz, et al. Semidefinite programming relaxations for the quadratic assignment problem. *J. Comb. Optim.*, 2(1):71–109, 1998.

[182] Victor Zverovich, Csaba I Fábián, Eldon FD Ellison, and Gautam Mitra. A computational study of a solver system for processing two-stage stochastic lps with enhanced benders decomposition. *Mathematical Programming Computation*, 4:211–238, 2012.

Bibliography - Selected Papers

[A1] Jakub Kůdela. Social distancing as p-dispersion problem. *IEEE Access*, 8:149402–149411, 2020.

[A2] Jakub Kůdela. Novel zigzag-based benchmark functions for bound constrained single objective optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 857–862. IEEE, 2021.

[A3] Jakub Kůdela. A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence*, 4:1238–1245, 2022.

[A4] Jakub Kůdela and Martin Juříček. Computational and exploratory landscape analysis of the gkls generator. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, page tbd, 2023.

[A5] Jakub Kůdela, Martin Juříček, and Roman Parák. A collection of robotics problems for benchmarking evolutionary computation methods. In *Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*, pages 364–379. Springer, 2023.

[A6] Jakub Kůdela and Radomil Matoušek. New benchmark functions for single-objective optimization based on a zigzag pattern. *IEEE Access*, 10:8262–8278, 2022.

[A7] Jakub Kůdela and Radomil Matoušek. Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Computing*, 26(24):13709–13733, 2022.

[A8] Jakub Kůdela and Radomil Matoušek. Combining lipschitz and rbf surrogate models for high-dimensional computationally expensive problems. *Information Sciences*, 619:457–477, 2023.

[A9] Jakub Kůdela and Pavel Popela. Warm-start cuts for generalized benders decomposition. *Kybernetika*, 53(6):1012–1025, 2017.

[A10] Jakub Kůdela and Pavel Popela. Chance constrained optimal beam design: Convex reformulation and probabilistic robust design. *Kybernetika*, 54(6):1201–1217, 2018.

[A11] Jakub Kůdela and Pavel Popela. Pool & discard algorithm for chance constrained optimization problems. *IEEE Access*, 8:79397–79407, 2020.

[A12] Jakub Kůdela, Radovan Šomplák, Vlastimír Nevrlý, Tomáš Lipovský, Veronika Smejkalová, and Ladislav Dobrovský. Multi-objective strategic waste transfer station planning. *Journal of Cleaner Production*, 230:1294–1304, 2019.

[A13] Radomil Matoušek, Ladislav Dobrovský, and Jakub Kůdela. The quadratic assignment problem: metaheuristic optimization using HC12 algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 153–154, 2019.

[A14] Radomil Matoušek, Ladislav Dobrovský, and Jakub Kůdela. How to start a heuristic? utilizing lower bounds for solving the quadratic assignment problem. *International Journal of Industrial Engineering Computations*, 13(2):151–164, 2022.

Appendix - Selected Papers

A1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Social Distancing as p-Dispersion Problem

JAKUB KUDELA[1]
[1]Institute of Automation and Computer Science, Brno University of Technology, Czech Republic

Corresponding author: Jakub Kudela (e-mail: Jakub.Kudela@vutbr.cz).

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic project No. CZ.02.1.01/0.0/0.0/16_026/0008392 "Computer Simulations for Effective Low-Emission Energy" and by IGA BUT: FSI-S-20-6538.

ABSTRACT The spread of COVID-19 and similar viruses poses new challenges for our society. There is a strong incentive towards safety measures that help to mitigate the outbreaks. Many countries have imposed social distancing measures that require a minimum distance between people in given places, such as schools, restaurants, shops, etc. This in turn creates complications for these places, as their function is to serve as many people as they were originally designed for. In this paper, we pose the problem of using the available space in a given place, such that the social distancing measures are satisfied, as a p-dispersion problem. We use recent algorithmic advancements, that were developed for the p-dispersion problem, and combine them with discretization schemes to find computationally attainable solutions to the p-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution on the other.

INDEX TERMS social distancing, p-dispersion problem, decremental clustering, COVID-19

I. INTRODUCTION

THE outbreak of the COVID-19 had an enormous impact on the world at large. To mitigate the spread of the virus, various technologies, such as Internet of Things, Unmanned Aerial Vehicles, blockchain, Artificial Intelligence, and 5G are already in use [1]. In this paper, we take a look at the problem of positioning people in a given area, such as in a restaurant, school, office, etc., in order to minimize the spread of viruses such as COVID-19. After the initial lockdown, many countries imposed a set of social distancing measures that should help to slow down the spread of the virus. These measures impose a minimum distance between people in a given area. This means that spaces that could previously serve a large number people need to be adjusted for these new measures. As it seems unlikely that we will see the construction of new places that will be designed to abide by these (hopefully temporary) measures, it is only natural to try to find the best use of the "facilities" that are already available. However, as the social distancing measures do not have to be stable and can change over time, we will pose the problem of using the available space to its full extent in the following way: Given a fixed number p of people, fit them into a predefined space in such a way, that the minimum distance between any two persons is maximized. Afterwards, by varying p, we can get the optimal (largest) distance that the people can be separated by, and, given a particular social distancing rule, we can determine the maximum number (and placement) of people that will fit into the predefined space. The problem of selecting p points in order to maximize the minimum distance between any pair is called the p-dispersion problem [2] and it is one of the classical combinatorial optimization problems. Although easy to formulate, effective and provably optimal methods for solving this problem are quite a recent development. Most notably, the state-of-the-art methods are based on the formulation developed in 2017 in [3] and the most successful method is the decremental clustering scheme published in 2020 in [4]. It is these advancements that made it possible to solve instances of the size sufficient for our purpose. In this paper, we devise a discretization scheme that is build on top of the decremental clustering to find computationally attainable solutions to the p-dispersion problem and investigate the trade-off between the level of discretization and computational efforts on one side, and the value of the optimal solution (the minimum distance between any two points) on the other. The investigation is carried out on two numerical examples, the first one is a place with a "general" shape, the second one with an "auditorium-like" shape.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

II. THE P-DISPERSION PROBLEM

A. DEFINITION AND FORMULATIONS

In the p-dispersion problem (pDP), we are given a set of n points, a dissimilarity (or distance) matrix $D = \{D(i,j) : 1 \leq i,j \leq n\}$ satisfying $D(i,j) \geq 0$ for every $1 \leq i,j \leq n$ and $D(i,i) = 0$ for every $1 \leq i \leq n$, and an integer $p \geq 2$. The goal is to choose p points from the set of n points in such a way, that the minimum pairwise dissimilarity (the distance between any two points) within the selected points is maximized. The pDP is an NP hard problem [5]. We denote this problem for given input parameters D and p as $\texttt{pDP}(D,p)$. One of the standard applications of the pDP is the location of nuclear power plants, where one is interested in minimizing the risk of losing multiple plants in the event that only one plant is subjected to an enemy attack. To achieve this, the desired selection of plants is that in which the interplant distances are as large as possible [3]. A more peaceful applications of the pDP can be found in location analysis of services, e.g., schools, hospitals, electoral districts, or waste collection plants. A comprehensive survey of the location applications of pDP can be found in [6] and [7]. Another application of the pDP is found in multiobjective optimization – if the Pareto frontier of a problem contains multiple solutions, one can solve a pDP to find p such solutions with most distinct features [8]. In the same paper, an application in portfolio optimization is presented – given a set of potential investment opportunities, one wishes to choose a subset that reduces the closeness in terms of features between the different investment options, which reduces the risk associated with the portfolio.

Within the methodological contributions to the solution of the pDP, several articles have dealt with the problem of solving the pDP to proven optimality. A mixed-integer quadratic formulation was introduced by [9], which can be partially solved by a series of relaxations and reformulation-linearization. A mixed-integer linear formulation of the problem using the "big M" constraints was defined in [6]. This formulation can be retroactively thought of as a linearization of the mixed integer quadratic model, that was developed 20 years afterward. Although the linear model is more compact than the quadratic one, it provides much weaker upper bounds.

Our attention will focus on a formulation introduced in [3], which is a novel pure binary compact formulation. Using this formulation, the authors reported substantial computational advancements when compared with the other formulations. Without loss of generality, we can assume that the dissimilarity matrix D is symmetric. Let (I,E) be the complete graph in which points $I = \{1,\ldots,n\}$ are the vertices and $E = \{(i,j) \in I \times I : i < j\}$ are the edges. Given any distance d, we define subsets of edges as

$$E(d) = \{(i,j) \in E : D(i,j) < d\} \subseteq E.$$

The compact pure binary formulation exploits the fact that the optimal distance is identical to at least one of the entries in the dissimilarity matrix. Let $D^0 < D^1 < \cdots < D^{k_{max}}$ be the different non-zero values in D. The associated index sets are $K = \{1,2,\ldots,k_{max}\}$ and $K_0 = \{0\} \cup K$. This formulation uses two types of binary variables: The binary location variable x_i indicates if the point $i \in I$ is selected. For $k \in K$, the binary variable z_k indicates if the location decisions (the particular selection of p points) satisfy a minimum distance of at least D^k. The pure binary program is the following:

$$\max\ D^0 + \sum_{k \in K} (D^k - D^{k-1}) z_k \tag{1}$$

$$\text{s.t.} \sum_{i \in I} x_i = p \tag{2}$$

$$z_k \leq z_{k-1},\ k \in K, k > 1 \tag{3}$$

$$x_i + x_j + z_k \leq 2,\ k \in K, (i,j) \in E(D^k) \backslash E(D^{k-1}) \tag{4}$$

$$x_i \in \{0,1\},\ i \in I \tag{5}$$

$$z_k \in \{0,1\},\ k \in K \tag{6}$$

The formulation (1)-(6) can be further strengthen using clique-like inequalities and computation can be sped-up by exploiting valid lower and upper bounds [3].

B. DECREMENTAL CLUSTERING METHOD

The decremental clustering method introduced in [4] for the pDP utilizes the formulation (1)-(6). The usage of clustering techniques for finding feasible solutions for combinatorial problems is hardly new. For example, in vehicle routing and scheduling, the "cluster-first, route-second" (see [10], [11]) and "route-first, cluster-second" (see [12], [13]) paradigms were used to ease the computational burden of the hard combinatorial problem. What sets the decremental clustering method apart is that it provides guarantees for optimality. Decremental clustering was also proposed for the solution of the vertex p-center problem in [14] and [15].

We present the decremental clustering method with the same notation and vocabulary as it was developed in [4]. A clustering of the n nodes, denoted by $\mathcal{C}$ is a family $\{C_i : i = 1,\ldots,m\}$ such that $C_i \cap C_j = \emptyset$ for every $1 \leq i < j \leq m$ and $\cup\{C_i : 1,\ldots,m\} = I$. A clustering $\mathcal{C}$ is said to be sufficiently refined if, for every set $C_i \in \mathcal{C}, D(C_i) := \max\{D(u,v) : u,v \in C_i, u < v\} < z^*$, where z^* is the optimal value of $\texttt{pDP}(D,p)$. The correctness of the decremental clustering method is supported by the following result (proved in [4]).

Lemma 1: Let $\mathcal{C}$ be a sufficiently refined clustering of the nodes of size m. Let $D^{\mathcal{C}}$ be a $m \times m$ dissimilarity matrix where $D^{\mathcal{C}}(i,j) = \max\{D(u,v) : u \in C_i, v \in C_j\}$. The optimal value ζ^* of the problem $\texttt{pDP}(D^{\mathcal{C}},p)$ provides an upper bound of problem $\texttt{pDP}(D,p)$.

The decremental clustering method works as follows. A lower bound $L \leq z^*$ is computed using a k-means algorithm [16], in a procedure named $\texttt{heuristicPDP}(D,p,s)$ whose pseudocode is described in Algorithm 1. Since the k-means clustering is a stochastic method, it is repeated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

multiple times as long as the value of the lower bound keeps increasing (the authors of [4] stop after $s = 10$ iterations without being able to improve its value). An initial upper bound U is computed as the largest dissimilarity between any two points. Using the lower bound L, we build an initial sufficiently refined clustering $\mathcal{C}$ and a reduced dissimilarity matrix $D^{\mathcal{C}}$, using the procedure `initialClustering`(D, p, L), whose pseudocode is described in Algorithm 2. The main idea of the procedure is to find the clusters with the largest inter-node distances and split them into two, until the inter-node distance in all clusters is less than L (which implies that it is less than z^*). After these initial steps, the method uses two auxiliary sets S and W (with $S \subseteq W$), where S represents the set of optimal nonsingleton clusters ($S = \{C_i : |C_i| \geq 2, i = 1, \ldots, m\}$), and W is the complete optimal solution to the restricted pDP (w.r.t. $D^{\mathcal{C}}$). Iteratively, the sets S and W are used to refine the current clustering, resulting in a refined clustering $\mathcal{C}$ and dissimilarity matrix $D^{\mathcal{C}}$, using the procedure `splitAndAdd`$(S, W, \mathcal{C}, D^{\mathcal{C}})$, which is described in Algorithm 3. The resulting reduced pDP is then solved, yielding an upper bound U on the full problem, and its optimal solution is used to update the sets S, W. The solution procedure `solvePDP`$(D^{\mathcal{C}}, p, U, W)$ has two parts – a heuristic "preprocessing" method and an exact solver. The heuristic procedure is based on the observation that in a large number of iterations, the optimal value of the pDP problem does not decrease from one iteration to the next, which is a common feature of decremental relaxation schemes [17]. Therefore, before executing the exact solver, the heuristic procedure checks the best possible selection of p points out of the $p + 1$ points obtained from the `splitAndAdd` procedure. If the value of this solution equals the upper bound U from the previous iteration, the associated solution is then optimal, and there is no need to execute the exact solver. In our implementation, the exact solver comprise of solving the model (1)-(6) using the modelling package `JuMP` [18] in `Julia` [19], and the `GUROBI` solver [20]. The pseudocode of the `solvePDP` procedure is described in Algorithm 4. The whole decremental clustering algorithm is described in Algorithm 5. It also incorporates a possible knowledge on a lower bound L of the optimal value of (1)-(6), which will be explained in the forthcoming section.

III. DISCRETIZATIONS

The continuous variant of the pDP is extremely difficult to solve and the techniques for approaching it are usually bound to convex feasible spaces [21]. In order to apply the pDP framework developed earlier for general spaces, such as classrooms, restaurants, beaches, etc., the feasible space of the problem – the possible locations of the points – is discretized. This discretization is carried out by triangulation (or mesh generation) of the two dimensional feasible area, with the vertices of the triangles being the possible feasible points [22]. Naturally, a question arises about the relationship between the granularity of the triangulation and the objective

Algorithm 1 `heuristicPDP`(D, p, s)

$D, p, s \leftarrow$ **inputs**
$L \leftarrow 0$
$streak \leftarrow 0$
repeat
  $\mathcal{C} \leftarrow$ `k-means`(D, p)
  for $i = [1 : p]$ **do**
    $k_i \leftarrow$ point in C_i closest to its center
  end for
  $d \leftarrow \min(D(k_i, k_j) : 1 \leq i < j \leq p)$
  if $L < d$ **then**
    $streak \leftarrow streak + 1$
  else
    $L \leftarrow d$
    $streak \leftarrow 0$
  end if
until $streak = s$
return L

Algorithm 2 `initialClustering`(D, p, L)

$D, p, L \leftarrow$ **inputs**
$m \leftarrow p$
$\mathcal{C} \leftarrow$ `k-means`(D, p)
`compute` $D^{\mathcal{C}}$
while $\max(D^{\mathcal{C}}(i, i) : 1 \leq i \leq m) > L$ **do**
  $i^* \leftarrow \arg\max(D^{\mathcal{C}}(i, i) : 1 \leq i \leq m)$
  $m \leftarrow m + 1$
  $C^1_*, C^2_* \leftarrow$ `k-means`$(C_{i^*}, 2)$
  $C_{i^*} \leftarrow C^1_*$
  $C_m \leftarrow C^2_*$
  `recompute` $D^{\mathcal{C}}$
end while
return $\mathcal{C}, D^{\mathcal{C}}$

Algorithm 3 `splitAndAdd`$(S, W, \mathcal{C}, D^{\mathcal{C}})$

$S, W, \mathcal{C}, D^{\mathcal{C}} \leftarrow$ **inputs**
if $S = \emptyset$ **then**
  $\mathcal{C} \leftarrow \mathcal{C}$, $D^{\mathcal{C}} \leftarrow D^{\mathcal{C}}$ (i.e., do nothing)
else
  $m \leftarrow$ `size` $D^{\mathcal{C}}$
  $(s^*, w^*) \leftarrow \arg\min(D^{\mathcal{C}}(s, w) : s \in S, w \in W)$
  if $w^* \in S$ **then**
    $i^* \leftarrow \arg\max(D^{\mathcal{C}}(u, u) : u \in \{s^*, w^*\})$
  else
    $i^* \leftarrow s^*$
  end if
  $C^1_*, C^2_* \leftarrow$ `k-means`$(C_{i^*}, 2)$
  $C_{i^*} \leftarrow C^1_*$
  $C_{m+1} \leftarrow C^2_*$
  `recompute` $D^{\mathcal{C}}$
end if
return $\mathcal{C}, D^{\mathcal{C}}$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

Algorithm 4 `solvePDP` $(D^{\mathcal{C}}, p, U, W)$

$D^{\mathcal{C}}, p, U, W \leftarrow$ **inputs**
for $k = [1 : p+1]$ **do**
  $U_h(k) \leftarrow \min(D^{\mathcal{C}}(i,j) : 1 \le i,j \le p+1, i \ne k, j \ne k)$
end for
if $\max(U_h(k) : 1 \le k \le p+1) = U$ **then**
  $k^* \leftarrow \arg\max(U_h(k) : 1 \le k \le p+1)$
  $W \leftarrow \{W(i) : 1 \le i \le p+1, i \ne k^*\}$
  $U \leftarrow U$
else
  $U, W \leftarrow$ `solve` (1)-(6)
end if
return U, W

Algorithm 5 `decrementalClustering` (D, p, L)

$D, p, L \leftarrow$ **inputs**
$L' \leftarrow$ `heuristicPDP`$(D, p, 10)$
$L \leftarrow \max(L, L')$, $U \leftarrow \max(D(i,j) : 1 \le i < j \le n)$
$\mathcal{C}, D^{\mathcal{C}} \leftarrow$ `initialClustering`(D, p, L)
$S \leftarrow \emptyset$, $W \leftarrow \emptyset$
repeat
  $\mathcal{C}, D^{\mathcal{C}} \leftarrow$ `splitAndAdd`$(S, W, \mathcal{C}, D^{\mathcal{C}})$
  $U, W \leftarrow$ `solvePDP` $(D^{\mathcal{C}}, p, U, W)$
  $S \leftarrow \{w \in W : |C_w| \ge 2\}$
until $S = \emptyset$ **or** $L = U$
return $U, X \leftarrow \{C_w : w \in W\}$

value of the optimal solution of the associated pDP – the finer the mesh, the better the solution (with higher smallest distance between any two selected points). We address this issue by devising a mesh refinement scheme and solving a series of pDPs on a progressively finer mesh. The mesh refinement works as follows: First, the area of each triangle in the triangulation is computed. The triangles with larger than average area are then split into 4 triangles by adding points is the middle of their sides. The process is illustrated in Figure 1. A side effect of the refinement scheme is that by solving the pDP on a coarser mesh, we obtain a guaranteed lower bound on the optimal value of the pDP on a finer mesh.

Another "discretization" can be devised in the space of possible values of the dissimilarity matrix D. As the problem gets more difficult with more unique values in D, with each unique value having a separate binary variable in the model (1)-(6), we can greatly reduce the number of these added variables by rounding the elements of D. The trade-off between the optimal objective value and computational efforts of these two discretization schemes are investigated in the following section. The pseudocode of the method used in the computational experiments, called `refinePDP`(D, p, L, r, T), is described in Algorithm 6. It supposes that an initial mesh with p and D is available. The other inputs are a lower bound L (if there is no prior knowledge about a possible lower bound, then $L = 0$), a rounding factor r (with $r = -1$ being rounding to the nearest tenths, $r = 1$ to nearest integers, $r = 2$ to nearest tens, etc., and $r = 0$ no rounding) and a time limit for the computation T.

Algorithm 6 `refinePDP` (D, p, L, r, T)

$D, p, L, r, t \leftarrow$ **inputs**
$D' \leftarrow$ `round`(D, r)
$U, X \leftarrow$ `decrementalClustering`(D', p, L)
$L \leftarrow U$
repeat
  $D, p \leftarrow$ `refineMesh`(D, p)
  $D' \leftarrow$ `round`(D, r)
  $U, X \leftarrow$ `decrementalClustering`(D', p, L)
  $L \leftarrow U$
until `runTime` $> T$
return U, X

IV. COMPUTATIONAL EXPERIMENTS

We investigate the effect of discretization on two examples. The first one is a general shape depicted in Figure 1 and the second one an auditorium-like shape shown in Figure 2. In both examples, we examine the computational efforts to solve the pDP problem for progressively finer mesh granularity and for different values of $p \in \{5, 10, 15, 30\}$ and $r \in \{0, 1, 2\}$. In both examples, the time limit T was set to 24 hours. For the first example, the maximal distance between any two points ("the problem diameter") was $\max(D) = 3240.8$, for the second example, it was $\max(D) = 2180$. For each problem instance, we report: n the number of points, Δ the square root of the area of the largest triangle in the mesh (a useful measure of the granularity of the grid), r the rounding factor, k_{max} the number of distinct elements in D', z'^* the optimal objective value of the problem with D', z^r the "real" objective value (without rounding), and t the time it took to find the optimum. If the computations were not finished (the time limit was reached), the best upper bound U is reported in square brackets. If the computation of the instance was terminated prematurely, because during the computation, the upper bound U was equal to the lower bound L from the solution of coarser discretization (i.e., the finer discretization did not improve on the optimal value of the solution) the instance is marked with a '*'. The instances that were not computed, because the time limit was already reached, are marked by a '–'. The computations were carried out on an ordinary computer with 3.2 GHz i5-4460 CPU and 16 GB RAM.

The numerical results of the computations are reported in Table 1 for the first example and Table 2 for the second one. The optimal placements (the ones with the best value of z^r) are shown in Figures 3-6 for the first example and in Figures 7-10 for the second one. The first general observations is that in order to decrease Δ by half, the number of points n needs to be roughly quadrupled (which follows from the way the mesh gets refined). The second general observation can be made about the impact of the rounding factor r: Apart from a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

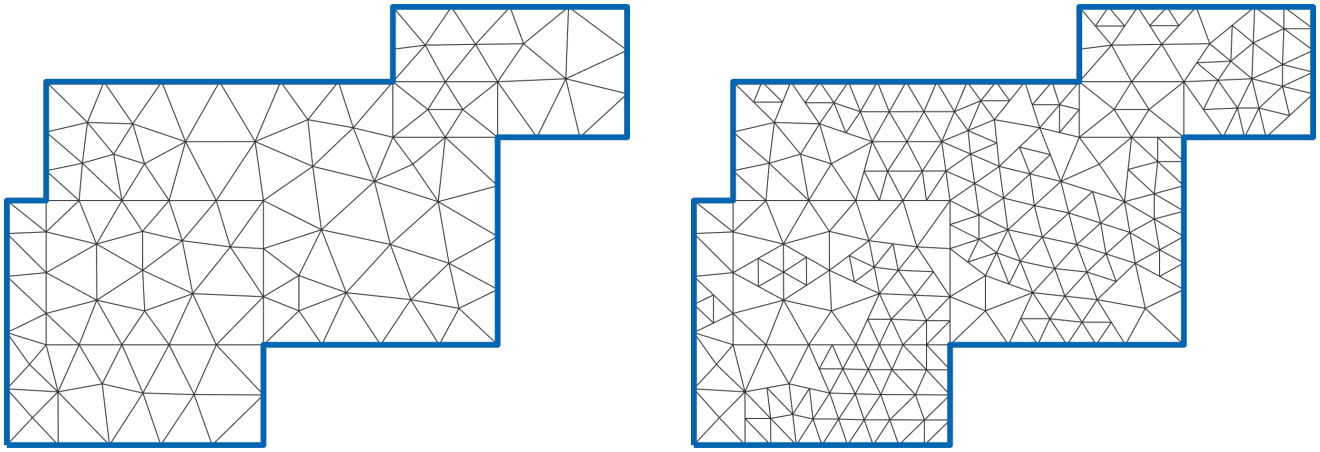


FIGURE 1. Initial mesh (left) and mesh after one round of refinement (right), first example.

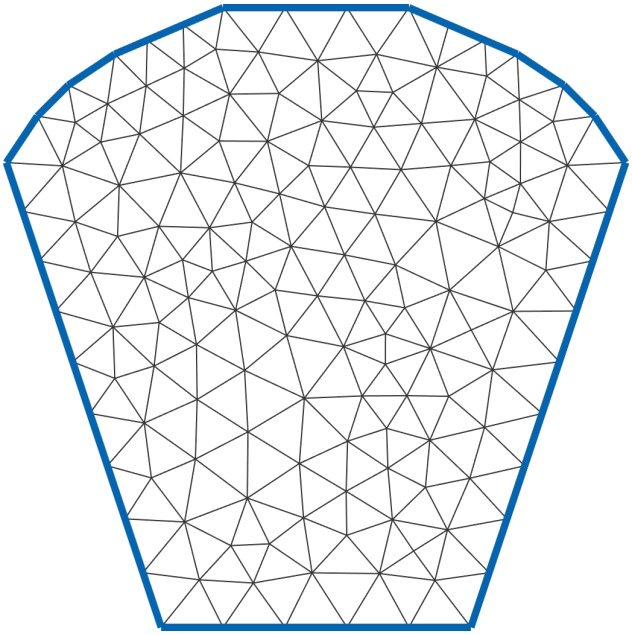


FIGURE 2. Initial mesh, second example.

single instance, there was no difference in the "real" objective value between instances with $r = 0$ and $r = 1$, while there was a huge difference between the computational time t. From these computational experiments, there is no doubt that the rounding procedure presents a substantial benefit, as the instances with rounding were computed around an order of magnitude faster than the instances without rounding, and some large instances could not be computed within the time limit without the use of rounding.

The difference between $r = 1$ and $r = 2$ is much more nuanced. In 83 % of the instances, the computations with $r = 2$ were faster. On the other hand, using $r = 2$ instead of $r = 1$ results on average in 0.43 % worse value of z^r. Premature termination is also more prevalent in the $r = 2$ case. Of the 48 successfully computed instances it occurred 14 times for $r = 2$, compared to 8 times for $r = 1$.

The coarseness of the mesh naturally plays a crucial role in both the objective value z^r and the computational time t, with finer meshes having higher objective value z^r, but because of the increase in the number of variables, take progressively longer to compute. The improvement of the objective value z^r based on the mesh refinement is captured in Figure 11, which shows the percentage improvements caused by the increases in the number of mesh points n. Additionally, the value of p also has an extensive impact on the computational time. Similar to the findings in [4], we also find that the decremental clustering scheme works very well for smaller values of p, but the computations become progressively more costly as p increases. The value of $p = 30$ seems to be close to the limit of applicability of the method. On the one hand, it means that in the context of social distancing it is well applicable for use in situations, when the available space does not allow for more than 30 persons, such as in classrooms, restaurants, or offices. On the other hand, it still can be used to compute valid upper bounds on the objective value even for larger problems, which can be explored by various heuristics.

There is also a significant difference in the "difficulty" between the two examples. Although the number of points n and unique values k_{max} were similar for both examples in the individual mesh refinement steps, the computational times differ quite a lot, mainly for larger values of p. This can be attributed to the "dual degeneracy" [4] occurring when a larger number of clusters can be rearranged from one iteration to the next to find solutions of the same cost. The second example is more symmetric than the first one, meaning that it has a larger number of the possible rearrangements. Naturally, the optimal placements in Figures 7-10 have a straightforward "symmetric" counterpart with the same objective value.

Lastly, the hexagonal pattern, that seems to emerge in Figures 6 and 10 (both with $p = 30$) is no accident – the hexagonal pattern is optimal for many location problems (including the p-dispersion problem) with numerous facilities covering a large area [23].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

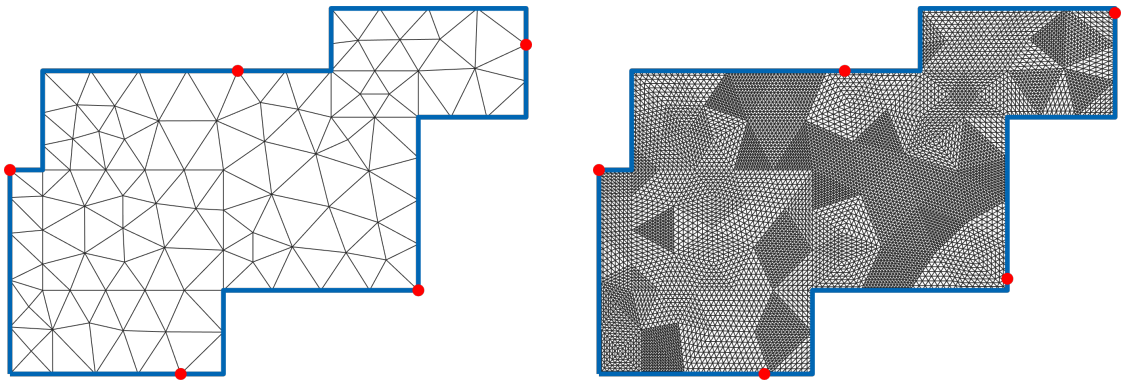


FIGURE 3. Optimal placements, $p = 5$, $r = 1$. Initial mesh with $n = 102$, $z^r = 1{,}273.88$ (left). Final mesh with $n = 8{,}745$, $z^r = 1{,}338.09$ (right).

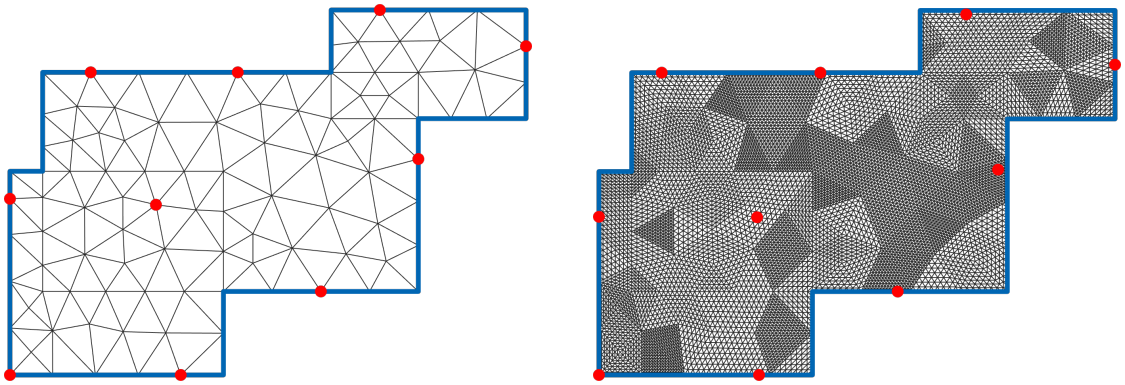


FIGURE 4. Optimal placements, $p = 10$, $r = 1$. Initial mesh with $n = 102$, $z^r = 749.72$ (left). Final mesh with $n = 8{,}745$, $z^r = 805.25$ (right).

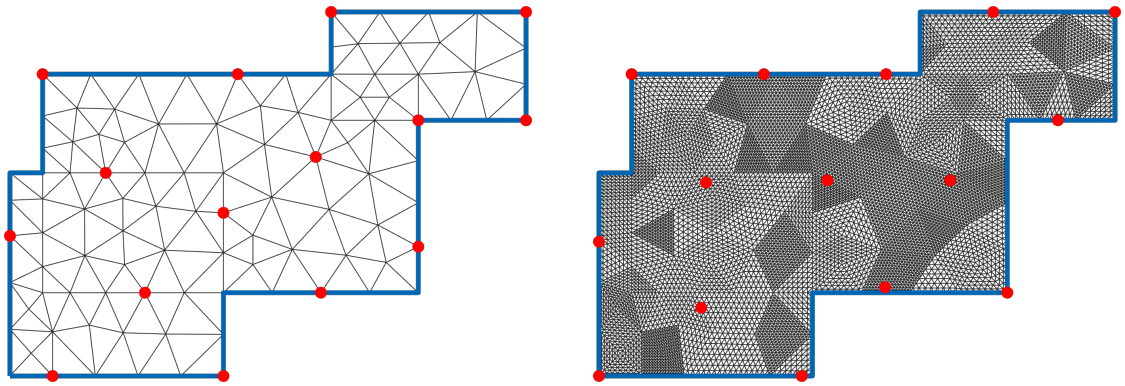


FIGURE 5. Optimal placements, $p = 15$, $r = 1$. Initial mesh with $n = 102$, $z^r = 552.63$ (left). Final mesh with $n = 8{,}745$, $z^r = 622.75$ (right).

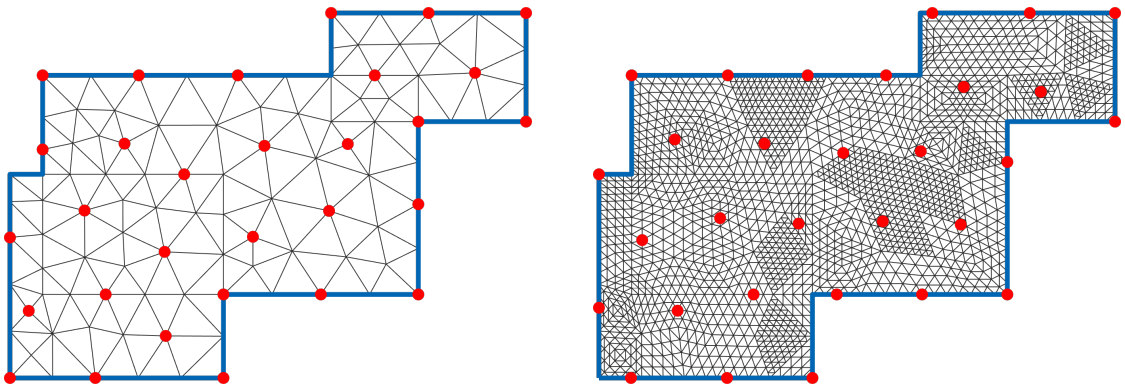


FIGURE 6. Optimal placements, $p = 30$, $r = 2$. Initial mesh with $n = 102$, $z^r = 325.77$ (left). Final mesh with $n = 1{,}891$, $z^r = 395.42$ (right).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

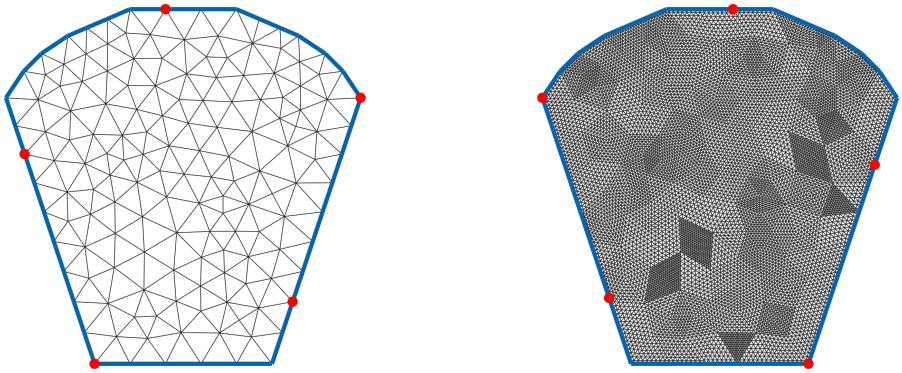


FIGURE 7. Optimal placements, $p = 5$, $r = 1$. Initial mesh with $n = 150$, $z^r = 1{,}139.49$ (left). Final mesh with $n = 9{,}313$, $z^r = 1{,}183.01$ (right).

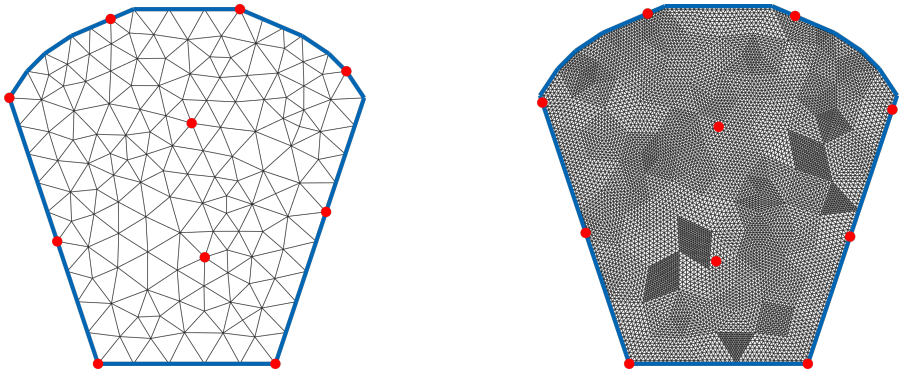


FIGURE 8. Optimal placements, $p = 10$, $r = 1$. Initial mesh with $n = 150$, $z^r = 694.62$ (left). Final mesh with $n = 9{,}313$, $z^r = 747.72$ (right).

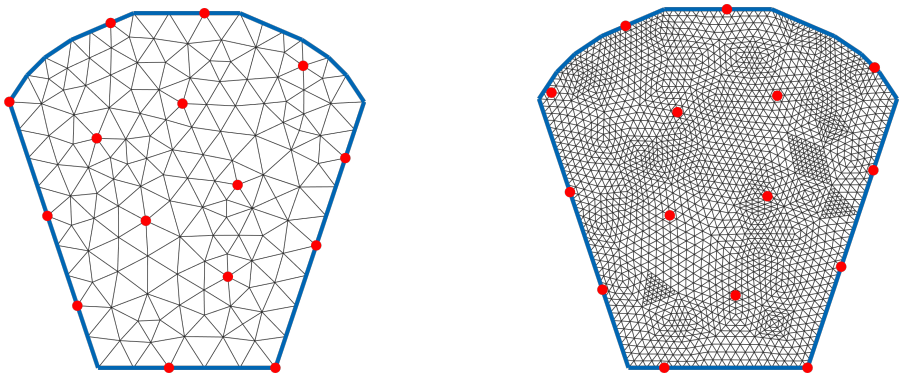


FIGURE 9. Optimal placements, $p = 15$, $r = 1$. Initial mesh with $n = 150$, $z^r = 518.40$ (left). Final mesh with $n = 2{,}273$, $z^r = 555.93$ (right).

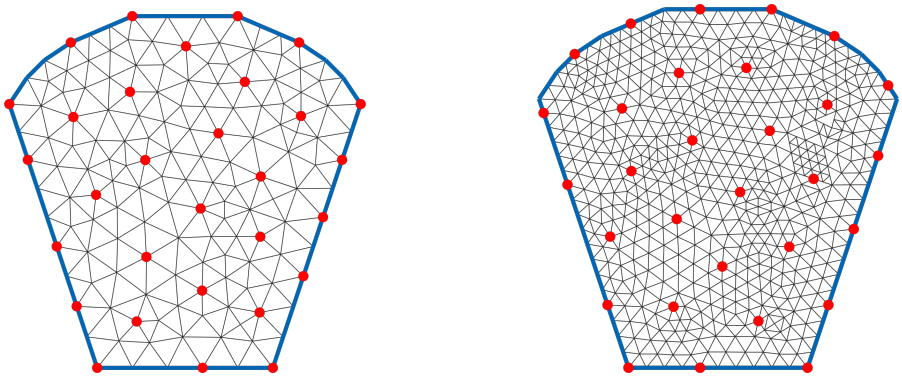


FIGURE 10. Optimal placements, $p = 30$, $r = 1$. Initial mesh with $n = 150$, $z^r = 323.69$ (left). Final mesh with $n = 575$, $z^r = 352.96$ (right).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

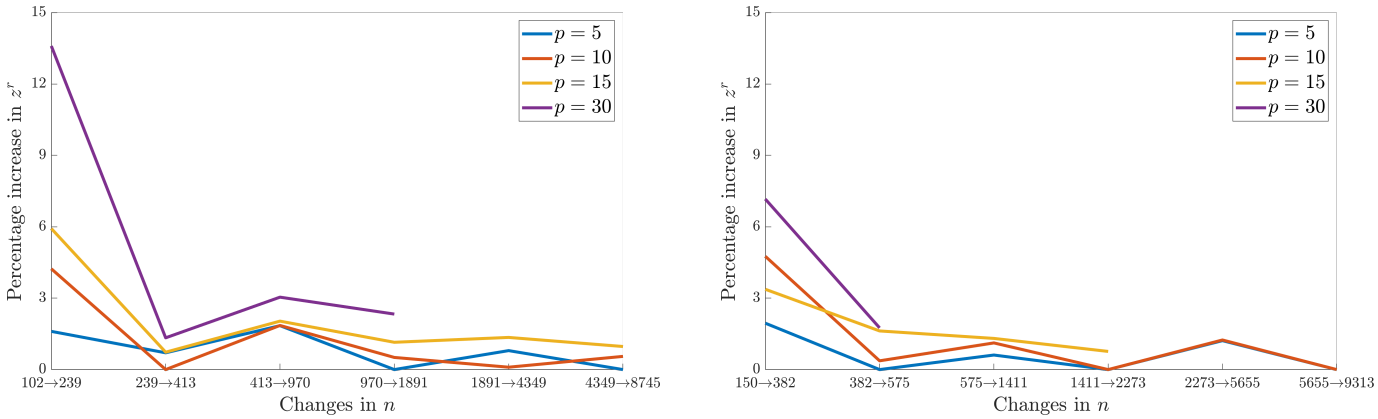


FIGURE 11. The effect of the mesh refinement on the value of z^r. First example on the left, second example on the right.

TABLE 1. Results of the computation, first example, $\max(D) = 3{,}240.8$.

				$p = 5$			$p = 10$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	5,043	1,273.88	1,273.88	0.25	749.72	749.72	10.9
102	198.4	1	2,060	1,274	1,273.88	0.42	750	749.72	1.96
		2	293	1,270	1,273.88	0.95	750	749.72	2.57
		0	27,919	1,294.33	1,294.33	1.58	781.46	781.46	107
239	141.0	1	2,688	1,294	1,294.33	1.59	781	781.46	46.8
		2	304	1,290	1,287.46	0.62	780	781.46	7.07
		0	83,555	1,303.45	1,303.45	1.84	781.46*	781.46	475*
413	92.7	1	2,918	1,303	1,303.45	1.61	781*	781.46	35.4*
		2	314	1,300	1,303.45	0.50	780*	781.46	19.3*
		0	459,211	1,327.51	1,327.51	5.25	795.95	795.95	991
970	66.8	1	3,029	1,328	1,327.51	1.31	796	795.95	198
		2	317	1,330	1,327.51	1.00	800	795.95	66.6
		0	1,749,744	1,327.51	1,327.51	31.4	800.01	800.01	4,061
1,891	43.2	1	3,113	1,328*	1,327.51	2.51*	800	800.01	407
		2	322	1,330*	1,327.51	2.95*	800*	795.95	242*
		0	9,220,027	1,338.09	1,338.09	41.2	[802.98]	[802.98]	T = 24h
4,349	30.2	1	3,157	1,338	1,338.09	16.7	801	800.81	1,426
		2	324	1,340	1,336.61	15.2	800*	795.95	1,170*
		0	37,239,029	1,338.09*	1,338.09	4,134*	–	–	–
8,745	19.5	1	3,202	1,338*	1,338.09	43.5*	805	805.25	2,546
		2	325	1,340*	1,336.61	54.3*	810	805.07	1,922

				$p = 15$			$p = 30$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	5,043	552.63	552.63	50.6	325.77	325.77	12.6
102	198.4	1	2,060	553	552.63	6.73	326	325.77	32.3
		2	293	550	546.54	4.41	330	325.77	23.3
		0	27,919	585.37	585.37	195	370.06	370.06	256
239	141.0	1	2,688	585	585.37	69	370	370.06	56.7
		2	304	590	585.37	22.9	370	365.37	33.2
		0	83,555	589.65	589.65	1,162	375.00	375.00	4,707
413	92.7	1	2,918	590	589.65	135	375	375.00	415
		2	314	590*	585.37	66.1*	380	375.00	206
		0	459,211	601.68	601.68	26,771	[390.82]	[390.82]	T = 24h
970	66.8	1	3,029	602	601.65	1,153	386	386.41	4,635
		2	317	600	595.33	791.8	390	385.07	3,334
		0	1,749,744	[614.82]	[614.82]	T = 24h	–	–	–
1,891	43.2	1	3,113	609	608.55	4,051	395	395.42	36,017
		2	322	610	606.28	2,837	400	395.42	24,706
		0	9,220,027	–	–	–	–	–	–
4,349	30.2	1	3,157	617	616.77	15,699	[413]	[413]	T = 24h
		2	324	620	615.21	16,695	[420]	[420]	T = 24h
		0	37,239,029	–	–	–	–	–	–
8,745	19.5	1	3,202	623	622.75	21,044	–	–	–
		2	325	620*	615.21	18,185*	–	–	–

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

TABLE 2. Results of the computation, second example, $\max(D) = 2{,}180$.

				$p = 5$			$p = 10$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	10,901	1,139.49	1139.49	21.9	694.62	694.62	87.1
150	143.4	1	1,846	1,139	1139.49	4.05	695	694.62	37.0
		2	205	1,140	1139.49	0.59	690	687.56	5.28
		0	71,690	1,161.70	1161.70	61.4	727.68	727.68	26.3
382	107.2	1	2,010	1,162	1161.70	5.65	728	727.68	12.7
		2	212	1,160	1161.70	6.68	730	727.68	10.3
		0	163,067	1,161.70*	1161.70	36.4*	730.36	730.36	202
575	69.7	1	2,062	1,162*	1161.70	47.59*	730	730.36	100
		2	215	1,160*	1161.70	5.65*	730*	727.68	18.4
		0	983,031	1,168.8	1168.82	374	738.54	738.54	3,741
1,411	53.0	1	2,120	1,169	1168.82	169	739	738.54	312
		2	217	1,170	1168.82	26.3	740	737.39	240
		0	2,258,704	1,168.8*	1168.82	104	738.54*	738.54	2,342*
2,273	34.5	1	2,144	1,169*	1168.82	170*	739*	738.54	460*
		2	218	1,170*	1168.82	21.9*	740*	727.68	138*
		0	15,805,824	1,183.01	1183.01	2,272	[767.81]	[767.81]	$T = 24$h
5,655	26.2	1	2,159	1,183	1183.01	263	748	747.72	1,185
		2	218	1,180	1177.31	227	750	746.81	1,215
		0	42,931,260	1,183.01*	1183.01	1,541	–	–	–
9,313	16.7	1	2,168	1,183*	1183.01	252*	748*	747.72	3,818*
		2	219	1,180*	1177.31	178*	750*	746.81	2,720*

				$p = 15$			$p = 30$		
n	Δ	r	k_{max}	z^*	z^r	t [s]	z^*	z^r	t [s]
		0	10,901	518.40	518.40	84.2	323.69	323.69	96.0
150	143.4	1	1,846	518	518.40	47.4	324	323.69	65.2
		2	205	520	518.40	8.93	320	315.65	35.9
		0	71,690	535.89	535.89	3,754	346.88	346.88	5,231
382	107.2	1	2,010	536	535.89	415	347	346.88	439
		2	212	540	535.04	155	350	345.76	284
		0	163,067	[562.73]	[562.73]	$T = 24$h	[361.08]	[361.08]	$T = 24$h
575	69.7	1	2,062	545	544.59	793	353	352.96	2,763
		2	215	540*	535.04	611*	350	345.76	1,577*
		0	983,031	–	–	–	–	–	–
1,411	53.0	1	2,120	552	551.72	12,203	[366]	[366]	$T = 24$h
		2	217	550	546.01	9,190	[360]	[366]	$T = 24$h
		0	2,258,704	–	–	–	–	–	–
2,273	34.5	1	2,144	556	555.93	51,344	–	–	–
		2	218	560	555.21	55,682	–	–	–
		0	15,805,824	–	–	–		–	–
5,655	26.2	1	2,159	[572]	[572]	$T = 24$h	–	–	–
		2	218	[570]	[570]	$T = 24$h	–	–	–

V. CONCLUSIONS

In this paper we have studied the problem of locating persons in a given area, that should abide to social distancing measures such as those arising in the time of COVID-19 and similar viruses. We have argued that the $p-$dispersion problem can be used to efficiently model these situations. We devised a discretization scheme that was build on top of the decremental clustering method to get computationally attainable solutions, which worked very well in the computational study on the two artificial examples, especially for smaller values of p. We have investigated the effect of rounding of the dissimilarity matrix D on the computational effort and conclude that it is an indispensable part of the discretization scheme that has virtually no disadvantages in terms of the quality of the obtained solution. We have also seen the substantial increase in computational efforts for higher values of p, which can be contributed to the "dual degeneracy" of the clustering scheme.

For future research, fast heuristics that run parallel to the decremental clustering scheme might further improve on the computational time and the size of the problems that are solvable by the presented method. Also, a mesh refinement scheme based on the current optimal placement (instead of the triangle size as presented here) could lead to improvements in the objective value.

REFERENCES

[1] V. Chamola, V. Hassija, V. Gupta, and M. Guizani, "A Comprehensive Review of the COVID-19 Pandemic and the Role of IoT, Drones, AI, Blockchain, and 5G in Managing its Impact", *IEEE Access*, vol. 8, pp. 90225–90265, May 2020. DOI: 10.1109/ACCESS.2020.2992341

[2] I. D. Moon and S. S. Chaudhry, "An analysis of network location problems with distance constraints", *Management Science*, vol. 30, no. 3, pp. 290–307, Mar. 1984. DOI: 10.1287/mnsc.30.3.290

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3016724, IEEE Access

[3] D. Sayah and S. Irnich, "A new compact formulation for the discrete p-dispersion problem," *European Journal of Operational Research*, vol. 256, no. 1, pp. 62–67, Jan. 2017. DOI: 10.1016/j.ejor.2016.06.036

[4] C. Contardo, "Decremental Clustering for the Solution of p-Dispersion Problems to Proven Optimality," *INFORMS Journal on Optimization*, vol.2, no. 2, pp. 134-–144, Apr. 2020. DOI: 10.1287/ijoo.2019.0027

[5] E. Erkut, "The discrete p-dispersion problem," *European Journal of Operational Research*, vol. 46, no. 1, pp. 48–60, May 1990. DOI: 10.1016/0377-2217(90)90297-O

[6] M. J. Kuby, "Programming models for facility dispersion: the p-dispersion and maxisum dispersion problems," *Geographical Analysis*, vol. 19, no. 4, pp. 315–329, Oct. 1987. DOI: 10.1111/j.1538-4632.1987.tb00133.x

[7] E. Erkut and S. Neuman, "Analytical models for locating undesirable facilities," *European Journal of Operational Research*, vol. 40, no. 3, pp. 275-–291, Jun. 1989. DOI: 10.1016/0377-2217(89)90420-7

[8] B. Saboonchi, P. Hansen, and S. Perron, "MaxMinMin p-dispersion problem: A variable neighborhood search approach," *Computers & Operations Research*, vol. 52, Part B, pp. 251—259, Dec. 2014. DOI: 10.1016/j.cor.2013.09.017

[9] D. Pisinger, "Upper bounds and exact algorithms for p-dispersion problems," *Computers & Operations Research*, vol. 33, no. 5, pp. 1380-–1398, May 2006. DOI: 10.1016/j.cor.2004.09.033

[10] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254-–265, Apr. 1987. DOI: 10.1287/opre.35.2.254

[11] O. Braysy and M. Gendreau, "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms," *Transportation Science*, vol. 39, no. 1, pp. 104-–118, Feb. 2005. DOI: 10.1287/trsc.1030.0056

[12] J. E. Beasley, "Route first–cluster second methods for vehicle routing," *Omega*, vol. 11, no. 4, pp. 403-–408, 1983. DOI: 10.1016/0305-0483(83)90033-6

[13] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: A review," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 179-–200, Mar. 2014. DOI: 10.1016/j.trc.2014.01.011

[14] D. Chen and R. Chen, "New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems," *Computers & Operations Research*, vol. 36, no. 5, pp. 1646-–1655, May 2009. DOI: 10.1016/j.cor.2008.03.009

[15] C. Contardo, M. Iori, and R. Kramer, "A scalable exact algorithm for the vertex p-center problem," *Computers & Operations Research*, vol. 103, pp. 211-–220, Mar. 2019. DOI: 10.1016/j.cor.2018.11.006

[16] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, University of California Press, pp. 281-–297, 1967.

[17] D. Aloise and C. Contardo, "A sampling-based exact algorithm for the solution of the minimax diameter clustering problem," *Journal of Global Optimization*, vol. 71, pp. 613-–630, Mar. 2018. DOI: 10.1007/s10898-018-0634-1

[18] I. Dunning, J. Huchette, and M. Lubin, "JuMP: a modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, May 2017. DOI: 10.1137/15M1020575

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65-–98, Feb. 2017. DOI: 10.1137/141000671

[20] *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2020. [Online]. Available: http://www.gurobi.com/.

[21] Z. Drezner and E. Erkut, "Solving the Continuous p-Dispersion Problem Using Non-Linear Programming", *The Journal of the Operational Research Society*, vol. 46, no. 4, pp. 516–520, Apr. 1995. DOI: 10.2307/2584599

[22] P. L. George, "Automatic mesh generation and finite element computation", *Handbook of Numerical Analysis*, vol. 4, pp. 69–190, 1996. DOI: 10.1016/S1570-8659(96)80003-2

[23] T. Drezner and Z. Drezner, "Leader-Follower Models in Facility Location", *Spatial Interaction Models: Facility Location Using Game Theory*, Ed.: L. Mallozi, E. D'Amato, P. M. Pardalos, Springer, pp. 73–104, 2017. DOI: 10.1007/978-3-319-52654-6

JAKUB KUDELA received his M.S. degree in Mathematical Engineering from Brno University of Technology in 2014 and received his Ph.D. degree in Applied Mathematics from Brno University of Technology in 2019.

From 2018, he works as a Research Assistant with the Institute of Automation and Computer Science, Brno University of Technology. His research interest includes the development of computational methods for various optimization problems and engineering applications.

A2

Novel Zigzag-based Benchmark Functions for Bound Constrained Single Objective Optimization

Jakub Kudela
Institute of Automation and Computer Science
Faculty of Mechanical Engineering
Brno University of Technology
Brno, Czech Republic
Email: Jakub.Kudela@vutbr.cz

***Abstract*—The development and comparison of new optimization methods in general, and evolutionary algorithms in particular, rely heavily on benchmarking. In this paper, the construction of novel zigzag-based benchmark functions for bound constrained single objective optimization is presented. The new benchmark functions are non-differentiable, highly multimodal, and have a built-in parameter that controls the complexity of the function. To investigate the properties of the new benchmark functions two of the best algorithms from the CEC'20 Competition on Single Objective Bound Constrained Optimization, as well as one standard evolutionary algorithm, were utilized in a computational study. The results of the study suggest that the new benchmark functions are very well suited for algorithmic comparison.**

I. Introduction

Benchmarking has a crucial role in the development of novel search algorithms as well as in the assessment and comparison of contemporary algorithmic ideas [1]. One of the subclasses of the derivative-free optimization methods are Evolutionary Algorithms (EAs), which proved to be very powerful for solving black-box optimization problems. However, because EAs generally lack theoretical performance results, their development and performance comparison rely mainly on benchmarking. Benchmarking experiments are set up for performance comparison on given problem classes and should support the selection of a suitable algorithm for a given real-world application [2]. Benchmarks are also used to qualify the theoretical predictions of the behaviour of algorithms [3].

There are two main lines of development in benchmarking for EAs, the IEEE Congress on Evolutionary Computation (CEC) competitions [4] and the Comparing Continuous Optimizer (COCO) benchmark suite [6]. The COCO suite is a platform for comparing unconstrained continuous optimizers for numerical optimization. An advantage of the COCO platform is a large number of algorithm results available for comparison. Up to now, 231 distinct (classical as well as contemporary) algorithms have been tested on the COCO suite. On the other hand, the competitions that are organized every year during the CEC aim to compare state-of-the-art stochastic search algorithms. The CEC competitions provide specific test environments for algorithm assessment and comparison. The benchmark functions are constructed from a set of popular benchmark functions, such as the Rastrigin's function, Rosenbrock's function, Griewank's function, Ackley's function, Schwefel's function, and others [4]. A tunable benchmark function for combinatorial problems was recently introduced in [5].

In this paper, we propose novel zigzag-based benchmark functions for bound constrained single objective optimization, that are non-differentiable and highly multimodal. The rest of the paper is organized as follows. Section II introduces the individual components of the new benchmark functions and provides insight into their construction. In Section III we report on computational experiments where we compare two state-of-the-art algorithms and one standard EA on a set of problems that utilize the new benchmark functions. The conclusions and future research are described in Section IV.

II. The Novel Benchmark Functions

The new benchmark functions are constructed as follows. First, we devise a "zigzag" function $z(x)$. For given parameters $k > 0, m > 0$ the zigzag function $z(x)$ at a point $x \in \mathbb{R}$ is computed as:

$$z(x) = \begin{cases} m\left(\frac{1}{2} + (-1)^{\lceil kx \rceil}\left(\frac{\lceil kx \rceil + \lfloor kx \rfloor}{2} - kx\right)\right), \text{if } (kx) \notin \mathcal{Z} \\ 0, \quad \text{if } \frac{kx}{2} \in \mathcal{Z} \\ m, \quad \text{otherwise,} \end{cases}$$

where $\frac{2}{k}$ is the period and m is the amplitude of the zigzag function, as depicted in Fig 1. The next step is a construction of a multimodal function $f(x)$, which is a sum of an absolute

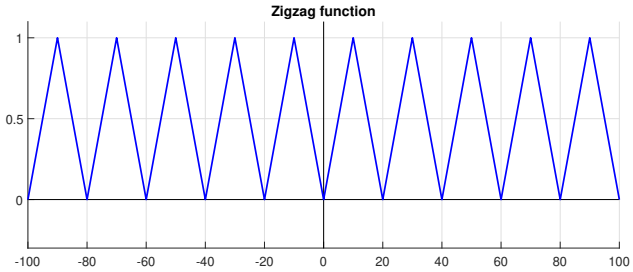


Fig. 1: Zigzag function $z(x)$ with $k = 0.1, m = 1$.

978-1-7281-8393-0/21/$31.00 ©2021 European Union

value of a high degree polynomial with one root in zero and an absolute value function. The scaling of these two parts is such that the function values on the interval $[-200, 200]$ lie between $[0, 200]$ (this allows us to compose the function with itself any number of times without running into severe numerical difficulties). The reason we care about the behaviour of the function on the interval $[-200, 200]$, and not just the interval $[-100, 100]$ where the optimization will be carried out, is because the benchmark functions will include a shift (and a rotation/scaling). The "polynomial" part of the function f is then multiplied with the zigzag function $z(x)$. The particular choice for the function $f(x)$ in this paper is the following:

$$f(x)=3\cdot 10^{-9}|(x-50)(x-190)x(x+70)(x+180)|z(x)+0.2|x|$$

where the individual parts of the function are shown in Fig. 2, and the impact of varying the parameter k of the zigzag function $z(x)$ is shown in Fig. 3. Fig. 3 also shows the structure of the function $f(x)$ composed with itself, i.e. the function $f(f(x))$, for different values of k. The function f has a single global optimum point in 0, is non-differentiable and highly multimodal (the "degree of multimodality" depending on the parameter k). Finally, the two proposed benchmark

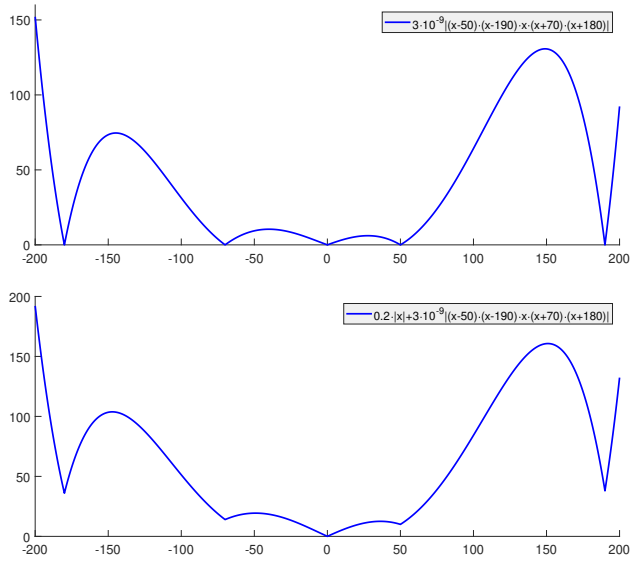


Fig. 2: Partial construction of the function $f(x)$.

functions $F_1(\mathbf{x})$ and $F_2(\mathbf{x})$, for $\mathbf{x} = [x_1, \ldots, x_D]^T$ and $\mathbf{x} \in [-100, 100]^D$, are the following:

$$f_1(\mathbf{x}) = \sum_{i=1}^{D} f(x_i)$$
$$F_1(\mathbf{x}) = f_1(\mathbf{M_1}(\mathbf{x} - \mathbf{s_1}))$$
$$f_2(\mathbf{x}) = \sum_{i=1}^{D} f(f(x_i))$$
$$F_2(\mathbf{x}) = f_2(\mathbf{M_2}(\mathbf{x} - \mathbf{s_2}))$$

where $\mathbf{s_1}, \mathbf{s_2} \in [-100, 100]^D$ are random shifts of the optimal solution and $\mathbf{M_1}, \mathbf{M_2}$ are random rotation/scaling matrices, with eigenvalues in the range $[0.5, 1]$. The contour and surface plots of $F_1(x)$ and $F_2(x)$ for $D = 2$ and different values of the parameter k can be seen in Fig. 4. The rotation/scaling matrices were constructed in the following way: for a given dimension D, we generate a random square matrix $\mathbf{A}$, and construct a matrix $\mathbf{B} = \mathbf{A}'\mathbf{A}$. Then we get matrices $\mathbf{P}, \mathbf{R}, \mathbf{Q}$ from the singular value decomposition of $\mathbf{B}$, i.e. $\mathbf{B} = \mathbf{PQR}'$. Lastly, we generate a D dimensional vector $\mathbf{v}$ whose individual components are uniformly distributed random values on the interval [0.5,1], and we construct the matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{P} \cdot \text{diag}(\mathbf{v}) \cdot \mathbf{R}'$, where diag$(\cdot)$ transfors a vector into a diagonal matrix. This ensures that the eigenvalues of $\mathbf{M}$ lie on the interval [0.5,1]. The rotation/scaling matrix is an integral part of the benchmark function [7], as it creates additional difficulty for the optimization algorithms [8].

III. Computational Experiments

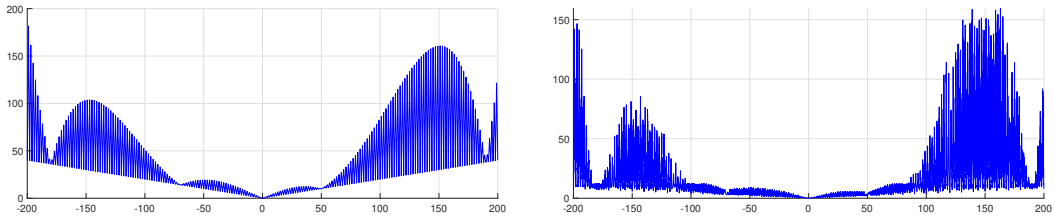
A. Optimization Algorithms and Experimental Settings

The first algorithm we chose for the computational comparison, is the canonical particle swarm optimization (PSO) algorithm that simulates swarm behaviors of social animals such as the bird flocking or fish schooling [9]. The particular implementation and parameter setting for the PSO is the one that was shipped along with the benchmark suite for the CEC'20 Competition on Single Objective Bound Constrained Optimization [4].

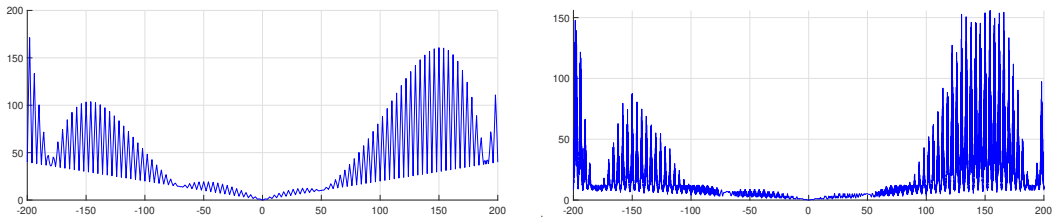
The second algorithm for the comparison is the winner of the CEC'20 Competition on Single Objective Bound Constrained Optimization, the Improved Multi-operator Differential Evolution (IMODE) algorithm [10]. This algorithm utilizes multiple differential evolution operators and a sequential quadratic programming local search procedure for accelerating its convergence.

The third algorithm for the comparison is the runner-up of the same competition, the Adaptive Gaining-Sharing Knowledge (AGSK) based algorithm [11]. This algorithm extends and improves the original GSK [12] algorithm by adding adaptive settings to the two important control parameters: the knowledge factor and the knowledge ratio, which control junior and senior gaining and sharing phases between the solutions during the optimization loop.

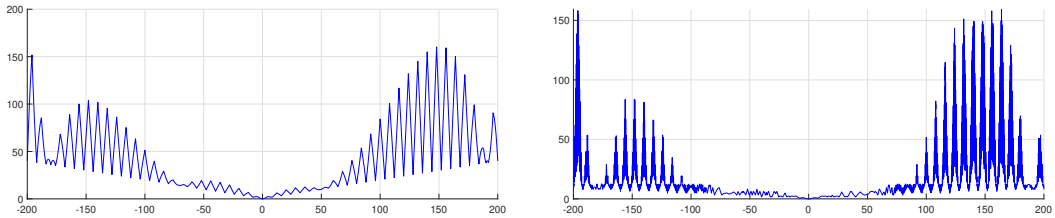
We use the same benchmark rules as the CEC'20 competition: the three algorithms are evaluated on the two benchmark functions with $D = [5, 10, 15, 20]$ dimensions, parameter $k = [2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}]$, and a search space of $[-100, 100]^D$. As a change in the parameter m can be though of as a scaling of the polynomial part of the function f, we set it to $m = 1$. The maximum number of function evaluations were set to 50,000, 1,000,000, 3,000,000 and 10,000,000 fitness function evaluations for problems with $D = [5, 10, 15, 20]$, respectively. All algorithms were run 30 times to obtain representative results. For every run, if the function value of the solution was



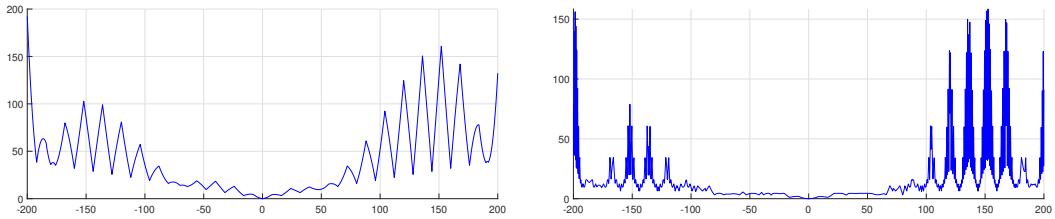
$f(x)$ and $f(f(x))$ for $k = 2^0$



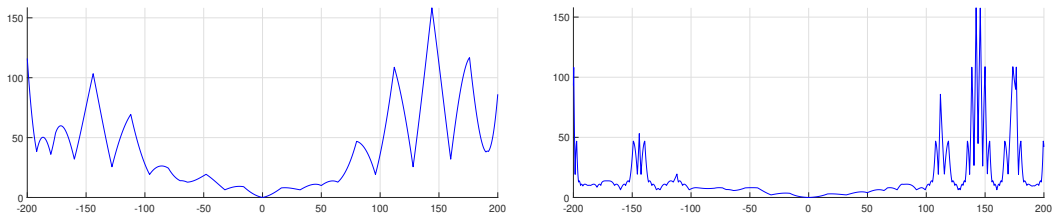
$f(x)$ and $f(f(x))$ for $k = 2^{-1}$



$f(x)$ and $f(f(x))$ for $k = 2^{-2}$



$f(x)$ and $f(f(x))$ for $k = 2^{-3}$



$f(x)$ and $f(f(x))$ for $k = 2^{-4}$

Fig. 3: Functions $f(x)$ (left) and $f(f(x))$ (right) for different values of parameter k

.

less than or equal to 1e–8, it was set as zero. The particular values of $\mathbf{M}$ and $\mathbf{s}$ can be found in the authors github [1].

For both IMODE and AGSK, we use the same parameter settings that they used in the CEC'20 competition [13]. The algorithms were run in a MATLAB R2020b, on a PC with 3.2 GHz Core I5 processor, 16 GB RAM, and Windows 10.

[1]https://github.com/JakubKudela89/Zigzag

B. Results

The results of the computational experiments with the three algorithms are summarized in Table I for the benchmark function $F_1(\mathbf{x})$, and Table II for the benchmark function $F_2(\mathbf{x})$. In both tables, we report the best value over the 30 independent runs (min), the median value, the mean value, the worst value (max), and the standard deviation (std). The best result of the three algorithms in the categories min, median,

mean, and max is highlighted for each problem instance.

Firstly, it is clear from the results that both the benchmark functions are not "impossible" to optimize, as there were plenty of instances where the algorithms found the optimal solution. However, the instances are not "too easy" so that the algorithms find the optimum reliably – this, in our opinion, makes these benchmark functions worth investigating. Another observation to be made is that for both test functions, reducing the zigzag parameter k really reduces the complexity of the problems. In particular, the changes from $k = 2^{-2}$ to $k = 2^{-3}$ and then to $k = 2^{-4}$ seem to have the biggest impact, while the statistical results for problems with $k = [2^0, 2^{-1}, 2^{-2}]$ are relatively stable (for the same dimension D). Unsurprisingly, the difficulty of the test problems also increases with the dimension D.

The most surprising results come from the comparison of the three algorithms. Let us first focus on the first benchmark function $F_1(\mathbf{x})$. In dimension $D = 5$, the most successful algorithm was the PSO, with both AGSK and IMODE being weaker (without any noticeable difference between them) for all instances but the most simple one with $k = 2^{-4}$. For $D = 10$, the situation is a bit different – while PSO again dominated the difficult instances $k = [2^0, 2^{-1}, 2^{-2}, 2^{-3}]$, it was IMODE that was best for the instance $k = 2^{-4}$, but AGSK performed better that IMODE on the instances $k = [2^0, 2^{-1}, 2^{-2}, 2^{-3}]$. For $D = 15$, AGSK is the worst of the three on all instances, with PSO dominating for $k = [2^0, 2^{-1}]$, and IMODE dominating the rest. The results for the largest instances with $D = 20$ have PSO being the best algorithm for all but the simplest instance $k = 2^{-4}$, where it is on the same level as IMODE. AGSK and IMODE behave similarly for $k = [2^0, 2^{-1}, 2^{-2}]$ while IMODE is clearly better for $k = [2^{-3}, 2^{-4}]$.

The results for the second benchmark function $F_2(\mathbf{x})$ are somewhat similar. PSO again dominates the difficult instances for $k = [2^0, 2^{-1}, 2^{-2}, 2^{-3}]$ for all dimension but for $D = 15$, where IMODE seems to work a bit better. On the simplest instance $k = 2^{-4}$, IMODE is best in dimensions $D = [15, 20]$, with $D = 10$ having AGSK as the winner, and $D = 5$ being solved perfectly by all three algorithms. AGSK is the weakest algorithm of the three in all dimensions, apart from $D = 10$.

Overall, on both of the newly proposed benchmark functions, neither of the two best algorithms from the CEC'20 competition performed significantly better than a standard PSO. We would argue that this a prime reason for investigating these benchmark functions even further and for including them in future competitions and benchmark suits.

IV. Conclusion

In this paper, we presented two novel zigzag-based benchmark functions for bound constrained single objective optimization, which have a simple build-in parameter that can be used to increase their complexity. The construction of these functions is straightforward enough to allow for a wide range of variations, extension, and further study. We also used the two best algorithms from the CEC'20 competition and a standard PSO for computational experiments on test instances utilizing the newly proposed benchmark functions. The results of the experiments suggest that the new benchmark functions are well suited for algorithmic comparison. Future research will encompass comparing a wider selection of algorithms, and developing multimodal benchmark functions [14] using the presented technique.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic project No. CZ.02.1.01/0.0/0.0/16_026/0008392 "Computer Simulations for Effective Low-Emission Energy" and by IGA BUT: FSI-S-20-6538.

References

[1] M. H. Hans and G. Beyer, *Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – A critical review*, Swarm and Evolutionary Computation, vol. 44, pp. 927–944, 2019.

[2] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs, *Analyzing the BBOB results by means of benchmarking concepts*, Evolutionary Computation, vol. 23, no. 1, pp. 161–185, 2015.

[3] R. L. Rardin and R. Uzsoy, *Experimental evaluation of heuristic optimization algorithms: A tutorial*, Journal of Heuristics, vol. 7, no. 3, pp. 261–304, 2001.

[4] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad, and P. P. Biswas, *Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization*, Tech. Rep., Zhengzhou University and Nanyang Technological University, 2019.

[5] T. Weise and Z. Wu, *Difficult Features of Combinatorial Optimization Problems and the Tunable W-Model Benchmark Problem for Simulating them*, in GECCO '18: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1769–1776, 2018.

[6] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockho, *COCO: a platform for comparing continuous optimizers in a black-box setting*, Optimization Methods and Software, vol. 36, pp. 114–144, 2021.

[7] K. R. Opara, A. A. Hadi, and A. W. Mohamed, *Parametrized Benchmarking: an outline of the idea and a feasibility study*, in GECCO '20: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, pp. 197–198, 2020.

[8] P. Bujok and R. Polakova, *Eigenvector Crossover in the Efficient jSO Algorithm,* MENDEL, vol. 25, no. 1, pp. 65–72, 2019.

[9] J. Kennedy and R. Eberhart, *Particle swarm optimization*, in Proceedings of ICNN'95 - International Conference on Neural Networks, pp. 1942–1948, 1995.

[10] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, *Improved Multi-operator Differential Evolution Algorithm for Solving Unconstrained Problems*, in 2020 IEEE Congress on Evolutionary Computation (CEC), article no. 19931315, 2020.

[11] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, and N. H. Awad, *Evaluating the Performance of Adaptive GainingSharing Knowledge Based Algorithm on CEC 2020 Benchmark Problems*, in 2020 IEEE Congress on Evolutionary Computation (CEC), article no. 19931514, 2020.

[12] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, *Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm*, International Journal of Machine Learning and Cybernetics, vol. 11, pp. 1501–1529, 2020.

[13] A. Kazikova, M. Pluhacek, and R. Senkerik, *Why Tuning the Control Parameters of Metaheuristic Algorithms Is So Important for Fair Comparison?,* MENDEL, vol. 26, no. 2, pp. 9–16, 2020.

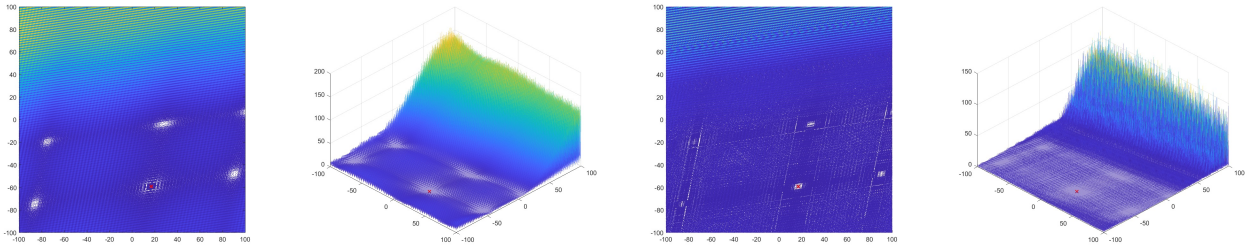
[14] B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, and P. N. Suganthan, *Novel benchmark functions for continuous multimodal optimization with comparative results*, Swarm and Evolutionary Computation, vol. 26, pp. 23–34, 2016.

TABLE I: Statistics of the best objective function values after the maximum number of function evaluations of the different algorithms on the first benchmark function $F_1(\mathbf{x})$ for different values of D and k. The algorithm with best value over the 30 independent runs (min), the best median value, the best mean value, and the best worst value (max) for the particular instance and is emphasized in bold.

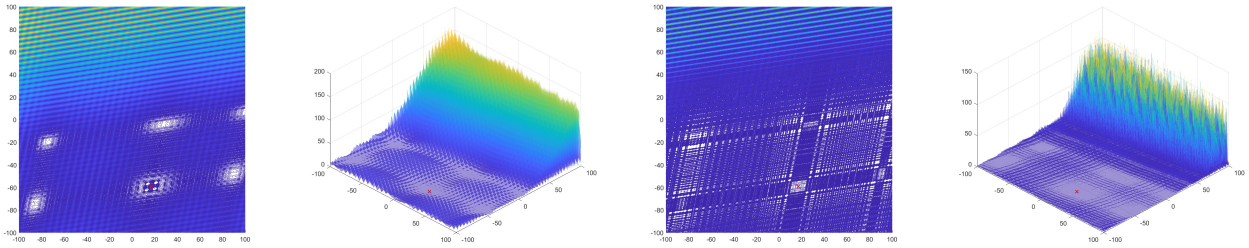
		D = 5			D = 10			D = 15			D = 20		
		PSO	AGSK	IMODE	PSO	AGSK	IMODE	PSO	AGSK	IMODE	PSO	AGSK	IMODE
$k = 2^0$	min	**0**	**0**	9.37e–06	1.73e–04	**0**	1.12e+00	**2.80e–01**	2.53e+00	2.70e+00	**1.01e+00**	5.23e+00	6.06e+00
	median	**0**	**0**	1.76e–02	**1.64e–01**	5.22e–01	1.91e+00	**1.91e+00**	5.06e+00	3.90e+00	**2.87e+00**	7.67e+00	8.44e+00
	mean	**0**	1.87e–02	6.98e–02	**4.08e–01**	7.61e–01	1.93e+00	**1.92e+00**	4.89e+00	3.86e+00	**3.18e+00**	7.67e+00	8.51e+00
	max	**0**	2.77e–01	4.08e–01	2.42e+00	**2.37e+00**	2.73e+00	**4.33e+00**	6.31e+00	5.24e+00	**6.94e+00**	9.72e+00	1.03e+01
	std	0	5.47e–02	7.46e–01	6.57e–01	7.48e–01	4.71e–01	1.12e+00	9.45e–01	5.92e–01	1.60e+00	1.15e+00	9.07e–01
$k = 2^{-1}$	min	**0**	**0**	6.75e–08	9.22e–06	**0**	3.68e–01	8.81e–01	**0**	2.64e+00	**1.12e+00**	5.52e+00	7.27e+00
	median	**0**	**0**	8.84e–04	**4.45e–01**	9.04e–01	2.34e+00	**3.38e+00**	4.90e+00	4.02e+00	**3.39e+00**	8.76e+00	8.85e+00
	mean	**0**	6.89e–02	4.23e–02	**6.80e–01**	1.02e+00	2.25e+00	**3.49e+00**	4.80e+00	3.99e+00	**3.60e+00**	8.67e+00	8.88e+00
	max	**2.57e–08**	1.00e+00	2.67e–01	**1.91e+00**	2.74e+00	3.19e+00	6.58e+00	6.11e+00	**5.10e+00**	**7.61e+00**	1.03e+01	1.05e+01
	std	0	2.00e–01	7.98e–02	6.34e–01	7.84e–01	6.64e–01	1.64e+00	1.14e+00	5.72e–01	1.38e+00	1.13e+00	7.54e–01
$k = 2^{-2}$	min	**0**	**0**	**0**	2.14e–03	**0**	1.31e+00	9.60e–01	3.01e+00	**5.40e–01**	**8.02e–01**	6.65e+00	4.25e+00
	median	**0**	1.10e–07	1.34e–04	**6.44e–01**	1.67e+00	2.13e+00	3.47e+00	5.01e+00	**3.16e+00**	**3.37e+00**	8.49e+00	8.08e+00
	mean	3.73e–02	3.25e–02	**2.99e–02**	**6.24e–01**	1.59e+00	2.12e+00	3.70e+00	4.82e+00	**3.04e+00**	**3.53e+00**	8.48e+00	7.91e+00
	max	8.02e–01	**2.87e–01**	5.22e–01	**1.61e+00**	2.99e+00	3.65e+00	7.54e+00	5.72e+00	**4.06e+00**	**7.86e+00**	1.01e+01	9.61e+00
	std	1.55e–01	7.68e–02	1.13e–01	3.26e–01	7.84e–01	5.39e–01	1.86e+00	7.13e–01	7.79e–01	1.30e+00	9.56e–01	1.09e+00
$k = 2^{-3}$	min	**0**	**0**	**0**	1.48e–05	**0**	8.41e–04	**0**	2.94e+00	8.67e–04	**0**	5.96e+00	2.74e+00
	median	**0**	**0**	**0**	**3.79e–02**	3.20e–01	1.55e+00	3.85e+00	4.65e+00	**9.60e–01**	**3.70e+00**	7.91e+00	5.83e+00
	mean	**0**	2.79e–04	1.52e–06	**4.21e–01**	8.00e–01	1.41e+00	3.75e+00	4.58e+00	**8.23e–01**	**3.64e+00**	8.09e+00	5.78e+00
	max	**0**	8.14e–03	2.22e–05	**1.92e+00**	2.74e+00	2.36e+00	8.06e+00	5.71e+00	**1.65e+00**	**7.17e+00**	1.05e+01	7.53e+00
	std	0	1.48e–03	5.48e–06	5.51e–01	9.79e–01	6.59e–01	1.92e+00	6.86e–01	5.20e–01	1.71e+00	1.04e+00	1.28e+00
$k = 2^{-4}$	min	**0**	**0**	**0**	3.97e–06	**0**	**0**	**0**	**0**	**0**	**0**	3.83e+00	1.28e+00
	median	**0**	**0**	**0**	4.59e–03	**0**	**0**	3.20e+00	3.83e+00	**0**	**3.07e+00**	7.60e+00	3.23e+00
	mean	**0**	**0**	**0**	2.00e–01	8.53e–02	**3.49e–07**	3.41e+00	3.62e+00	**0**	3.21e+00	7.37e+00	**2.96e+00**
	max	**0**	**0**	**0**	1.28e+00	1.28e+00	**7.75e–06**	8.88e+00	4.99e+00	**0**	6.12e+00	9.06e+00	**3.97e+00**
	std	0	0	0	4.11e–01	3.24e–01	1.48e–06	2.11e+00	1.19e+00	0	1.60e+00	1.17e+00	7.34e–01

TABLE II: Statistics of the best objective function values after the maximum number of function evaluations of the different algorithms on the first benchmark function $F_2(\mathbf{x})$ for different values of D and k. The algorithm with best value over the 30 independent runs (min), the best median value, the best mean value, and the best worst value (max) for the particular instance and is emphasized in bold.

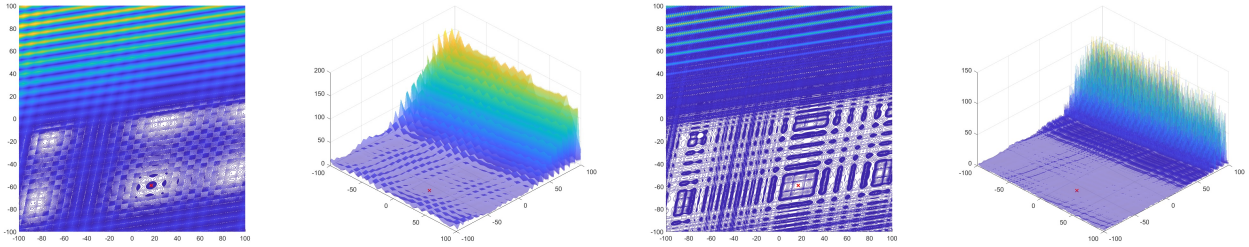
		D = 5			D = 10			D = 15			D = 20		
		PSO	AGSK	IMODE	PSO	AGSK	IMODE	PSO	AGSK	IMODE	PSO	AGSK	IMODE
$k = 2^0$	min	**0**	**0**	4.44e–05	1.70e–05	**0**	1.01e–01	**9.66e–02**	3.64e–01	2.40e–01	**8.69e–02**	6.03e–01	5.26e–01
	median	**0**	6.20e–03	1.20e–02	**1.68e–02**	1.69e–01	2.13e–01	**3.65e–01**	6.85e–01	4.15e–01	**4.75e–01**	9.09e–01	8.01e–01
	mean	**1.75e–02**	2.24e–02	1.84e–02	**4.03e–02**	1.61e–01	2.03e–01	4.91e–01	6.47e–01	**4.00e–01**	**4.99e–01**	8.89e–01	7.81e–01
	max	1.30e–01	1.25e–01	**5.18e–02**	**1.92e–01**	2.87e–01	2.61e–01	1.56e+00	8.40e–01	**4.94e–01**	2.23e+00	1.08e+00	**9.30e–01**
	std	2.95e–02	3.19e–02	1.72e–02	4.95e–02	7.54e–02	4.38e–02	4.34e–01	1.12e–01	6.23e–02	3.87e–01	1.24e–01	9.74e–02
$k = 2^{-1}$	min	**0**	**0**	7.68e–06	3.68e–07	**0**	1.31e–01	**1.11e–02**	3.73e–01	2.90e–01	**5.30e–02**	4.25e–01	6.77e–01
	median	**0**	9.77e–08	1.91e–03	**7.61e–03**	1.22e–01	2.39e–01	**3.16e–01**	6.73e–01	5.40e–01	**3.42e–01**	9.89e–01	1.18e+00
	mean	**0**	3.74e–03	5.26e–03	**2.32e–02**	1.29e–01	2.52e–01	**3.41e–01**	6.44e–01	5.32e–01	**3.81e–01**	9.98e–01	1.16e+00
	max	**0**	3.67e–02	2.81e–02	**1.23e–01**	3.67e–01	4.12e–01	8.91e–01	8.59e–01	**6.68e–01**	**9.25e–01**	1.32e+00	1.37e+00
	std	0	9.20e–03	6.91e–03	3.68e–02	1.08e–01	7.34e–02	2.24e–01	1.21e–01	9.66e–02	2.33e–01	2.14e–01	1.39e–01
$k = 2^{-2}$	min	**0**	**0**	1.32e–08	8.14e–06	**0**	4.56e–02	**1.06e–01**	1.38e–01	1.57e–01	**2.05e–01**	2.69e–01	5.38e–01
	median	**0**	**0**	1.17e–05	**6.24e–03**	1.29e–01	2.12e–01	3.68e–01	4.74e–01	**3.11e–01**	**4.06e–01**	9.12e–01	8.98e–01
	mean	**1.83e–04**	2.44e–03	1.99e–03	**2.82e–02**	1.27e–01	1.96e–01	4.00e–01	4.57e–01	**3.20e–01**	**4.27e–01**	8.84e–01	8.82e–01
	max	**5.49e–03**	2.35e–02	2.71e–02	**2.00e–01**	3.62e–01	3.38e–01	9.49e–01	6.54e–01	**4.37e–01**	**7.04e–01**	1.25e+00	1.10e+00
	std	1.00e–03	5.73e–03	5.61e–03	4.80e–02	8.63e–02	7.96e–02	2.18e–01	1.23e–01	6.62e–02	1.44e–01	1.91e–01	1.33e–01
$k = 2^{-3}$	min	**0**	**0**	**0**	2.67e–08	**0**	4.74e–02	**0**	1.12e–01	1.45e–02	**7.57e–02**	2.93e–01	2.98e–01
	median	**0**	**0**	3.32e–08	**1.38e–04**	7.16e–02	8.30e–02	3.02e–01	2.60e–01	**1.04e–01**	**2.53e–01**	5.87e–01	3.93e–01
	mean	1.08e–03	1.04e–03	**4.02e–06**	**2.84e–02**	7.51e–02	9.08e–02	2.76e–01	2.54e–01	**9.97e–02**	**2.77e–01**	5.52e–01	3.96e–01
	max	3.26e–02	1.42e–02	**4.90e–05**	**1.27e–01**	1.59e–01	1.72e–01	5.15e–01	3.32e–01	**1.63e–01**	**4.94e–01**	7.02e–01	5.05e–01
	std	5.95e–03	3.35e–03	1.07e–05	4.18e–02	5.08e–02	3.00e–02	1.30e–01	5.24e–02	3.69e–02	1.21e–01	9.42e–02	4.94e–02
$k = 2^{-4}$	min	**0**	**0**	**0**	6.19e–08	**0**	**0**	**0**	**0**	**0**	2.19e–02	1.12e–01	**9.46e–05**
	median	**0**	**0**	**0**	4.38e–05	**0**	**0**	8.40e–02	1.53e–01	**0**	1.97e–01	3.05e–01	**1.54e–01**
	mean	**0**	**0**	**0**	2.97e–03	**0**	1.00e–06	1.22e–01	1.47e–01	**0**	2.01e–01	2.91e–01	**1.52e–01**
	max	**0**	**0**	**0**	2.19e–02	**0**	1.12e–05	3.91e–01	2.28e–01	**0**	3.70e–01	3.85e–01	**2.16e–01**
	std	0	0	0	7.57e–03	0	2.68e–06	1.16e–01	4.86e–02	0	7.70e–02	6.51e–02	4.33e–02



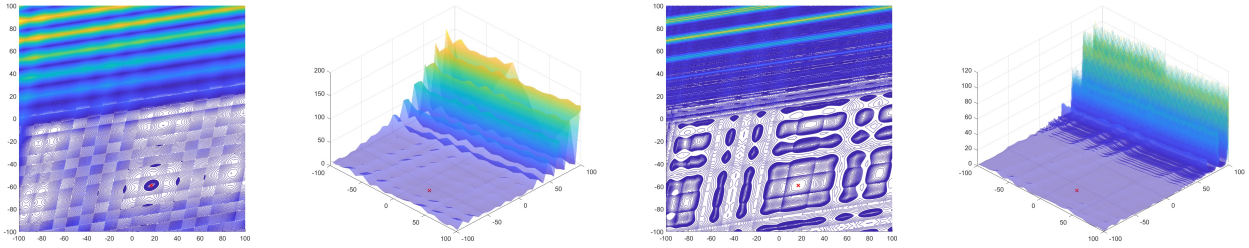
$F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ for $k = 2^0$



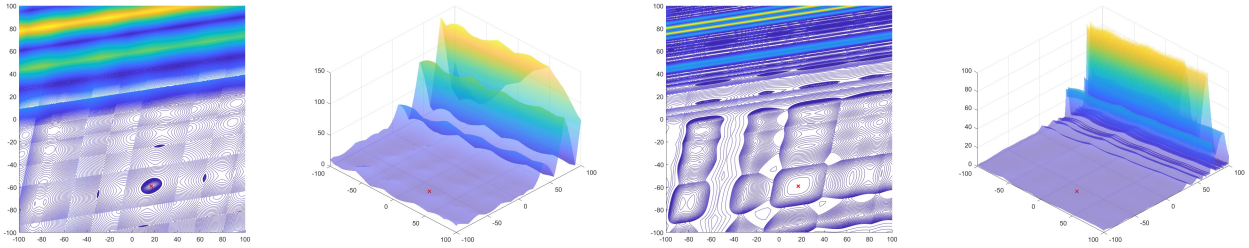
$F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ for $k = 2^{-1}$



$F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ for $k = 2^{-2}$



$F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ for $k = 2^{-3}$



$F_1(\mathbf{x})$ and $F_2(\mathbf{x})$ for $k = 2^{-4}$

Fig. 4: Contour and surface plots of the benchmark functions $F_1(\mathbf{x})$ (left) and $F_2(\mathbf{x})$ (right) for different values of parameter k. The optimum is highlighted by a red marker.

A3

nature machine intelligence

Article

https://doi.org/10.1038/s42256-022-00579-0

A critical problem in benchmarking and analysis of evolutionary computation methods

Received: 8 February 2022

Accepted: 1 November 2022

Published online: 12 December 2022

Check for updates

Jakub Kudela

Benchmarking is a cornerstone in the analysis and development of computational methods, especially in the field of evolutionary computation, where theoretical analysis of the algorithms is almost impossible. In this Article, we show that some of the frequently used benchmark functions have their respective optima in the centre of the feasible set and that this poses a critical problem for the analysis of evolutionary computation methods. We carry out an analysis of seven recently published methods and find that these contain a centre-bias operator that lets them find optima in the centre of the benchmark set with ease. However, this mechanism makes their comparison with other methods (that do not have a centre-bias) meaningless. We compare the computational performance of these seven new methods to two long-standing ones in evolutionary computation ('differential evolution' and 'particle swarm optimization') on shifted problems and on more advanced benchmark problems. Only one of the seven methods performed consistently better than the pair of old methods, three performed on par, two performed very badly and the worst one performed barely better than a random search. We provide several suggestions that could help to improve analysis and benchmarking in evolutionary computation.

Inspired by natural behaviours, the field of evolutionary computation (EC) has produced a multitude of pivotal metaheuristic algorithms over its rich history. Among them are such classical methods as genetic algorithms, evolutionary strategy, differential evolution and particle swarm optimization. These methods found their places in various complex applications where the use of exact algorithms was inadequate or too computationally expensive. However, over the past few years, there has been an explosion of 'novel' methods that draw on natural principles. The bestiary of EC[1], a catalogue of nature-based algorithms, already lists over 250 algorithms that claim to be inspired by natural processes. After a multitude of these methods were found to hide a lack of novelty behind metaphor-rich jargon[2–6], a call was made from within the EC community[7]. In the letter, the collective of authors and signatories identified four main issues with the high-volume inflow of new methods: useless metaphors, lack of novelty, poor experimental validation and comparison, and publishing these methods in off-topic journals. In this text, we will show that there is another serious problem with the experimentation and comparison of different methods.

As metaheuristics are generally difficult to analyse analytically, most of the reasoning about their viability is done through benchmarking[8]. If a method performs well on a generally accepted set of problems, it has a high chance of being seen as valid. Over the years, many different benchmark functions and sets have been proposed in journal articles[9], but the most popular ones have been constructed for special sessions (competitions) on black-box optimization at two conferences: the Institute of Electrical and Electronics Engineers (IEEE) Congress on Evolutionary Computation (CEC), and the Genetic and Evolutionary Computation Conference (GECCO), where the Black-Box Optimization Benchmarking (BBOB) workshop is held. The BBOB functions are a part of the COCO platform for comparing optimization

Institute of Automation and Computer Science, Brno University of Technology, Brno, Czech Republic. e-mail: Jakub.Kudela@vutbr.cz

https://doi.org/10.1038/s42256-022-00579-0

Table 1 | The often-used benchmark functions, dimension 30

ID	Name	Function	Type	Range	f^*	$f(0)$	x^*
F01	Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	U, S	[−100, 100]	0		
F02	Schwefel 2.22	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	U, N	[−100, 100]	0	0	[0, 0, ...]
F03	Schwefel 1.2	$f(x)=\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j^2\right)$	U, N	[−100, 100]	0	0	[0, 0, ...]
F04	Schwefel 2.21	$f(x)=\max_i \lvert x_i\rvert, 1\le i\le n$	U, S	[−100, 100]	0	0	[0, 0, ...]
F05	Rosenbrock	$f(x)=\sum_{i=1}^{n-1} \left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	U, N	[−30, 30]	0	29.0	[1, 1, ...]
F06	Step	$f(x)=\sum_{i=1}^{n} (x_i+0.5)^2$	U, S	[−100, 100]	0	7.50	[−0.5, −0.5, ...]
F07	Quartic with noise	$f(x)=\sum_{i=1}^{n} ix_i^4+\mathrm{rand}(0,1)$	U, S	[−1.28, 1.28]	0	0	[0, 0, ...]
F08	Schwefel 2.26	$f(x)=\sum_{i=1}^{n} -x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	M, S	[−500, 500]	-1.25×10^4	0	[420.9, 420.9, ...]
F09	Rastrigin	$f(x)=\sum_{i=1}^{n} \left[x_i^2-10\cos(2\pi x_i)+10\right]$	M, S	[−5.12, 5.12]	0	0	[0, 0, ...]
F10	Ackley	$f(x)=-20\exp\left(-0.2\left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right)^{0.5}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	M, N	[−32, 32]	0	0	[0, 0, ...]
F11	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	M, N	[−600, 600]	0	0	[0, 0, ...]
F12	Penalized1	$f(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)\}+\sum_{i=1}^{n} u(x_i,10,100,4), y_i=1+\frac{x_i+1}{4}$, $u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a<x_i<a \\ k(-x_i-a)^m & x_i<a \end{cases}$	M, N	[−50, 50]	0	1.67	[−1, −1, ...]
F13	Penalized2	$f(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right]+(x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	M, S	[−50, 50]	0	3.00	[1, 1, ...]

U, unimodal; M, multimodal; S, separable; N, nonseparable; f^*, the optimal function value; $f(0)$, function value at the zero vector; x^*, optimal solution.

algorithms[10], while the benchmarks from the CEC competitions (which started in 2005 and continue to this day) can be found on the github of one of the authors[11]. As was shown previously[12], the characteristics of the functions used in these two benchmarks are very different. The CEC benchmarks are composed of similar subfunctions, possibly giving an advantage to algorithms that perform well on these fewer subfunctions. It was also found that the CEC functions share more similarities among themselves than with those found in the BBOB[12]. A tremendously useful feature of both the repository of the CEC competitions and the COCO platform is their inclusion of algorithms and data. In the repository of the CEC competitions, the best-performing algorithms are shared along with their results. On the COCO platform, the results of different methods can be found. However, only a handful of the source codes for the methods are included, and some of the links to the repositories of the methods are no longer working[13]. Some authors also voiced their critique of the artificial nature of these benchmark sets[14], and instead advocated for testing optimization algorithms on real-world problems[15].

Another standard benchmark set that is extensively used for comparing optimization methods consists of some of the most well-known functions such as Ackley, Griewank, Rosenbrock, Rastrigin, or Schwefel, but it also contains a small design flaw. A large portion of the functions found in this set have the optimum at a zero vector (or in the centre of the feasible set). This fact, on its own, does not pose a serious problem, but comparing methods that incorporate a "check-the-middle" procedure or a centre-bias (or zero-bias) operator on these benchmarks to other methods is essentially meaningless. This issue was also identified during the CEC 2021 competition[16].

Table 2 | Mean error over 30 independent runs on the unshifted benchmark functions

ID	HHO	SMA	GBO	BOA	MPA	KMA	STOA	PSO	DE	LSHADE	HSES
F01	2.79×10^{-15}	0	5.43×10^{-267}	3.31×10^{-9}	4.39×10^{-61}	0	9.32×10^{-19}	0.186	3.82×10^{-2}	1.28×10^{-12}	1.48×10^{-10}
F02	1.71×10^{-8}	5.06×10^{-234}	2.25×10^{-133}	4.91×10^{-10}	1.49×10^{-31}	0	1.04×10^{-12}	3.54	2.79	6.37×10^{-6}	3.05×10^{-3}
F03	2.63×10^{-16}	0	4.10×10^{-216}	3.59×10^{-9}	1.34×10^{-8}	0	1.21×10^{-9}	3.30×10^{4}	1.89×10^{4}	4.13×10^{-4}	4.79×10^{2}
F04	8.42×10^{-13}	1.61×10^{-227}	4.42×10^{-119}	1.37×10^{-6}	5.89×10^{-23}	0	3.92×10^{-6}	38.4	13.5	5.08×10^{-3}	3.82×10^{-2}
F05	2.06×10^{-3}	0.371	39.2	48.8	44.3	25.7	27.9	2.92×10^{4}	6.09×10^{2}	20.7	27.7
F06	1.08×10^{-5}	2.60×10^{-3}	6.36×10^{-8}	**7.75**	8.64×10^{-8}	0	2.08	89.8	6.68	9.76×10^{-13}	3.38×10^{-10}
F07	1.03×10^{-4}	7.49×10^{-5}	2.81×10^{-4}	4.16×10^{-3}	4.30×10^{-4}	8.67×10^{-5}	1.30×10^{-3}	0.408	8.56×10^{-2}	3.19×10^{-3}	1.58×10^{-2}
F08	0.111	0.222	**6.17×10^{3}**	**1.53×10^{4}**	**5.93×10^{3}**	**3.55×10^{3}**	**6.89×10^{3}**	3.25×10^{3}	1.43×10^{4}	1.64×10^{2}	**7.25×10^{3}**
F09	0	0	0	83.8	0	0	1.16	1.80×10^{2}	3.98×10^{2}	10.8	33.9
F10	2.10×10^{-11}	8.88×10^{-16}	8.88×10^{-16}	9.66×10^{-7}	8.88×10^{-16}	8.88×10^{-16}	**19.3**	3.68	1.53	2.38×10^{-7}	7.45×10^{-4}
F11	7.40×10^{-15}	0	0	7.47×10^{-10}	0	0	8.60×10^{-3}	1.87	1.07	6.89×10^{-12}	3.79×10^{-10}
F12	4.15×10^{-7}	1.05×10^{-3}	2.77×10^{-9}	0.477	3.41×10^{-5}	1.01×10^{-6}	0.136	61.0	1.13	5.36×10^{-14}	3.32×10^{-10}
F13	7.34×10^{-6}	1.45×10^{-3}	2.01×10^{-2}	4.86	0.210	2.21×10^{-3}	1.66	1.06×10^{4}	6.36	1.06×10^{-12}	3.19×10^{-7}

Bold indicates results that are worse than PSO or DE.

The inclusion of the centre-bias operator is hardly ever plainly visible from the description of the method. This is both a problem of the often-used metaphor-rich jargon and the difficulty of the theoretical analysis of stochastic systems. Possible mechanisms leading to a centre-bias operator in metaheuristics are the contraction operator that was found in the Grey Wolf Optimization algorithm[17] and a search bias that was found in the Salp Swarm Optimization algorithm[18]. The centre-bias was also found in the Sooty Tern Optimization Algorithm and the Tunicate Swarm Algorithm[19].

In the remainder of this text, we will take a closer look at seven recently published methods in journals such as *Applied Soft Computing*, *Information Sciences* and *Future Generation Computer Systems*. The methods in question are the Komodo Mlipir Algorithm (KMA)[20], the Slime Mould Algorithm (SMA)[21], the Butterfly Optimization Algorithm (BOA)[22], the Gradient-Based Optimizer (GBO)[23], the enhanced Marine Predators Algorithm (MPA)[24], the Harris Hawks Optimization (HHO)[25] and the Sooty Tern Optimization Algorithm (STOA)[26]. We will show that all of the abovementioned methods contain the centre-bias operator. We will try to carry out a fair comparison between these relatively new methods and two older ones—Differential Evolution (DE) and Particle Swarm Optimization (PSO). Additionally, we include in the computations two methods that performed well in the CEC competitions, namely LSHADE[27] (which was the basis of many of the best-performing methods in the past CEC competitions) and HSES[28] (winner of the CEC 2018 competition). For all the methods, we chose to use the parameter setting recommended in the corresponding publications, without any parameter tuning[29]. The code for all the experiments can be found at https://doi.org/10.24433/CO.1268126.v1 (ref. [30]).

Results

Problematic benchmark problems

Out of the seven abovementioned methods, six performed the computational comparison on a very similar benchmark set, with thirteen of the considered problems summarized in Table 1. The only one that did not use these functions was the GBO (it still used functions that exhibit the same issue). Except for GBO and BOA, these were the first thirteen problems in the considered benchmark sets. Of these thirteen problems, eight have their respective optimum at the zero vector. Four other problems have the optimum quite close to the zero vector and give very good results when evaluated at the zero vector. Only a single problem (F08) has the optimum located far from zero.

Behaviour without a shift

We use a previously devised methodology[19] to analyse the behaviour of the methods. First, the different methods were evaluated on the thirteen benchmark functions without any additional modifications. The dimension of the problems was set to 30 and the maximum number of allowed function evaluations was set to 50,000. As the performance measure, we chose the mean error (difference between optimal function value and best function value found after the max iteration count) over 30 independent runs. The results are reported in Table 2 (in the format produced by the respective methods). Although there is no practical meaning in numbers like 2.25×10^{-133}, these were the numbers reported by the respective methods (and the numbers on which statistical analyses were performed in the respective publications). The more interesting thing to note is that all the methods in question (apart from PSO and DE) perform extremely well on the problems that have optimum at the zero vector (F01–F04, F07, F09–F11). The only exception is STOA, which performed badly on F10. Additionally, on the four problems that produce good values when evaluated at the zero vector (F05, F06, F12, F13) the methods perform very well—all but BOA outperform the old DE and PSO methods. On F08, however, the performance of these methods was poor, and GBO, BOA, MPA, KMA, and STOA are all worse than PSO. Interestingly, HSES is also worse than PSO on F08—this might be explained by the fact that the CEC competitions allow for much more function evaluations (which is what HSES was designed and tuned for).

Behaviour with a shift

Now, we examine the effects of modifying these benchmark problems. Following previous work[19], we introduce a shift operator, which moves the evaluated point by a predetermined vector s, that is, instead of the benchmark function being $f(x)$, it is modified to $f(x+s)$. Since the functions stay basically the same, one would expect the algorithms to perform consistently. The shift vector was chosen as 10% of the range in each dimension—for example, for F01, $s=[20, 20, \ldots]$. We ran the considered methods again 30 times (this time, any result smaller than 1×10^{-8} was considered as zero). The results (mean error) are shown in Table 3. This time, only one of the newly proposed methods (disregarding LSHADE and HSES) was able to consistently find the optimum, and only for a single problem. It was GBO for F01, which is not very surprising, as GBO uses Newton's method in each step and the F01 problem is a quadratic function (GBO also performed very well on F06, which is almost a quadratic function). Besides this single instance, the supreme

Table 3 | Mean error over 30 independent runs on the shifted benchmark functions

ID	HHO	SMA	GBO	BOA	MPA	KMA	STOA	PSO	DE	LSHADE	HSES
F01	2.80×10^{-3}	9.36×10^{-3}	0	**5.10×10^{3}**	2.24×10^{-6}	1.42×10^{-2}	**3.48×10^{3}**	0.246	2.61×10^{-2}	0	0
F02	4.24×10^{-2}	0.279	3.01×10^{-2}	**35.5**	0.633	**3.62**	**38.2**	2.77	3.06	9.55×10^{-6}	3.23×10^{-3}
F03	4.19	23.4	9.43×10^{2}	**9.08×10^{4}**	9.57×10^{2}	6.23×10^{2}	7.32×10^{3}	3.09×10^{4}	1.64×10^{4}	4.16×10^{-4}	4.13×10^{2}
F04	7.28×10^{-3}	7.37×10^{-2}	**18.5**	**20.0**	11.2	0.333	**19.9**	38.3	13.1	3.75×10^{-3}	3.84×10^{-2}
F05	2.44×10^{-2}	1.09	1.07×10^{2}	**2.95×10^{6}**	1.57×10^{2}	17.2	**5.88×10^{5}**	2.89×10^{4}	5.88×10^{2}	21.2	29.4
F06	4.76×10^{-3}	3.87×10^{-2}	3.96×10^{-5}	**1.24×10^{4}**	1.04×10^{-4}	2.05	**3.93×10^{3}**	72.1	7.10	0	1.19×10^{-7}
F07	5.77×10^{-4}	7.91×10^{-3}	**8.35×10^{-2}**	**4.03**	**0.236**	2.62×10^{-3}	**0.867**	0.382	8.18×10^{-2}	3.33×10^{-3}	1.64×10^{-2}
F08	7.92×10^{-2}	0.122	**6.09×10^{3}**	**1.50×10^{4}**	**6.43×10^{3}**	1.19×10^{3}	**7.55×10^{3}**	3.89×10^{3}	1.50×10^{4}	1.80×10^{2}	**4.41×10^{3}**
F09	1.28×10^{-2}	3.57×10^{-2}	48.4	1.15×10^{2}	48.6	13.9	33.1	1.59×10^{2}	3.98×10^{2}	11.0	44.4
F10	1.17×10^{-2}	8.20×10^{-2}	**4.00**	**14.7**	**2.74**	0.711	**12.6**	3.39	1.44	2.57×10^{-7}	7.55×10^{-4}
F11	5.03×10^{-3}	0.279	1.14×10^{-2}	**1.10×10^{2}**	5.93×10^{-3}	6.17×10^{-2}	**32.6**	1.82	1.06	0	0
F12	1.13×10^{-5}	6.94×10^{-4}	**5.68**	**1.31×10^{2}**	**2.50**	5.17×10^{-5}	**57.4**	4.19×10^{2}	1.27	0	0
F13	1.39×10^{-4}	8.78×10^{-3}	**13.7**	**1.97×10^{6}**	0.387	1.28×10^{-2}	**2.47×10^{5}**	4.83×10^{3}	5.94	0	4.11×10^{-7}

Results in bold are those that are worse than PSO or DE.

performance of the novel methods is gone, while the performance of PSO, DE, LSHADE and HSES stay roughly the same. Even more importantly, it seems that GBO, MPA, and in particular both BOA and STOA offer only a slight (or no) advantage over the pair of two-decades-old methods. Although drastically corrected, the results for HHO, KMA and SMA suggest that they may contain some computational advancements (even against LSHADE or HSES on problems such as F05, F07-F09). Or perhaps they were tuned to perform well on these benchmark problems[24]. We also performed the same analysis for adding both shift and rotation, which had the same conclusion (the results can be found in Supplement Table 1).

Comparison on a collection of real-world problems

As many authors emphasize the need to test the algorithms on real-world applications, the next comparison considers benchmark functions based on these types of problems. The main issue with the real-world problems is that they are rarely part of any benchmark sets, with two notable exceptions. The first one is the CEC 2011 competition benchmark set, which is unfortunately still rarely used in practice[14]. The second one consists of 13 engineering problems[31] in relatively low dimensions (between 2 and 11), which can be found in Table 4.

Except for KMA, the other six considered methods compared their performance with other algorithms on at least three of real-world problems (in their respective publications). The most popular problems were P12 (in five of the seven considered methods), and P2 and P3 (both in four). Even though the authors of the six methods that used the real-world problems shared the source code of the methods (and the abovementioned problematic benchmarks), they did not include the implementation used to optimize the real-world problems. This is a rather important issue, as all considered real-world problems have constraints and there are different techniques that can be used for their incorporation into the objective. Another problem is the selection of algorithms used for comparisons. Although most of the considered methods were compared against PSO or DE in the synthetic benchmarks, only MPA and STOA also included them in the comparison on the real-world problems.

For the computational experiments with these real-world problems, we used a previously published implementation[31]. We also chose to incorporate the constraints into the objective by a simple linear penalization[31]. The maximum number of function evaluations was set to 50,000 and each method was run 30 times. The results of these computations (mean objective function value) are summarized in Table 4. The most striking thing to note is that all these real-world problems can be consistently solved by DE. The second important observation is that although HHO and SMA performed well on the shifted benchmarks, this did not translate to the set of real-world problems. Conversely, even though STOA performed very badly on the shifted benchmarks, it achieved relatively good results on the real-world problems (similar to PSO). BOA remained among the worst methods. On the other hand, the results of the best-performing methods (DE, LSHADE, MPA, KMA, and GBO) are extremely similar. Additionally, for 8 of the 13 problems, more than half of the 11 methods achieved the exact same results. This makes this setting of the benchmarking set and performance metric (best result after a given number of function evaluations) not well suited for comparing advanced algorithms.

Comparison on a more advanced benchmark set

To study the performance of the seven studied methods on difficult problems for which they were not tuned, we decided to test them on a more advanced benchmark set and followed a procedure employed previously[19] to carry out the numerical experiments. We chose the ambiguous benchmark set[32] with 32 multimodal, nondifferentiable, nonseparable functions in dimensions 5, 10, 15 and 20 (eight in each dimension). Even though some of the methods were able to optimize thousand-dimensional functions (on the problematic benchmark set), we felt that more 'moderate' dimensions were sufficient for the comparison. The number of available function evaluations were increased to 50,000, 200,000, 500,000 and 1,000,000 for dimensions 5, 10, 15 and 20, respectively. Additionally, we implemented a random search (RS) method that did not perform any optimization and only used the maximum number of function evaluation to randomly sample points and found the best one among them. The methods were run 30 times on each of the 32 problems. Detailed results of this comparison are reported in Supplementary Table 4.

To give a meaningful ranking of the methods, we chose the Glicko-2 rating system[33]. Detailed results of the rating can be found in Supplementary Table 2, while its graphical representation is shown in Fig. 1. We also performed the Friedman rank-sum test, the results of which closely follow the Glicko-2 rating (these can be found in Supplementary Table 3). Only one method, MPA, was ranked significantly higher than either PSO or DE. Three methods got approximately the same rating as PSO—KMA, GBO and SMA. Although they do not seem to bring any advantage over PSO/DE, they performed reasonably well even on difficult problems. The same cannot be said about HHO, STOA and BOA. Although HHO got very good results on the shifted benchmarks from the previous evaluation, it performed quite badly on the

Table 4 | Mean objective value over 30 independent runs on the collection of real-world problem

	Description	Dim	HHO	SMA	GBO	BOA	MPA	KMA	STOA	PSO	DE	LSHADE	HSES
P01	Speed reducer	7	7.540×10^{3}	**2.994×10^{3}**	**2.994×10^{3}**	3.142×10^{3}	**2.994×10^{3}**	**2.994×10^{3}**	3.005×10^{3}	3.028×10^{3}	**2.994×10^{3}**	**2.994×10^{3}**	**2.994×10^{3}**
P02	Tension/ compression spring design	3	1.344×10^{-2}	1.335×10^{-2}	**1.267×10^{-2}**	1.727×10^{-2}	**1.267×10^{-2}**	**1.267×10^{-2}**	1.361×10^{-2}	1.378×10^{-2}	**1.267×10^{-2}**	**1.267×10^{-2}**	1.436×10^{-2}
P03	Pressure vessel design	4	3.672×10^{4}	6.348×10^{3}	**6.248×10^{3}**	2.381×10^{4}	**6.248×10^{3}**	6.256×10^{3}	$6.603E\times10^{3}$	8.331×10^{3}	**6.248×10^{3}**	**6.248×10^{3}**	6.515×10^{3}
P04	Three-bar truss design problem	2	2.645×10^{2}	2.697×10^{2}	**2.639×10^{2}**	2.640×10^{2}	**2.639×10^{2}**	**2.639×10^{2}**	**2.639×10^{2}**	**2.639×10^{2}**	**2.639×10^{2}**	**2.639×10^{2}**	**2.639×10^{2}**
P05	Design of gear train	4	3.161×10^{-8}	5.203×10^{-9}	9.756×10^{-10}	8.031×10^{-5}	2.799×10^{-11}	1.252×10^{-11}	2.155×10^{-9}	2.797×10^{-9}	**2.701×10^{-12}**	1.526×10^{-11}	6.377×10^{-10}
P06	Cantilever beam	5	1.382	**1.340**	**1.340**	4.117	**1.340**	**1.340**	1.352	**1.340**	**1.340**	**1.340**	**1.340**
P07	Minimize I-beam vertical deflection	4	8.798×10^{4}	**1.307×10^{2}**	**1.307×10^{-2}**	1.362×10^{-2}	**1.307×10^{-2}**	**1.307×10^{-2}**	**1.307×10^{-2}**	1.330×10^{-2}	**1.307×10^{-2}**	**1.307×10^{-2}**	1.310×10^{-2}
P08	Tubular column design	2	26.56	**26.49**	**26.49**	26.57	**26.49**	**26.49**	26.50	**26.49**	**26.49**	**26.49**	**26.49**
P09	Piston lever	4	2.432×10^{2}	12.15	17.70	86.72	**1.057**	**1.057**	67.99	30.15	**1.057**	**1.057**	91.36
P10	Corrugated bulkhead design	4	7.012	11.39	**6.843**	7.110	**6.843**	**6.843**	6.930	**6.843**	**6.843**	**6.843**	**6.843**
P11	Car side impact design	11	25.90	23.09	22.98	24.86	22.86	22.89	23.30	23.26	**22.84**	**22.84**	23.06
P12	Design of welded beam design	4	2.794	1.727	**1.725**	2.128	**1.725**	**1.725**	1.839	**1.725**	**1.725**	**1.725**	**1.725**
P13	Reinforced concrete beam design	3	1.655×10^{2}	**1.594×10^{2}**	**1.594×10^{2}**	1.597×10^{2}	**1.594×10^{2}**	**1.594×10^{2}**	**1.594×10^{2}**	**1.594×10^{2}**	**1.594×10^{2}**	**1.594×10^{2}**	1.597×10^{2}

Values in bold are the best-scoring methods on the particular problem.

ambiguous benchmark set. Similarly, STOA performed reasonably well on the real-world benchmarks, but it did not translate to the ambiguous benchmark set. Lastly, BOA had the worst results of the seven new methods, and on some problems was even outperformed by RS. It should be emphasized that even the best-performing algorithm of the methods in question (KMA) was dominated by the two algorithms from the CEC competitions (LSHADE and HSES).

Discussion

Based on the described computational comparisons it is clear that the seven studied methods contain a centre-bias (or a zero-bias) operator that helps them perform extremely well on functions that have their respective optima in the centre of the feasible set (or give very good values when evaluated there). Some of these methods were analysed on and tuned to benchmark sets where such functions constitute the majority of the problems. When compared with other methods, they seemed to offer superior performance. However, when evaluated on shifted problems, most of their superior performance all but vanishes. And when evaluated on a proper benchmark set, some of them turn out to perform just barely better than a random search.

In a recent large computational study[34], the authors compared 95 EC techniques on a benchmark set of 195 test functions implemented in scipy[35]. Out of the methods presented in this paper, DE ranked the best (thirteenth), followed by SMA and PSO, while both BOA and STOA rank among the worst. More interestingly, the authors also studied the effect of reducing the number of considered test functions and algorithms on the resulting ranking. They have found that STOA improved its rank from eighty-sixth to being the best algorithm if the number of problems/algorithms was reduced by half (with similar statements holding for more algorithms). A question is whether this behaviour could be explained by the sensitivity of the test functions to the centre-bias operator that STOA contains.

The one method that performed consistently well on all benchmark sets considered in this work was MPA. Not coincidentally, MPA was the only method that was in its original publication thoroughly tested also on more complex benchmark problems (from CEC 2014 and CEC 2017 competitions) and was compared to several state-of-the-art methods based on DE (best-performing methods from some of the CEC competitions). The second-best method from the comparison on the ambiguous benchmark set and one that also performed well on the real-world benchmark set was KMA. Although KMA was not tested on any advanced benchmark set, it was compared to some of the DE-based state-of-the-art methods. Unfortunately, comparing newly proposed methods to state-of-the-art techniques is still the exception rather than the norm[9,34].

Similar to other authors[14,15], we believe that benchmark sets should also include instances of real-world problems. This seems particularly important since it was recently found that a large portion (near 30%) of nature-inspired algorithms studied previously[36] had no application associated to them. But, as we have seen, there are only a handful of frequently used real-world engineering problems, and these are too easy to serve as benchmarks. The EC community should initiate the construction of a large set of challenging real-world benchmark problems. The Mixed Integer Programming Library (MIPLIB)[37], developed

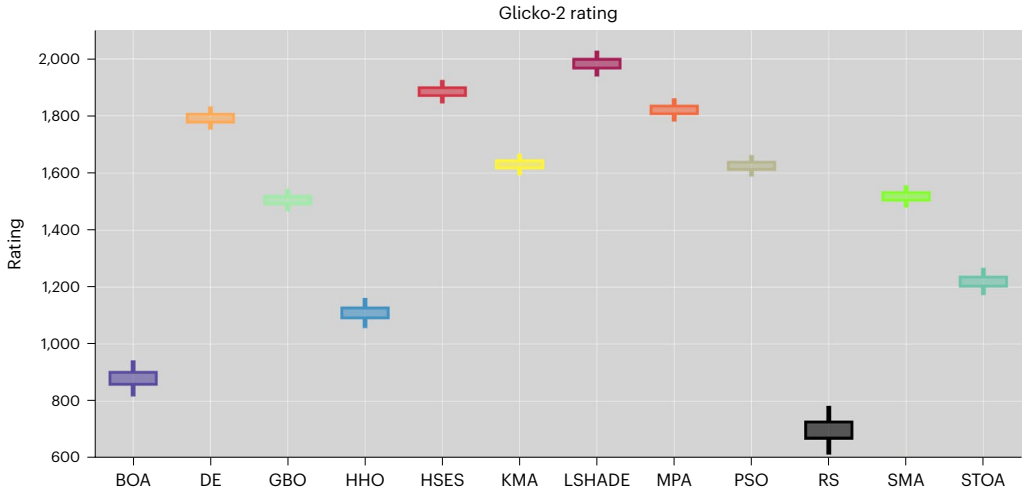


Fig. 1 | Results of the Glicko-2 rating on the ambiguous benchmark set. Data are presented as mean values +/− 1 standard deviation (rectangular areas) and mean values +/− 3 standard deviations (vertical lines), based on 25 games played between each method.

by the mathematical programming community, could serve as a great source of inspiration. Its most current version (MIPLIB 2017) consists of 1,065 carefully chosen real-world integer programming problems (from both academia and industry), 240 of which are specifically selected for benchmarking the performance of optimization solvers.

One possible reason for the ubiquity of the troublesome benchmark problems is their ease of use. They are relatively straightforward to program, and all of the methods considered in this paper included them in their respective source code (except for GBO, which used different functions with the same problem). We advocate for the construction of an easy-to-use cross-platform repository, that would collect: (1) several heterogeneous benchmark sets (BBOB, CEC competitions, ambiguous benchmark set, real-world benchmarks, and similar ones) with a unified way of calling the test problems; (2) trusted implementations (source codes) of both standard EC methods and up-to-date state-of-the-art techniques; (3) data obtained from running the algorithms (from (2)) on the benchmarks (from (1)). The use of heterogeneous benchmarks in testing EC methods has long been advocated for[14], and we have seen that the best-performing method from our comparison was the one that was originally tested on multiple benchmark sets.The ability to compare both standard and state-of-the-art methods is also needed (again, the best methods from our comparison were compared against the state-of-the-art ones). Having a single repository with both the source code and the corresponding data to these methods would greatly benefit both the researchers who want to experiment with different modifications of these methods, or, conversely, the ones who have only limited experience with the programming language in which the method is written, but nevertheless want to use them in their comparisons. In this regard, we feel that the COCO platform is a step in the right direction, but it lacks breadth (it uses only the BBOB benchmarks and provides limited source codes) and is still very underutilized.

It is our opinion that the field of evolutionary computation needs to adopt more rigorous benchmarks and approaches to analyse and compare newly proposed methods.

Methods

Source code

The links for the source codes for the different methods can be found in the respective publications. For the DE and PSO algorithms, we used the implementations that were shipped alongside the files for the CEC 2021 (ref. [38]) and CEC 2020 (ref. [39]) competitions, respectively. All the parameters for the considered methods, as well as the code used to run the experiments (written in MATLAB), can be found at https://doi.org/10.24433/CO.1268126.v1 (ref. [30]).

Ambiguous benchmark set

The functions in the ambiguous benchmark set[32] are based on a zigzag pattern[40]. The functions were chosen with the help of several of the best-performing methods from different CEC competitions, as well as some standard ones. Each of the functions induced statistically significant ranking among the algorithms, but the ranking for the whole set remained ambiguous (hence the name).

Glicko-2 rating

The Glicko-2 rating is an Elo-base system that uses games between the methods (based on randomly selected runs). Glicko-2 was found to be particularly suitable for comparing evolutionary algorithms[33,41], giving similar conclusions to the nonparametric Wilcoxon's test[42]. We used the implementation of this ranking within the IOHProfiler[43], a web-based benchmarking and profiling tool for (meta)heuristics used in optimization.

Data availability

Data used for the benchmark functions will be made available at https://doi.org/10.24433/CO.1268126.v1 (ref. [30]).

Code availability

The code that supports the findings of this study will be made available at https://doi.org/10.24433/CO.1268126.v1 (ref. [30]).

References

1. Campelo, F. & Aranha, C. Evolutionary computation bestiary. https://github.com/fcampelo/EC-Bestiary (accessed 7 February 2022).
2. Weyland, D. A rigorous analysis of the harmony search algorithm: how the research community can be misled by a novel methodology. *Int. J. Appl. Metaheuristic Comput.* **12**, 50–60 (2010).
3. Camacho Villalón, C. L., Dorigo, M. & Stützle, T. The intelligent water drops algorithm: why it cannot be considered a novel algorithm. *Swarm Intell.* **13**, 173–192 (2019).
4. Camacho Villalón, C. L., Stützle, T. & Dorigo, M. Grey wolf, firefly and bat algorithms: three widespread algorithms that do not contain any novelty. In *Int. Conference on Swarm Intelligence* 121–133 (Springer, 2020).

5. Camacho Villalón, C. L., Stützle, T. & Dorigo, M. *Cuckoo Search ≡ μ+λ – Evolution Strategy — A Rigorous Analysis of an Algorithm that has Been Misleading the Research Community for More Than 10 Years and Nobody Seems to have Noticed* TR/IRIDIA/2021-006 (IRIDIA, Université Libre de Bruxelles, 2021).
6. Piotrowski, A. P., Napiorkowski, J. J. & Rowinski, P. M. How novel is the "novel" black hole optimization approach? *Inf. Sci.* **267**, 191–200 (2014).
7. Aranha, C. et al. Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intell.* **16**, 1–6 (2022).
8. Hellwig, M. & Beyer, H. G. Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – a critical review. *Swarm Evol. Comput.* **44**, 927–944 (2019).
9. Garcia-Martinez, C., Gutierrez, P. D., Molina, D., Lozano, M. & Herrera, F. Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. *Soft Comput.* **21**, 5573–5583 (2017).
10. Hansen, N., Auger, A., Mersmann, O., Tuvar, T. & Brockhoff, D. COCO: a platform for comparing continuous optimizers in a black-box setting. Preprint at https://arxiv.org/abs/1603.08785 (2016).
11. Suganthan, N. P. Github repository of CEC competitions. *GitHub* https://github.com/P-N-Suganthan (2022).
12. Garden, R. W. & Engelbrecht, A. P. Analysis and classification of optimization benchmark functions and benchmark suites. In *IEEE Congress on Evolutionary Computation* 1664–1669 (2014).
13. *COCO Data Archives* (2022); https://numbbo.github.io/data-archive/
14. Piotrowski, A. P. Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. *Inf. Sci.* **297**, 191–201 (2015).
15. Tzanetos, A. & Dounias, G. Nature inspired optimization algorithms or simply variations of metaheuristics? *Artif. Intell. Rev.* **54**, 1841–1862 (2021).
16. Kumar, A., Suganthan, P. N., Mohamed, A. W., Hadi, A. A. & Mohamed, A. K. Special session & competitions on single objective bound constrained numerical optimization. In *IEEE Congress on Evolutionary Computation* (IEEE, 2021).
17. Niu, P., Niu, S., Liu, N. & Chang, L. The defect of the Grey Wolf optimization algorithm and its verification method. *Knowl.-Based Syst.* **171**, 37–43 (2019).
18. Castelli, M., Manzoni, L., Mariot, L., Nobile, M. S. & Tangherloni, A. Salp Swarm Optimization: a critical review. *Expert Syst. Appl.* **189**, 116029 (2022).
19. Kudela, J. Commentary on: "STOA: A bio-inspired based optimization algorithm for industrial engineering problems" [EAAI, 82 (2019), 148–174] and "Tunicate Swarm Algorithm: A new bio-inspired based metaheuristic paradigm for global optimization" [EAAI, 90 (2020), no. 103541]. *Eng. Appl. Artif. Intell.* **113**, 104930 (2022).
20. Suyanto, S., Ariyanto, A. A. & Ariyanto, A. F. Komodo Mlipir Algorithm. *Appl. Soft Comput.* **114**, 108043 (2022).
21. Li, S., Chen, H., Wang, M., Heidari, A. A. & Mirjalili, S. Slime mould algorithm: a new method for stochastic optimization. *Future Gener. Comput. Syst.* **111**, 300–323 (2020).
22. Arora, S. & Singh, S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Comput.* **23**, 715–734 (2019).
23. Ahmadianfar, I., Bozorg-Haddad, O. & Chu, X. Gradient-based optimizer: a new metaheuristic optimization algorithm. *Inf. Sci.* **540**, 131–159 (2020).
24. Oszust, M. Enhanced marine predators algorithm with local escaping operator for global optimization. *Knowl.-Based Syst.* **232**, 107467 (2021).
25. Heidari, A. A. et al. Harris hawks optimization: algorithm and applications. *Future Gener. Comput. Syst.* **97**, 849–872 (2019).
26. Dhiman, G. & Kaur, A. STOA: a bio-inspired based optimization algorithm for industrial engineering problems. *Eng. Appl. Artif. Intell.* **82**, 148–174 (2019).
27. Tanabe, R. & Fukunaga, A. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation* 1658–1665 (IEEE, 2014).
28. Zhang, G. & Shi, Y. Hybrid sampling evolution strategy for solving single objective bound constrained problems. In *2018 IEEE Congress on Evolutionary Computation* (IEEE, 2018).
29. Fister, I. et al. On selection of a benchmark by determining the algorithms' qualities. *IEEE Access* **9**, 51166–51178 (2021).
30. *CodeOcean Capsule* (2022); https://doi.org/10.24433/CO.1268126.v1
31. Bayzidi, H., Talatahari, S., Saraee, M. & Lamarche, C.-P. Social network search for solving engineering optimization problems. *Comput. Intell. Neurosc.* **9**, 8548639 (2021).
32. Kudela, J. & Matousek, R. New benchmark functions for single-objective optimization based on a zigzag pattern. *IEEE Access* **10**, 8262–8278 (2022).
33. Vecek, N., Crepinsek, M., Mernik, M. & Hrncic, D. A comparison between different chess rating systems for ranking evolutionary algorithms. In *2014 Federated Conference on Computer Science and Information Systems* 511–518 (IEEE, 2014).
34. Del Ser, J. et al. More is not always better: insights from a massive comparison of meta-heuristic algorithms over real-parameter optimization problems. In *IEEE Symposium Series on Computational Intelligence* (IEEE, 2021).
35. Scipy benchmark functions. *GitHub* https://github.com/scipy/scipy/tree/main/benchmarks/benchmarks/go_benchmark_functions (2022).
36. Tzanetos, A. & Dounias, G. A comprehensive survey on the applications of swarm intelligence and bio-inspired evolutionary strategies. *Mach. Learn. Paradigms* **18**, 337–378 (2020).
37. Gleixner, A. et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library. *Math. Program. Comput.* **13**, 443–490 (2021).
38. Mohamed, A.W. et al. *Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization* (Cairo University, 2020).
39. Yue, C.T. et al. Problem *Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization* Technical report 201911 (Computational Intelligence Laboratory, Zhengzhou University, 2019).
40. Kudela, J. Novel zigzag-based benchmark functions for bound constrained single objective optimization. In *2021 IEEE Congress on Evolutionary Computation* (IEEE, 2021).
41. Vecek, N., Crepinsek, M. & Mernik, M. On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms. *Appl. Soft Comput.* **54**, 23–45 (2017).
42. Osaba, E. et al. A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm Evol. Comput.* **64**, 100888 (2021).
43. Doerr, C., Wang, H., Ye, F., van Rijn, S. & Back, T. IOHprofiler: a benchmarking and profiling tool for iterative optimization heuristics. Preprint at https://arxiv.org/abs/1810.05281 (2018).

Acknowledgements
This work was supported by the Grant Agency of the Czech Republic project 22-31173S and by the Brno University of Technology project FSI-S-20-6538.

Author contributions
J.K. performed the conceptualization, design, data analysis and interpretation, drafting of the manuscript and critical revision of the manuscript for important intellectual content.

Competing interests
The author declares no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-022-00579-0.

Correspondence and requests for materials should be addressed to Jakub Kudela.

Peer review information *Nature Machine Intelligence* thanks Alexandros Tzanetos and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

A4

Computational and Exploratory Landscape Analysis of the GKLS Generator

Jakub Kudela*
Jakub.Kudela@vutbr.cz
Institute of Automation and Computer Science
Brno University of Technology
Brno, Czech Republic

Martin Juricek
200543@vutbr.cz
Institute of Automation and Computer Science
Brno University of Technology
Brno, Czech Republic

ABSTRACT

The GKLS generator is one of the most used testbeds for benchmarking global optimization algorithms. In this paper, we conduct both a computational analysis and the Exploratory Landscape Analysis (ELA) of the GKLS generator. We utilize both canonically used and newly generated classes of GKLS-generated problems and show their use in benchmarking three state-of-the-art methods (from evolutionary and deterministic communities) in dimensions 5 and 10. We show that the GKLS generator produces "needle in a haystack" type problems that become extremely difficult to optimize in higher dimensions. We also conduct the ELA on the GKLS generator and then compare it to the ELA of two other widely used benchmark sets (BBOB and CEC 2014), and discuss the results.

CCS CONCEPTS

• **Computing methodologies** → Randomized search; Heuristic function construction; Continuous space search.

KEYWORDS

Benchmarking, Exploratory Landscape Analysis, GKLS, Global optimization, Black-box optimization

ACM Reference Format:
Jakub Kudela and Martin Juricek. 2023. Computational and Exploratory Landscape Analysis of the GKLS Generator. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3583133.3590653

1 INTRODUCTION

Solving real-world black-box optimization problems is a very challenging task, especially in problems with expensive function evaluation that require simulation runs [10]. The development, utilization, and comparison of optimization methods on such real-world problems are usually prohibitively expensive. For such tasks, benchmarking optimization methods on artificially constructed testbeds become pivotal, with the expectation that the behavior of the methods on these benchmark sets translates well into real-world problems.

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3590653

Over the years, various benchmark suites have been proposed, in which different global function properties are represented. In the evolutionary computation community, the two most utilized benchmark sets are the Black-Box Optimization Benchmarking (BBOB) suite [4] which is now part of the COCO platform [5], and the suites that were presented at the Congress on Evolutionary Computation (CEC) competitions [8].

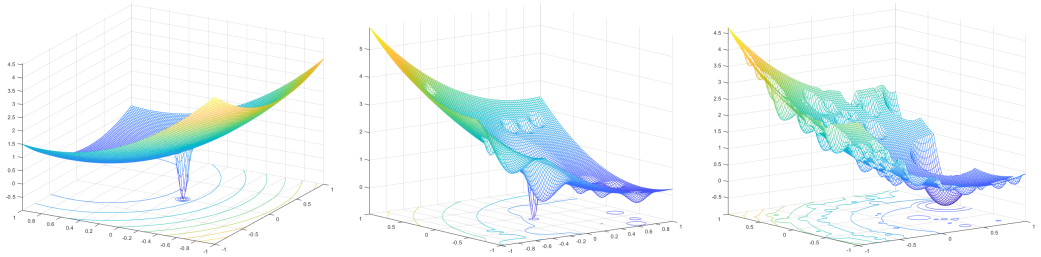
In the global (deterministic) optimization community, one of the most popular benchmark sets is the one produced by the GKLS generator [3]. The advantage of the GKLS generator is that for each generated problem, the location and function value of its local and global minima are known. Although the GKLS generator can be used to create various types of problems (based on input parameters), there are 8 classes of problems (2 for dimensions 2, 3, 4, and 5) each containing 100 functions that are generally used [14, 19]. The GKLS generator was also recently used for the construction of general-constrained [16] test problems.

In order to quantify the low-level properties of optimization problems, various features of the landscape can be computed [11]. Such analysis falls under the field of Exploratory Landscape Analysis (ELA) [12]. The most notable uses of ELA are in the visualization of the problem space of various optimization benchmark problem sets [17], and in automated algorithm selection [6]. In this paper, we conduct computational analysis and ELA of the GKLS-generated problems. An extended version of this paper can be found in [9].

2 GKLS GENERATOR

In the GKLS generator, a prespecified number of test problems (a class of problems) is constructed by defining a convex quadratic function (a paraboloid) which is systematically distorted by polynomials in order to produce local (and one global) minima. The input parameters for this construction are the following: type of the problem (ND: non-differentiable, D: differentiable, D2: twice-differentiable), problem dimension (D), number of local minima (h), the value of the global minimum (f^*), radius (r) of the attraction region of the global minimizer, and the distance (d) from the global minimizer to the vertex of the quadratic function. All problems are constructed on $[-1, 1]^D$.

A visualization of different functions that can be generated by the GKLS is shown in Figure 1. Several interesting observations can be made regarding the generated functions. Firstly, they are all relatively well-conditioned. The local minimum of the "big" paraboloid is always in the domain and has a function value of 0. The "attraction regions" of the different local minima do not overlap (this is by design) - this also means that they become more shallow when their number increases. The eight most used classes



$h = 2$, type D, $d = 0.9$, $r = 0.1$ $h = 20$, type D, $d = 0.9$, $r = 0.1$ $h = 100$, type ND, $d = 0.66$, $r = 0.5$

Figure 1: Functions generated by the GKLS in dimension $D = 2$ ($f^* = -1$ for all functions).

Table 1: Most used GKLS test classes.

Class	"Difficulty"	Type	D	f^*	d	r	h
1	simple	D	2	-1	0.90	0.20	10
2	hard	D	2	-1	0.90	0.10	10
3	simple	D	3	-1	0.66	0.20	10
4	hard	D	3	-1	0.90	0.20	10
5	simple	D	4	-1	0.66	0.20	10
6	hard	D	4	-1	0.90	0.20	10
7	simple	D	5	-1	0.66	0.30	10
8	hard	D	5	-1	0.66	0.20	10

(each with 100 functions) of GKLS-generated problems are shown in Table 1.

3 COMPUTATIONAL ANALYSIS

Although the classes in the same dimension share some characteristics, what is arguably more important is whether different algorithms will "perform" in the same way on these classes of problems. We run three state-of-the-art methods both from the evolutionary computation (EC) and deterministic optimization communities on the "canonical" GKLS-generated problems in dimensions 5 and 10 (which uses the same "simple" and "hard" parameters as dimension 5). We also construct a new class ("mod") with 50 GKLS-generated problems (again, in $D = 5$, and 10) by the following procedure:

- Each problem $i = 1, \ldots, 50$ in this class has different values of the parameters type_i, d_i, r_i, and h_i, but the same $f_i^* = -1$.
- type_i is decided by a coin flip between types D and ND (with the same probability).
- d_i is a uniformly distributed random number on [0,1].
- $r_i = d_i/u_i$, where u_i is a uniformly distributed random integer on [2,10], i.e. $r_i \in [d_i/2, d_i/10]$.
- $h_i = \text{round}(10^{c_i})$, where c_i is a uniformly distributed random number on [1,3], i.e. $h \in [10, 10^3]$.

From the EC side, we chose two methods to run on the GKLS-generated problems. The first selected method was Adaptive Gaining-Sharing Knowledge (AGSK), which was the runner-up of the CEC'20 competition [13]. The second method is L-SHADE or Success-history based adaptive differential evolution with linear population size reduction [21]. From the deterministic methods, we selected BIRMIN [14] as one of the best-performing methods from a recent extensive numerical study [19].

The implementation and parameter choices for LSHADE and AGSK were taken from [1] (the implementations are available at the GitHub[1] of one of the authors). The implementation and parameter choices for BIRMIN were taken from the DIRECTGOLib [20].

For the numerical comparison, we run each method once on every problem from each of the three classes (100 problems in both "simple" and "hard" classes, and 50 problems in the "mod" class) in dimensions $D = [5, 10]$, with a budget of $5{\cdot}10^4{\cdot}D$ available function evaluations. For every run, if the objective function value of the resulting solution was less than or equal to 1E−8, it was considered as zero. The code for the experiments (and the generator for the test problems) can be found at the author's Github[2].

For dimension $D = 5$, the Empirical cumulative distributions (ECDs) of simulated runtimes, measured in the number of function evaluations for 51 targets $10^{[-8..2]}$ (similar analysis which is done in the COCO platform) are shown in Figure 2. There is a noticeable difference in the behavior of the three algorithms on the "simple" and "hard" classes. On the "simple" class all three algorithms were able to find either a good or the optimal solutions faster than on the "hard" class. There is also a quite large difference between the performance of the three different methods - BIRMIN clearly dominated the two EC methods, and AGSK turned out to be better at finding good solutions at the later stages of the search than LSHADE.

The results change quite dramatically when looking at the "mod" class. Although the performance of BIRMIN is still superior to that of the two EC methods, the margin narrowed substantially. What is more, the relative performance (against the "hard" class) of AGSK decreased, while for LSHADE it increased. For all three classes, the local minimum of the "big" paraboloid (error value 1) was found by every method within a few hundred function evaluations for BIRMIN and a few thousand function evaluations for the EC methods.

The results for dimension $D = 10$ show another substantial change - we can see that all three classes are basically equivalent. The plateaus in the ECD plots between the 0.19 and 0.20 values on the y-axis indicate the points where the methods found the local minimum of the "big" paraboloid. Although this happened relatively early for all methods, finding better local optima proved to be extremely challenging.

[1] https://github.com/subhodipbiswas/MadDE
[2] https://github.com/JakubKudela89/GKLS-GECCO

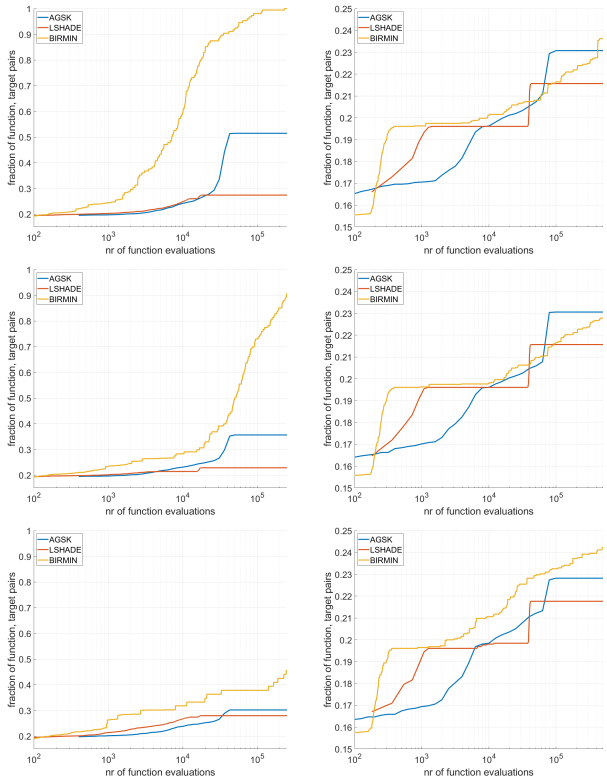


Figure 2: ECD of simulated runtimes, measured in number of f-evaluations for the 51 targets $10^{[-8..2]}$ in dimension a) $D = 5$, b) $D = 10$.

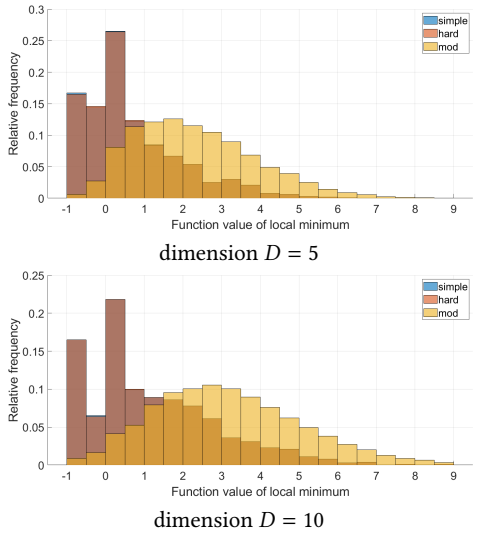


Figure 3: Relative frequencies of function values of local minima of the different classes.

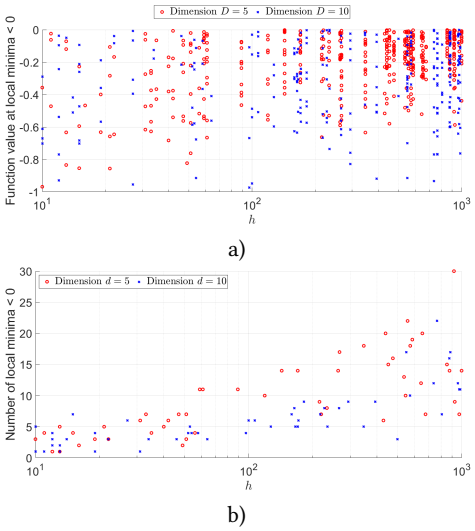


Figure 4: Scatter plots of the dependence of a) function values of local minima with function value < 0, b) number of local minima with function value < 0 on h and dimension D.

Looking back to the function plots shown in Figure 1, we see that having more local minima (i.e., higher h) meant that they became shallower. We can see this effect in Figure 3, where the relative frequencies of function values of the local minima for the three classes are plotted (the "simple" and "hard" classes have almost the same values of all the local minima, which is a consequence of the implemented pseudorandom number generator). In Figure 4 we can see that although there are really fewer local minima in dimension $D = 10$ with negative function values, they are spread out more evenly than in the case of dimension $D = 5$, even when the number of local minima is high.

The functions that the GKLS generator produces are of the "needle in a haystack" kind [2]. If one does not stumble upon the region of the space where the global minimizer (or at least a local minimizer with "good" function value) has its region of attraction, either by chance (in the case of the EC methods) or by space partition (whose cost is bound to be exponential in the dimension D), the solution one gets is the local minimum of the "big" paraboloid. The functions give no hints on where such good points might be.

4 EXPLORATORY LANDSCAPE ANALYSIS

We use ELA features [7] to show how the GKLS-generated problems compare to the BBOB and CEC 2014 benchmark suits. We chose ELA feature sets which only require samples of input and function value pairs and dimension $D = 10$ for all considered suits. We chose to ignore the features that were sensitive to function transformations [18] and used uniform sampling with $250 \cdot D$ samples their computation [15].

We then followed the methodology described in [17] for the selection and visualization of the relevant ELA features. The features that produced constant results on every problem and those that produced invalid values were removed. Another batch of features that got removed were the highly correlated ones.

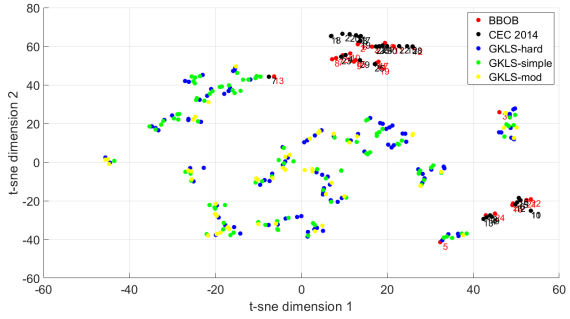


Figure 5: The t-sne visualization of the ELA features (after normalization and using the first seven components from the PCA) of the benchmark sets.

The values of the ELA features on the different benchmark sets were then normalized, and we used Principal Component Analysis (PCA) to reduce the number of features even further. Using the first 7 PCA components explained 99.68% of the variance. For visualizing the results, we used the t-Distributed Stochastic Neighbor Embedding (t-sne). In this visualization, which is shown in Figure 5, benchmark problems that have similar ELA features should be shown close to each other. We can see that the t-sne visualization grouped most of the functions from the BBOB and CEC 2014 suits together (in a few groups), while the "similar" problems generated in the three GKLS suites take up most of the space.

5 CONCLUSION

In this paper, we analyzed the problems constructed by the GKLS generator. In the computational analysis, we have shown that it produces "needle in a haystack " type problems which get extremely difficult to optimize as the problem dimension grows. The GKLS generator could be successfully used for benchmarking state-of-the-art methods in lower dimensions ($D = 5$) on some of the simpler instances. However, in the higher dimension ($D = 10$), the performance of the three considered methods was hard to differentiate as the problems became extremely difficult (given the computational budget). This difficulty of the generated instances was also largely unaffected by the choice of parameters that the generator has.

It is possible that the GKLS generator could be modified to have a much "deeper" local minima. As the task of finding the global minimum is practically impossible in higher dimensions, having problems with lots of "good" local minima (i.e., better ones than the local minimum of the "big" paraboloid) could be useful for analyzing the exploration capabilities of optimization methods.

It is not very clear how one could meaningfully use the results of the computations of the ELA features or the t-sne plots on such "needle in a haystack" problems. It is probably impossible to have any sample-based features that would both uncover that the problem is a "needle in a haystack" and be computationally tractable (as it would amount to finding the "needle" in a reasonable amount of function evaluations).

Lastly, it would be interesting to find real-world black-box continuous problems which have the "needle in a haystack" character and show if the methods that were "trained" on such problems (such as BIRMIN) really offer an advantage (at least in lower dimensions).

ACKNOWLEDGMENTS

This work was supported by the IGA BUT No. FSI-S-23-8394 "Artificial intelligence methods in engineering tasks".

REFERENCES

[1] Subhodip Biswas, Debanjan Saha, Shuvodeep De, Adam D Cobb, Swagatam Das, and Brian A Jalaian. 2021. Improving differential evolution through Bayesian hyperparameter optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 832–840.

[2] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. Optimization with randomized search heuristics—the (A) NFL theorem, realistic scenarios, and difficult functions. *Theoretical Computer Science* 287, 1 (2002), 131–144.

[3] Marco Gaviano, Dmitri E Kvasov, Daniela Lera, and Yaroslav D Sergeyev. 2003. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software (TOMS)* 29, 4 (2003), 469–480.

[4] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. 2012. *Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup.* Technical Report. INRIA.

[5] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. 2021. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *Optimization Methods and Software* 36 (2021), 114–144. Issue 1. https://doi.org/10.1080/10556788.2020.1808977

[6] Pascal Kerschke and Heike Trautmann. 2019. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary computation* 27, 1 (2019), 99–127.

[7] Pascal Kerschke and Heike Trautmann. 2019. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In *Applications in Statistical Computing*. Springer, 93–123.

[8] Jakub Kudela. 2022. A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence* 4 (2022), 1238–1245.

[9] Jakub Kudela and Martin Juricek. 2023. Computational and Exploratory Landscape Analysis of the GKLS Generator. *arXiv preprint arXiv:2304.08913* (2023).

[10] Jakub Kudela and Radomil Matousek. 2022. Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Computing* 26, 24 (2022), 13709–13733.

[11] Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V Kononova. 2022. BBOB Instance Analysis: Landscape Properties and Algorithm Performance across Problem Instances. *arXiv preprint arXiv:2211.16318* (2022).

[12] Olaf Mersmann, Mike Preuss, and Heike Trautmann. 2010. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In *Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11*. Springer, 73–82.

[13] Ali Wagdy Mohamed, Anas A Hadi, Ali Khater Mohamed, and Noor H Awad. 2020. Evaluating the performance of adaptive gainingsharing knowledge based algorithm on CEC 2020 benchmark problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[14] Remigijus Paulavičius, Yaroslav D Sergeyev, Dmitri E Kvasov, and Julius Žilinskas. 2020. Globally-biased BIRECT algorithm with local accelerators for expensive global optimization. *Expert Systems with Applications* 144 (2020), 113052.

[15] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. 2020. Exploratory landscape analysis is strongly sensitive to the sampling strategy. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II 16*. Springer, 139–153.

[16] Yaroslav D Sergeyev, Dmitri E Kvasov, and Marat S Mukhametzhanov. 2021. A Generator of Multiextremal Test Classes With Known Solutions for Black-Box-Constrained Global Optimization. *IEEE Transactions on Evolutionary Computation* 26, 6 (2021), 1261–1270.

[17] Urban Škvorc, Tome Eftimov, and Peter Korošec. 2020. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing* 90 (2020), 106138.

[18] Urban Škvorc, Tome Eftimov, and Peter Korošec. 2022. A Comprehensive Analysis of the Invariance of Exploratory Landscape Analysis Features to Function Transformations. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[19] Linas Stripinis and Remigijus Paulavičius. 2022. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms. *arXiv preprint arXiv:2209.05759* (2022).

[20] Linas Stripinis and Remigijus Paulavičius. 2022. *DIRECTGOLib - Global Optimization test problems Library.* https://doi.org/10.5281/zenodo.6617799

[21] Ryoji Tanabe and Alex S Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 1658–1665.

A5

A collection of robotics problems for benchmarking evolutionary computation methods

Jakub Kůdela[*], Martin Juříček, and Roman Parák

Institute of Automation and Computer Science, Brno University of Technology
Technicka 2, 621 00 Brno, Czech Republic
Jakub.Kudela@vutbr.cz[*], 200543@vutbr.cz, Roman.Parak@vutbr.cz

Abstract. The utilization of benchmarking techniques has a crucial role in the development of novel optimization algorithms, and also in performing comparisons between already existing methods. This is especially true in the field of evolutionary computation, where the theoretical performance of the method is difficult to analyze. For these benchmarking purposes, artificial (or synthetic) functions are currently the most widely used ones. In this paper, we present a collection of real-world robotics problems that can be used for benchmarking evolutionary computation methods. The proposed benchmark problems are a combination of inverse kinematics and path planning in robotics that can be parameterized. We conducted an extensive numerical investigation that encompassed solving 200 benchmark problems by seven selected metaheuristic algorithms. The results of this investigation showed that the proposed benchmark problems are quite difficult (multimodal and non-separable) and that they can be successfully used for differentiating and ranking various metaheuristics.

Keywords: Evolutionary computation · Metaheuristics · Benchmarking · Robotics.

1 Introduction

The field of evolutionary computation (EC) produced over its long history several crucial metaheuristic optimization algorithms, that took inspiration from natural processes. These algorithms found their use in numerous complex applications, where the utilization of conventional methods was inadequate or overly computationally demanding [23]. Over the last decade, there has been an explosion of "novel" methods that draw on natural principles [4]. Many of these novel methods have been found to hide their lack of novelty behind a metaphor-rich jargon [3], or flawed experimental analysis [27,21].

As nature-inspired metaheuristics are usually hard to analyze analytically, their utility is conventionally analyzed through benchmarking [13]. There have been many different benchmark functions and sets proposed by various authors

[9,22], but the most popular and widely used benchmark set have been developed for special sessions (competitions) on black-box optimization in two EC conferences: the IEEE Congress on Evolutionary Computation (CEC), and the Genetic and Evolutionary Conference (GECCO), where the Black-Box Optimization Benchmarking (BBOB) workshop was held. The BBOB functions constitute a part of the COCO platform for comparing optimization algorithms [12], while the benchmark functions from the different CEC competitions can be found on GitHub of one of the authors. It was shown that the characteristics of the functions used in the CEC and BBOB benchmark sets are very different [10]. However, the use of these benchmark sets is not without critique, as some authors criticized the artificial nature of these benchmark sets [31], and recommended testing optimization algorithms on real-world problems instead [39].

A possible way of comparing the characteristics of the different benchmark sets is by using the Exploratory Landscape Analysis (ELA) [25]. In this approach, the benchmark functions are represented by a collection of landscape features (numerical measures) that describe the different aspects of the functions. The ELA can subsequently be used for designing representative benchmark suites [5], or for feature-based algorithm selection [37].

One of the areas that utilized EC methods to a large extent is robotics [40,29]. The locomotion of snake-like robots using genetic algorithms (GA) was investigated in [14]. The inverse kinematics (IK) problem in robot path tracking [15,30] was approached using particle swarm optimization (PSO) variants [8] or slime mould algorithm [41]. EC methods were used in tracking control of redundant mobile manipulator [20], robot part sequencing and allocation problem with collision avoidance [6], or in the control tuning of omnidirectional mobile robots [33]. Another robotics application where EC methods are widely utilized is trajectory or path planning [1,24]. Using GA and PSO, the authors of [28] studied energy-efficient robot configuration and motion planning. Another applications of EC methods [32] investigated an autonomous unmanned aerial vehicle path-planning for predisaster assessment or path planning and tracking of a quadcopter for payload hold-release missions [2].

In this paper, we present a collection of parametrizable problems in robotics that combine inverse kinematics and path planning problems, and show that they can be successfully used in benchmarking EC methods. The rest of the paper is structured as follows. Section 2 describes the 6-DOF (Degrees of Freedom) collaborative robotic arm that is used as the base framework. In Section 3 are defined the benchmark problems and their parametrization. In Section 4 we briefly describe the seven EC methods that were selected for running numerical tests on the proposed benchmarks. Section 5 gives a detailed account of the results of the numerical test. In Section 6, we provide ELA of the problems and a comparison to problems from the BBOB and CEC 2014 sets. Finally, conclusions and future research directions are discussed in Section 7.

2 Forward Kinematics of a Robotic Arm

As the framework for the benchmark problems, we chose the 6-DOF collaborative robotic arm UR3 CB-Series[1], shown in Figure 1. The solution of forward kinematics (FK) requires the knowledge of the so-called Denavit-Hartenberg (D-H) table, shown in Table 1. The table has become a standard used in robotics, having been first published in [7]. The commonly used convention of the D-H table is defined by four parameters. These parameters describe how the reference frame of each link is attached to the robot. Each inertial reference frame of each link is then assigned an additional robot reference frame. The parameters are defined for each joint $i \in [1, n]$, which represents the table:

- α_i : angle about common normal from z_i to z_{i+1}
- θ_i : angle about previous z axis from x_i to x_{i+1}
- a_i : length of the common normal or radius about previous z axis for revolute joint
- d_i : offset along previous z axis to the common normal

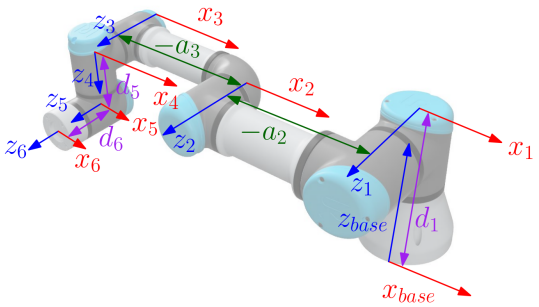


Fig. 1. UR3 D-H parameters

The solution for the calculation of the FK itself using the D-H parameters can then be approached by multiplying the individual D-H matrices. For the solution, so-called standard (distal) matrices or modified (proximal) matrices can be used. The basic difference between the standard D-H parameters and the modified D-H parameters is the locations of the coordinates system attached to the links. The modification consists of both the calculation of the D-H matrix itself and the change of the D-H table. To switch from a standard D-H table to a modified D-H table, it is necessary to perform the shift operation $\alpha_i = \alpha_{i-1}$ and $a_i = a_{i-1}$ where $i \in [1, n]$, $\alpha_1 = \alpha_n$, $a_1 = a_n$. The calculation of the forward kinematics is then a matrix in which the vector $\mathbf{p}$ represents the translational

[1] https://www.universal-robots.com/cb3/

part and $\mathbf{R}$ represents the rotation matrix, which can then be converted into quaternions or euler angles. The modified D-H matrix can then have its advantage when calculating the Jacobian or determining the rotation and translation of the individual joints of the robot. Homogeneous transformation matrix $\mathbf{T}_i$ is represented as the product of four basic transformations [34]. The calculation of the resulting matrix $\mathbf{T}_n$ of the n-axis robot can generally be described mathematically as follows:

$$\mathbf{T}_n = \prod_{i=1}^{n} \mathbf{T}_i \tag{1}$$

$$\mathbf{T}_i = \mathbf{Rot}_{z_i\theta_i}\mathbf{Trans}_{z_i d_i}\mathbf{Trans}_{x_i a_i}\mathbf{Rot}_{x_i\alpha_i} =$$

$$= \left[\begin{array}{ccc|c} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & r_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & r_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p} \\ \hline 0\ 0\ 0 & 1 \end{array}\right]$$

$$\mathbf{T}_i = \mathbf{Rot}_{x_i\alpha_{i-1}}\mathbf{Trans}_{x_i a_{i-1}}\mathbf{Rot}_{z_i\theta_i}\mathbf{Trans}_{z_i d_i} =$$

$$= \left[\begin{array}{ccc|c} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & -d_i\cos\alpha_{i-1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{p} \\ \hline 0\ 0\ 0 & 1 \end{array}\right]$$

Table 1. Universal Robots UR3 CB-Series D-H table

Joint i	θ_i [rad]	α_i [rad]	a_i [m]	d_i [m]
1	0	$\pi/2$	0	0.1519
2	0	0	-0.24365	0
3	0	0	-0.21325	0
4	0	$\pi/2$	0	0.11235
5	0	$-\pi/2$	0	0.08535
6	0	0	0	0.0819

3 Definition of the Benchmark Problems

The α_i, a_i, and d_i values are fixed (as they depend on the particular robot design) and the robot is controlled by changing the angles θ_i. The [x,y,z]-coordinate position of the last link of the robot can be found as the translation part $\mathbf{p}$ in the matrix $\mathbf{T}_n$. We will denote this relationship simply as

$$[x, y, z]^T = FK(\theta), \tag{2}$$

where FK is the forward kinematics solution, and $\theta = [\theta_1, \ldots, \theta_6]^T$,$\theta_i \in [-2\pi, 2\pi]$, $i = 1, \ldots, 6$. In the proposed benchmark problems, we will be interested in the trajectories of the robot's last link (end-effector), which corresponds to the way θ changes in time τ, and can be expressed as

$$[x(\tau), y(\tau), z(\tau)]^T = FK(\theta(\tau)). \tag{3}$$

The first quality of the trajectory we will use is its length L, which (starting in $\tau = 0$ and ending in $\tau = 1$) can be expressed as

$$L = \int_0^1 \sqrt{x'(\tau)^2 + y'(\tau)^2 + z'(\tau)^2}\mathrm{d}\tau. \tag{4}$$

The second quality of the trajectory will be its closeness to a predefined point $[x_p, y_p, z_p]$, which can be written as

$$\min_{\tau \in [0,1]} || \left[x(\tau) - x_p, y(\tau) - y_p, z(\tau) - z_p\right] ||_2, \tag{5}$$

and, in the case of multiple predefined points $[x_p^j, y_p^j, z_p^j], j = 1, \ldots, P$, the closeness ($C$) to the farthest one

$$C = \max_{j=1,\ldots,P} \min_{\tau \in [0,1]} || \left[x(\tau) - x_p^j, y(\tau) - y_p^j, z(\tau) - z_p^j\right] ||_2. \tag{6}$$

As continuous control would pose too complex of a problem, we will restrict our attention to a situation where the angles θ change linearly from one setting θ^a to the next θ^b, i.e.

$$\theta(\tau) = \theta^a + \tau(\theta^b - \theta^a). \tag{7}$$

In the case where we want to have multiple points of change $\theta^0, \ldots, \theta^M$ one of the possibilities is to model it as M time intervals of length 1, i.e.:

$$\hat{\theta}(\tau) = \theta^\iota + (\tau - \iota)(\theta^\iota - \theta^{\iota+1}), \quad \text{for } \tau \in [\iota, \iota + 1], \iota = 0, \ldots, M - 1. \tag{8}$$

Even with the restriction on linear change in θ, the expressions (4) and (6) would be hard to compute analytically, which is why we resort to a discretization of τ into $M \cdot N$ evenly spaced values $[\tau_1 = 0, \ldots, \tau_{M \cdot N} = M]$ and compute:

$$[x(\tau_i), y(\tau_i), z(\tau_i)] = FK(\hat{\theta}(\tau_i)), \quad i = 1, \ldots, M \cdot N \tag{9}$$

$$\hat{L} = \sum_{i=1}^{M \cdot N - 1} || \left[x(\tau_{i+1}) - x(\tau_i), y(\tau_{i+1}) - y(\tau_i), z(\tau_{i+1}) - z(\tau_i)\right] ||_2 \tag{10}$$

$$\hat{C} = \max_{j=1,\ldots,P} \min_{\tau_i, i=1,\ldots,M \cdot N} || \left[x(\tau_i) - x_p^j, y(\tau_i) - y_p^j, z(\tau_i) - z_p^j\right] ||_2. \tag{11}$$

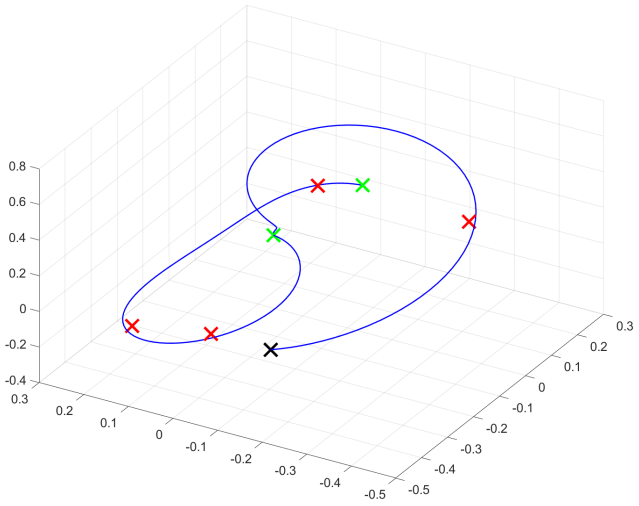


Fig. 2. Trajectory (blue line) of one solution starting of the black cross with $M = 2$ points of change (green crosses) and $P = 4$ predefined points (red crosses). Using $N = 100$, $\gamma = 100$ the objective value $f(\theta^1, \theta^2) = 4.8641$.

For a given starting position θ^0, the objective function for all the considered benchmark problems has the form:

$$f(\theta^1, \ldots, \theta^M) = \gamma \cdot \hat{L} + \hat{C}, \tag{12}$$

where the parameter $\gamma \geq 0$ lets us control the degree to which we prefer trajectories with shorter length (higher γ), or higher precision in reaching the predefined points (lower γ). The resulting optimization problem is a "simple" box-constrained one:

$$\begin{aligned} &\text{minimize } f(\theta^1, \ldots, \theta^M) \\ &\text{subject to } \theta^\iota \in [-2\pi, 2\pi]^6, \iota = 1, \ldots, M \end{aligned}$$

The resulting benchmark function can be parametrized by:

(i) the number of points of change M, which determine the dimension of the optimization problem $D = 6 \cdot M$
(ii) the number of predefined points P to which the trajectory should get close to
(iii) the coefficient γ that scales the two objectives (trajectory length and closeness to the farthest predefined point)

Figure 2 shows a solution to one problem instance with $M = 2$, $P = 4$, and $\gamma = 100$ (with $N = 100$ as the discretization constant). Figure 3 shows the sensitivity of the objective function value on the first two components of θ_1. It can readily be seen that the objective is multimodal and nonseparable, which are both desirable characteristics in benchmark functions.

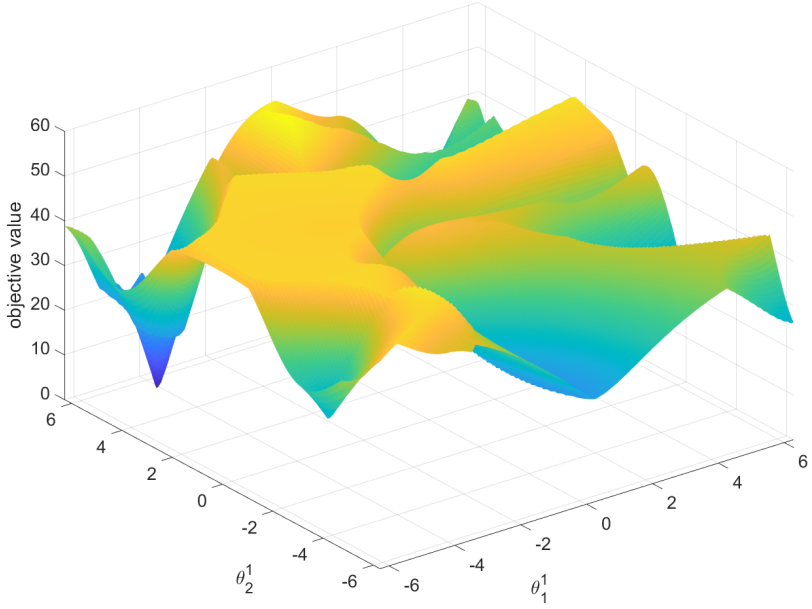


Fig. 3. Sensitivity of the objective value of the solution shown in Figure 2 on the first two components of θ^1.

4 Selected Algorithms for the Numerical Investigation

In order to showcase the capabilities of the proposed benchmark functions in differentiating various metaheuristics, we chose seven representative methods. Four of them constitute the "standard algorithms":

PSO: One of the oldest selected methods for benchmarking is Particle swarm optimization (PSO) [18]. This method was designed by simulating a simplified social model inspired by the foraging behaviour of a bird flocking or fish schooling.

DE: Another of the old methods is Differential evolution (DE) [36]. In essence, DE represents a method that aims to maintain and create new populations of candidate solutions by combining existing ones according to given rules and keeping the candidate solution with the best properties in the defined optimization problem.

CMA-ES: The last old methods selected for benchmarking is the Covariance matrix adaptation evolution strategy (CMA-ES) [11]. CMA-ES combines the use of evolution strategy and covariance matrix adaptation to apply numerical optimization.

ABC: One of the more recent metaheuristics is Artificial bee colony (ABC) [16]. This method, like PSO, is inspired by the biological behaviour of animals, in this case, based on the intelligent foraging behaviour of a bee swarm.

The other three methods constitute some of the most successful algorithm in the CEC competitions.

HSES: Hybrid Sampling Evolution Strategy (HSES) was the winner of the CEC'18 Competition. It is an evolution strategy optimization algorithm that combined CMA-ES and the univariate sampling method [42].

AGSK: Adaptive Gaining-Sharing Knowledge (AGSK) was the runner-up of the CEC'20 competition. The algorithm improved the original GSK algorithm by adding adaptive settings to its two control parameters: the knowledge factor and ratio, which control junior and senior gaining and sharing phases during the optimization process [26].

LSHADE: The last selected method is L-SHADE or Success-history based adaptive differential evolution with linear population size reduction [38]. This metaheuristic method has its basis in adaptive DE, which involves success-history-based parameter adaptation. The proposed method then provides an extension in the form of using linear population size reduction, which results in population size reduction according to a linear function.

We also decided to add to the comparison a random search (RS) method, that simply sampled (using uniform sampling) maximum available number of points and chose the best one among them.

5 Numerical Investigation

For the numerical investigation of the selected algorithms on the proposed benchmark functions we chose the problems with $M = [1, \ldots, 5]$ and $P = [3, \ldots, 6]$ (i.e., 20 possibilities). For each of the 20 problems, 10 random instances (random points in the reachable space of the robot) were generated. As the metaheuristics are stochastic, each of them was run 20 times on a given instance (to get statistically representative results). In total, each of the seven compared algorithms was run on 200 optimization problems. The maximum number of function evaluation (FES) was set to $FES = 10{,}000 \cdot D$. The benchmark functions (as well as the metaheuristic algorithms) were implemented in MATLAB and can be found at a public Zenodo[2] and GitHub repository[3]. We did not perform any parameter tuning [17].

A representative result of the computations can be seen in Figure 4, where the best solutions/trajectories (out of the 20 runs) found by the different methods for one problem instance (with $P = 4$, $M = 2$) are shown. For this instance, CMAES found the best solution, followed by PSO, ABC, and LSHADE. An interesting observation is that these solutions are qualitatively quite different. Although they all come close to the desired points, the order in which they approach differs.

[2] https://doi.org/10.5281/zenodo.7584647
[3] https://github.com/JakubKudela89/Robotics-Benchmarking

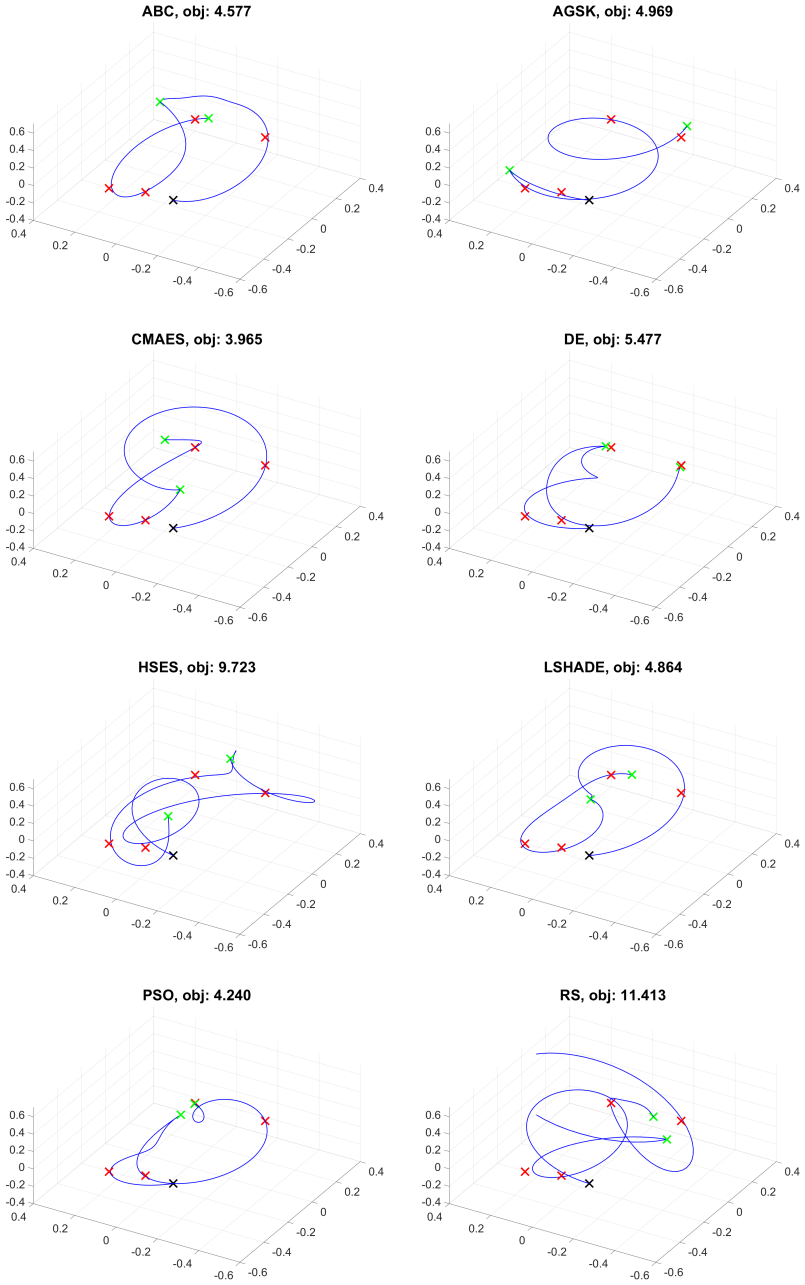


Fig. 4. Best solutions/trajectories (out of the 20 runs) found by the different methods for one problem instance (with $P = 4$, $M = 2$).

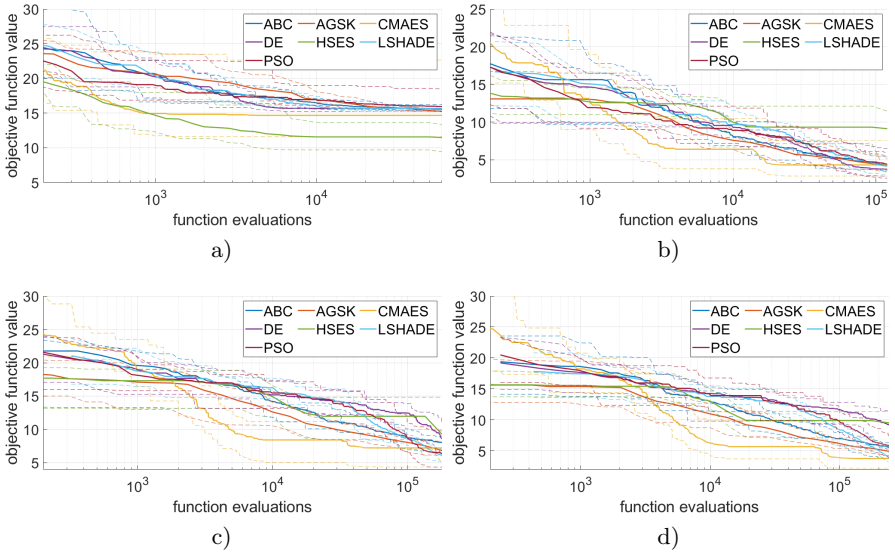


Fig. 5. Convergence plots for different instances with, a) $D = 6$ ($P = 4, M = 1$, instance 4), b) $D = 12$ ($P = 3, M = 2$, instance 1), c) $D = 18$ ($P = 5, M = 3$, instance 5), d) $D = 24$ ($P = 4, M = 4$, instance 7). Solid lines show the median values, while dashed lines show the worst and the best values (over the 20 runs).

Representative convergence plots for the considered methods are shown in Figure 5, where the solid lines show the progression of the median values, while the dashed lines show the worst and best values. These convergence plots show that there was quite a large difference between the selected methods in basically all stages of the search. It also shows that there is a substantial variance in the performance of a given algorithm within the instances.

For the ranking of the methods, we chose to focus only on the values at end of the search (after using all the FES available function evaluations). The results of the Friedman rank tests on the 20 problems (each having 10 instances on which each of the methods was run 20 times) are shown in Table 2. From these results, we can see that the proposed benchmark function were successful in differentiating the various methods, especially for problems in higher dimensions. What is interesting is the effect of M and P on the resulting ranking. As M directly influences the problem dimension, it had, unsurprisingly, a substantial effect on the ranking. Interestingly, the effect of P was also noticeable - increasing P generally meant that the (relative) performance of DE and ABC deteriorated, while the performance of HSES and PSO improved. Overall, the best-performing method was LSHADE, followed by CMAES and AGKS. The relatively bad ranking of HSES (and also AGSK) might be explained by the fact that the CEC competitions allow for much more function evaluations (which is what both HSES and AGSK were designed and tuned for).

Table 2. Mean ranks from Friedman tests for the different benchmark problems. Best three methods are highlighted in bold.

	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$D = 6$, FES = 60,000								
$P = 3, M = 1$	**3.70**	**2.36**	6.18	3.93	4.87	**2.83**	4.98	7.16
$P = 4, M = 1$	4.24	**2.87**	5.96	4.34	**2.78**	**3.24**	5.52	7.06
$P = 5, M = 1$	4.06	**2.99**	6.28	3.63	**2.97**	**3.44**	5.63	7.02
$P = 6, M = 1$	4.38	**2.98**	5.98	3.94	**2.40**	**3.48**	5.69	7.17
$D = 12$, FES = 120,000								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 2$	4.65	4.18	**3.72**	**2.20**	6.41	**2.58**	4.48	7.81
$P = 4, M = 2$	4.83	**4.08**	4.43	**2.77**	5.31	**2.43**	4.26	7.91
$P = 5, M = 2$	4.76	**3.78**	4.63	**2.78**	5.66	**2.16**	4.35	7.91
$P = 6, M = 2$	4.60	**3.76**	5.15	**2.97**	5.20	**2.44**	4.06	7.86
$D = 18$, FES = 180,000								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 3$	4.22	**3.97**	**2.73**	4.90	6.15	**2.00**	4.17	7.87
$P = 4, M = 3$	4.58	**3.50**	**3.08**	5.12	5.72	**1.70**	4.40	7.92
$P = 5, M = 3$	4.61	**3.63**	**3.23**	5.18	5.65	**2.00**	3.86	7.86
$P = 6, M = 3$	4.37	**3.38**	**3.32**	6.47	5.15	**1.75**	3.74	7.83
$D = 24$, FES = 240,000								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 4$	4.29	**3.89**	**2.13**	6.28	5.55	**1.65**	4.27	7.96
$P = 4, M = 4$	4.31	**3.66**	**2.41**	6.43	5.70	**1.51**	4.04	7.96
$P = 5, M = 4$	4.40	**3.62**	**2.71**	6.45	5.69	**1.54**	3.71	7.91
$P = 6, M = 4$	4.51	**3.66**	**2.10**	6.96	5.18	**1.89**	3.88	7.84
$D = 30$, FES = 300,000								
	ABC	AGSK	CMAES	DE	HSES	LSHADE	PSO	RS
$P = 3, M = 5$	4.26	4.36	**1.80**	6.67	5.24	**1.61**	**4.10**	7.98
$P = 4, M = 5$	4.13	**3.96**	**2.03**	6.84	5.55	**1.49**	4.05	7.98
$P = 5, M = 5$	4.28	3.85	**2.03**	6.90	5.67	**1.56**	**3.79**	7.95
$P = 6, M = 5$	4.45	**3.62**	**1.88**	7.00	5.43	**1.67**	4.03	7.93

6 Exploratory Landscape Analysis

We use ELA features to show how the proposed robotics problems compare to the BBOB and CEC 2014 benchmark suits. In order to calculate the ELA features, we used the flacco library [19]. As we are not able to supply exact function definitions (in our case, the function evaluates a simulation of the movement of the robotic arm), we chose ELA feature sets which only require samples of input and function value pairs: `ela_distr`, `ela_meta`, `disp`, `nbc`, `pca`, and `ic`. We used uniform sampling with $250D$ samples. As the dimensions of the problems in BBOB ($D = 2$, 3, 5, 10, 20, and 40), CEC 2014 ($D = 2$, 10, 30, 50, and 100),

Table 3. Minimum and Maximum values of the relevant ELA features on the three benchmark sets. Extremal values are highlighted in bold.

ELA feature	BBOB		CEC 2014		This study	
	min	max	min	max	min	max
ela_distr.skewness	**-2.97E+00**	**8.28E+00**	-6.63E-01	6.47E+00	-4.76E-01	9.54E-01
ela_distr.kurtosis	-4.94E-01	**9.67E+01**	-3.38E-01	6.50E+01	**-8.43E-01**	2.30E+00
ela_distr.number_of_peaks	**1.00E+00**	1.80E+01	**1.00E+00**	**2.60E+01**	**1.00E+00**	9.00E+00
ela_meta.lin_simple.adj_r2	1.38E-04	**1.00E+00**	**-2.30E-03**	8.23E-01	-1.11E-03	2.19E-01
ela_meta.lin_simple.intercept	**-9.17E+02**	9.62E+08	5.22E+02	**5.63E+10**	2.37E+01	4.37E+01
ela_meta.lin_w_interact.adj_r2	2.14E-04	**1.00E+00**	**-9.41E-04**	9.04E-01	5.78E-03	2.61E-01
ela_meta.quad_simple.adj_r2	3.98E-03	**1.00E+00**	**-3.61E-03**	9.88E-01	4.24E-02	2.52E-01
ela_meta.quad_w_interact.adj_r2	3.67E-05	**1.00E+00**	**-1.26E-02**	**1.00E+00**	1.64E-01	3.78E-01
disp.ratio_median_05	**7.17E-01**	1.01E+00	7.26E-01	1.02E+00	9.93E-01	**1.05E+00**
nbc.nb_fitness.cor	**-6.41E-01**	**-1.78E-01**	-6.30E-01	-1.90E-01	-5.83E-01	-4.83E-01
pca.expl_var.cor_init	8.18E-01	9.09E-01	8.18E-01	9.09E-01	9.23E-01	**9.23E-01**
pca.expl_var_PC1.cov_init	**1.07E-01**	**1.00E+00**	1.10E-01	**1.00E+00**	2.36E-01	4.36E-01

and our robotics problems ($D = 6$, 12, 18, 24, and 30) differ, we used $D = 10$ for the BBOB and CEC 2014 benchmarks, and $D = 12$ for our robotics problems, as these were the closest choices. There were 24 problems in the BBOB set, 30 problems in the CEC 2014 set, and 40 robotics problems (10 instances for $P =$ 3, 4, 5, and 6).

We followed the methodology described in [35] for the selection and visualization of the relevant ELA features. The features that produced constant results on every problem and those that produced invalid values were removed. Another set of removed features were the ones that were sensitive to scaling and shifting. The last batch of features that got removed were the highly correlated ones. The 12 features that remained, along with their maximum and minimum values on the three benchmark sets are shown in Table 3.

For further analysis, the values of the ELA features on the three benchmark sets were normalized, and we used Principal Component Analysis (PCA) to reduce the number of features even further. Figure 6 shows a representation of the 12 PCA components obtained when comparing the ELA features (normalized) calculated on the combined set of CEC 2014, BBOB, and robotics problems. Using the first 8 components explained 99.85% of the variance.

For visualizing the results, we used the t-Distributed Stochastic Neighbor Embedding (t-sne). In the this visualization, which is shown in Figure 7, benchmark problems that have similar ELA features should be shown close to each other. We can see that the proposed robotics benchmarks are not very similar to functions in either BBOB or CEC 2014 sets. They are also not very similar to each other, at least in the sense of the performed analysis.

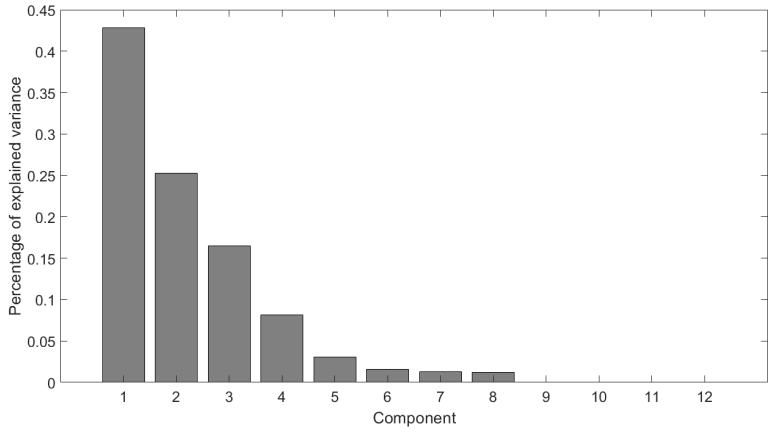


Fig. 6. The amount of explained variance per component when performing PCA on the ELA features calculated on the combined set of 2014 CEC, BBOB, and robotics problems.

7 Conclusion

In this paper, we presented a collection of parametrizable benchmark functions for comparing EC methods. The benchmark functions were a blend of real-world robotics problems in inverse kinematics and path planning. We conducted a thorough numerical investigation of the proposed benchmark problems - each of the seven selected methods for numerical comparison was used to solve 200 benchmark problems. The results of this investigation are that the proposed benchmark problems are quite difficult (multimodal and non-separable) and that they can be successfully used for differentiating and ranking various metaheuristics. The proposed methods can be applied to different types of 6-DOF robotic manipulators by simply changing the parameters of the D-H table.

There is still further work to be done. The benchmark problems will be implemented in more languages, especially the ones that are most used for the development of EC methods (e.g., Python and C++). We also plan on the integration with a proper simulator of the robotic arm in Unity. Another research directions will be in investigating more problem types (such as energy optimization, or following a fixed sequence of points), the effect of different choices for the robotic arm model, and in comparing deterministic and surrogate-based techniques.

Acknowledgements This work was supported by the IGA BUT No. FSI-S-23-8394 “Artificial intelligence methods in engineering tasks”.

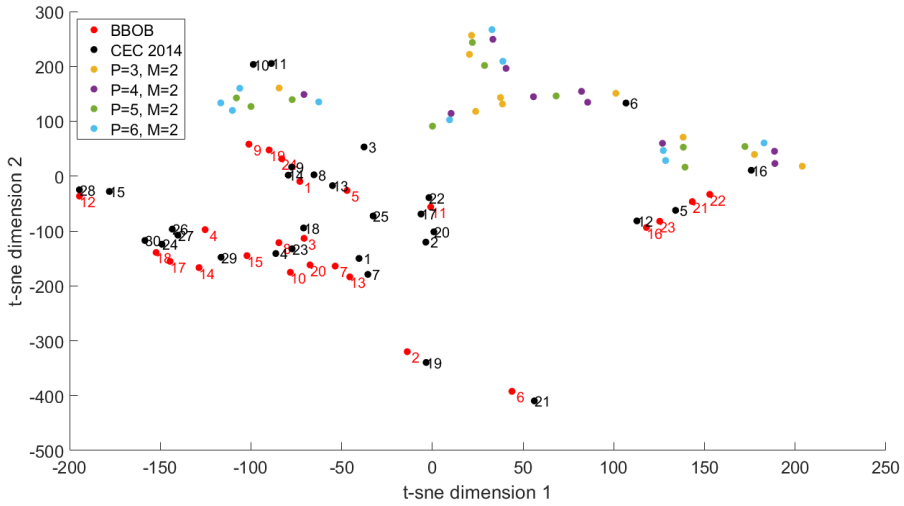


Fig. 7. The t-sne visualization of the ELA features (after normalization and using the first components from the PCA) of the three benchmark sets.

References

1. Batista, J., Souza, D., Silva, J., Ramos, K., Costa, J., dos Reis, L., Braga, A.: Trajectory planning using artificial potential fields with metaheuristics. IEEE Latin America Transactions **18**(05), 914–922 (2020)
2. Belge, E., Altan, A., Hacıoğlu, R.: Metaheuristic optimization-based path planning and tracking of quadcopter for payload hold-release mission. Electronics **11**(8), 1208 (2022)
3. Camacho Villalón, C.L., Stützle, T., Dorigo, M.: Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. In: International conference on swarm intelligence. pp. 121–133. Springer (2020)
4. Campelo, F., Aranha, C.d.C.: Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary. In: CEUR Workshop Proceedings. vol. 3007, p. 6. CEUR Workshop Proceedings (2021)
5. Cenikj, G., Lang, R.D., Engelbrecht, A.P., Doerr, C., Korošec, P., Eftimov, T.: Selector: Selecting a representative benchmark suite for reproducible statistical comparison. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 620–629. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022)
6. Croucamp, M., Grobler, J.: Metaheuristics for the robot part sequencing and allocation problem with collision avoidance. In: EPIA Conference on Artificial Intelligence. pp. 469–481. Springer (2021)
7. Denavit, J., Hartenberg, R.S.: A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. Journal of Applied Mechanics **22**(2), 215–221 (06 2021)
8. Dereli, S., Köker, R.: A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: quantum behaved particle swarm algorithm. Artificial Intelligence Review **53**(2), 949–964 (2020)

9. García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F.: Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft Computing **21**(19), 5573–5583 (2017)
10. Garden, R.W., Engelbrecht, A.P.: Analysis and classification of optimisation benchmark functions and benchmark suites. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 1641–1649. IEEE (2014)
11. Hansen, N.: The cma evolution strategy: A tutorial. arXiv preprint, arXiv:1604.00772 (2016). https://doi.org/10.48550/ARXIV.1604.00772
12. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: Coco: A platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software **36**(1), 114–144 (2021)
13. Hellwig, M., Beyer, H.G.: Benchmarking evolutionary algorithms for single objective real-valued constrained optimization–a critical review. Swarm and evolutionary computation **44**, 927–944 (2019)
14. Hulka, T., Matoušek, R., Dobrovský, L., Dosoudilová, M., Nolle, L.: Optimization of snake-like robot locomotion using ga: Serpenoid design. Mendel Journal **26**(1), 1–6 (2020)
15. Kanagaraj, G., Masthan, S.S., Vincent, F.Y.: Meta-heuristics based inverse kinematics of robot manipulator's path tracking capability under joint limits. Mendel Journal **28**(1), 41–54 (2022)
16. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Tech. Rep. TR06, Erciyes University (2005)
17. Kazikova, A., Pluhacek, M., Senkerik, R.: Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison? In: Mendel. vol. 26, pp. 9–16 (2020)
18. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks. vol. 4, pp. 1942–1948 vol.4 (1995)
19. Kerschke, P., Trautmann, H.: Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. In: Applications in Statistical Computing, pp. 93–123. Springer (2019)
20. Khan, A.H., Li, S., Chen, D., Liao, L.: Tracking control of redundant mobile manipulator: An rnn based metaheuristic approach. Neurocomputing **400**, 272–284 (2020)
21. Kudela, J.: A critical problem in benchmarking and analysis of evolutionary computation methods. Nature Machine Intelligence **4**, 1238–1245 (2022)
22. Kudela, J., Matousek, R.: New benchmark functions for single-objective optimization based on a zigzag pattern. IEEE Access **10**, 8262–8278 (2022)
23. Kudela, J., Matousek, R.: Recent advances and applications of surrogate models for finite element method computations: A review. Soft Computing pp. 1–25 (2022)
24. Kumar, R., Singh, L., Tiwari, R.: Comparison of two meta–heuristic algorithms for path planning in robotics. In: 2020 International Conference on Contemporary Computing and Applications (IC3A). pp. 159–162. IEEE (2020)
25. Mersmann, O., Preuss, M., Trautmann, H.: Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In: International Conference on Parallel Problem Solving from Nature. pp. 73–82. Springer (2010)
26. Mohamed, A.W., Hadi, A.A., Mohamed, A.K., Awad, N.H.: Evaluating the performance of adaptive gainingsharing knowledge based algorithm on cec 2020 benchmark problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020)

27. Niu, P., Niu, S., Chang, L., et al.: The defect of the grey wolf optimization algorithm and its verification method. Knowledge-Based Systems **171**, 37–43 (2019)
28. Nonoyama, K., Liu, Z., Fujiwara, T., Alam, M.M., Nishi, T.: Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization. Energies **15**(6), 2074 (2022)
29. Parak, R., Matousek, R.: Comparison of multiple reinforcement learning and deep reinforcement learning methods for the task aimed at achieving the goal. Mendel Journal **27**(1), 1–8 (2021)
30. Pattnaik, S., Mishra, D., Panda, S.: A comparative study of meta-heuristics for local path planning of a mobile robot. Engineering Optimization **54**(1), 134–152 (2022)
31. Piotrowski, A.P.: Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions. Information Sciences **297**, 191–201 (2015)
32. Qadir, Z., Zafar, M.H., Moosavi, S.K.R., Le, K.N., Mahmud, M.P.: Autonomous uav path-planning optimization using metaheuristic approach for predisaster assessment. IEEE Internet of Things Journal **9**(14), 12505–12514 (2021)
33. Serrano-Pérez, O., Villarreal-Cervantes, M.G., González-Robles, J.C., Rodríguez-Molina, A.: Meta-heuristic algorithms for the control tuning of omnidirectional mobile robots. Engineering Optimization (2019)
34. Siciliano, B., Khatib, O.: ed. Springer Handbook of Robotics. Berlin, Heidelberg: Springer-Verlag (2016)
35. Škvorc, U., Eftimov, T., Korošec, P.: Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. Applied Soft Computing **90**, 106138 (2020)
36. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**(4), 341–359 (1997)
37. Tanabe, R.: Benchmarking feature-based algorithm selection systems for black-box numerical optimization. IEEE Transactions on Evolutionary Computation (2022)
38. Tanabe, R., Fukunaga, A.S.: Improving the search performance of shade using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC). pp. 1658–1665. IEEE (2014)
39. Tzanetos, A., Dounias, G.: Nature inspired optimization algorithms or simply variations of metaheuristics? Artificial Intelligence Review **54**(3), 1841–1862 (2021)
40. Yadav, V., Botchway, R.K., Senkerik, R., Oplatkova, Z.K.: Robotic automation of software testing from a machine learning viewpoint. Mendel Journal **27**(2), 68–73 (2021)
41. Yin, S., Luo, Q., Zhou, G., Zhou, Y., Zhu, B.: An equilibrium optimizer slime mould algorithm for inverse kinematics of the 7-dof robotic manipulator. Scientific Reports **12**(1), 1–28 (2022)
42. Zhang, G., Shi, Y.: Hybrid sampling evolution strategy for solving single objective bound constrained problems. In: 2018 IEEE Congress on Evolutionary Computation (CEC). pp. 1–7. IEEE (2018)

A6

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

New Benchmark Functions for Single-Objective Optimization Based on a Zigzag Pattern

JAKUB KUDELA[1] and RADOMIL MATOUSEK[1]
[1]Institute of Automation and Computer Science, Brno University of Technology, Czech Republic

Corresponding authors: Jakub Kudela (e-mail: Jakub.Kudela@vutbr.cz), Radomil Matousek (email: matousek@fme.vutbr.cz).

This work was supported by IGA Brno University of Technology: No. FSI-S-20-6538.

ABSTRACT Benchmarking plays a crucial role in both development of new optimization methods, and in conducting proper comparisons between already existing methods, particularly in the field of evolutionary computation. In this paper, we develop new benchmark functions for bound-constrained single-objective optimization that are based on a zigzag function. The proposed zigzag function has three parameters that control its behaviour and difficulty of the resulting problems. Utilizing the zigzag function, we introduce four new functions and conduct extensive computational experiments to evaluate their performance as benchmarks. The experiments comprise of using the newly proposed functions in 100 different parameter settings for the comparison of eight different optimization algorithms, which are a mix of canonical methods and best performing methods from the Congress on Evolutionary Computation competitions. Using the results from the computational comparison, we choose some of the parametrization of the newly proposed functions to devise an ambiguous benchmark set in which each of the problems introduces a statistically significant ranking among the algorithms, but the ranking for the entire set is ambiguous with no clear dominating relationship between the algorithms. We also conduct an exploratory landscape analysis of the newly proposed benchmark functions and compare the results with the benchmark functions used in the Black-Box-Optimization-Benchmarking suite. The results suggest that the new benchmark functions are well suited for algorithmic comparisons.

INDEX TERMS numerical optimization, benchmarking, single objective problems, exploratory landscape analysis

I. INTRODUCTION

BENCHMARKING plays a pivotal part in the development of new algorithms as well as in the comparison and assessment of contemporary algorithmic ideas [1]. For instance, one of the most powerful classes of algorithms for solving black-box optimization problems are Evolutionary Algorithms (EAs). An issue with EAs is that there are only a few theoretical performance results, which means that their performance comparisons and development rely heavily on benchmarking. These benchmarking experiments are constructed for performance comparisons on given classes of problems and should support the selection of appropriate algorithms for a given real-world application [2]. Another utilization of benchmarking is in the qualification of the theoretical predictions of the behaviour of algorithms [3].

There are, currently, two main lines of development in benchmarking for EAs, the IEEE Congress on Evolutionary Computation (CEC) competitions [4], that have been running in the current form since 2013, and the the Black-Box-Optimization-Benchmarking [5] workshops (BBOB workshops), which started in 2009. Both of these venues provide a variety of materials that describe the various benchmarks, and provide code for the best-performing algorithms in a given year. Also, the BBOB benchmarks are run on a platform called the Comparing Continuous Optimizer (COCO) benchmark suite [6]. The COCO suite serves as a place for comparing various algorithms for unconstrained continuous numerical optimization. On the other hand, the CEC competitions that are organized every year aim at comparing state-of-the-art stochastic search methods on test environments that are specifically crafted each year. These benchmark functions

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

$$z(x,k,m,\lambda)=\begin{cases}1-m+\frac{m}{\lambda}(|x|/k-\lfloor|x|/k\rfloor), & \text{if } |x|/k-\lfloor|x|/k\rfloor\leq\lambda,\\ 1-m+\frac{m}{1-\lambda}(1-|x|/k+\lfloor|x|/k\rfloor), & \text{otherwise,}\end{cases} \tag{1}$$

$$\begin{aligned}\phi_1(x,k,m,\lambda)&=3{\cdot}10^{-9}|(x-40)(x-185)x(x+50)(x+180)|z(x,k,m,\lambda)+10|\sin(0.1x)|\\ \phi_2(x,k,m,\lambda)&=\phi_1(\phi_1(x,k,m,\lambda),k,m,\lambda)\\ \phi_3(x,k,m,\lambda)&=3|\ln(1000|x|+1)|z(x,k,m,\lambda)+30-30|\cos(\frac{x}{10\pi})|\\ \phi_4(x,k,m,\lambda)&=\phi_3(\phi_3(x,k,m,\lambda),k,m,\lambda)\end{aligned} \tag{2}$$

are developed from a set of commonly used benchmark functions, such as the Ackley's function, Griewank's function, Rastrigin's function, Rosenbrock's function, Schwefel's function, and many others [7]. However, recent research has found that the BBOB and CEC benchmark suites have a poor coverage of the corresponding problem space [8, 9, 10], have a very similar distribution in certain landscape analysis measures [11], and are highly correlated from the perspective of the performance of different algorithms [12, 13, 14].

In this paper, we introduce four new benchmark functions for bound-constrained single-objective optimization that are based on a zigzag pattern, and are non-differentiable and highly multimodal. Inspired by recently proposed tunable benchmark functions for combinatorial problems [15], the newly proposed functions have three parameters that can change their behaviour and difficulty. We conduct extensive numerical experiments to investigate how the different parametrizations work as benchmarks and show that they can be successfully used for algorithmic comparisons.

The rest of the paper is organized as follows. Section II introduces the individual components of the zigzag function and the newly proposed benchmark functions, and provides insight into their construction. In Section III we report on extensive computational experiments where we compare 100 different parametrizations of the newly proposed benchmark functions on eight EAs that are a mix of canonical methods as well as the best performing methods from the CEC competitions. In Section IV we devise an ambiguous benchmark set that is based on selected parametrizations of the benchmark functions. The individual problems in this set displayed good capabilities in giving unambiguous rankings for the considered algorithms, but taken as a whole set produced no clear dominating relationship between the performances of all the considered algorithms. In Section V, we conduct exploratory landscape analysis of the proposed functions and show how they compare to the functions from the BBOB suite. The conclusions and future research directions are outlined in Section VI.

II. NEW BENCHMARK FUNCTIONS

The newly proposed benchmark functions are constructed in the following way. First, a so-called "zigzag" $z(x,k,m,\lambda)$ (or triangular wave) function is constructed based on the formula in (1), where $\lfloor\cdot\rfloor$ is the floor operator. Apart from the point $x\in\mathcal{R}$ at which it should be evaluated, the zigzag function also contains three other parameters: $k>0$ that controls its period, $m\in[0,1]$ that controls its amplitude, and $\lambda\in(0,1)$ that controls the location of its local minima. The effect of the individual parameters on the shape of the zigzag function can be seen in Fig. 1. For $m=0$ the zigzag function is identically equal to 1 for any point x (regardless of the values of the other two parameters).

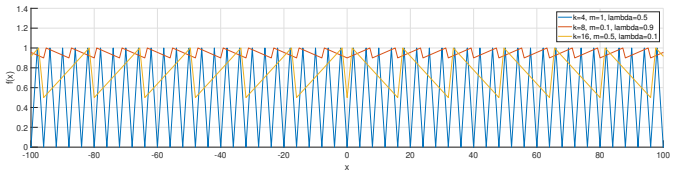


FIGURE 1. Zigzag function for different values of the parameters.

In the following step, we construct four basic benchmark functions (F1-F4) that combined the zigzag function with different multimodal functions, and that inherit the parameters from the zigzag function. Their construction starts with four 1-D functions $\phi_1,\ldots,\phi_4$, which are formulated in (2).

The common theme of all of these functions is that they are bounded on the interval [-200, 200] by a maximum value of 200 (which allows for composing the functions with themselves without running to numerical difficulties), and there is a single global minimum located at 0, with function value 0. Although the optimization will take place only on the interval [-100, 100] we will use a shift vector to change the placement of the optimal point (hence, the need for the functions to be "well-behaved" on [-200, 200]).

To obtain the benchmark functions for a dimension D, we use a simple sum of the functions ϕ for the individual components, but modify the inputs by a shift vector $\mathbf{s}\in[-100,100]^D$ and a rotation/scaling matrix $\mathbf{M}$:

$$\begin{aligned}f_j(\mathbf{x},k,m,\lambda)&=\sum_{i=1}^{D}\phi_j(x_i,k,m,\lambda) && j=1,\ldots,4\\ \mathrm{F}j(\mathbf{x},k,m,\lambda)&=f_j(\mathbf{M}_j(\mathbf{x}-\mathbf{s}_j),k,m,\lambda) && j=1,\ldots,4\end{aligned}$$

The matrices M are constructed in the exact same manner as in [16]. Both of these components are an integral part of good benchmark functions [17], as they create some additional difficulty for the various optimization algorithms [4, 18]. The shapes of the functions F1-F4 in one dimension

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

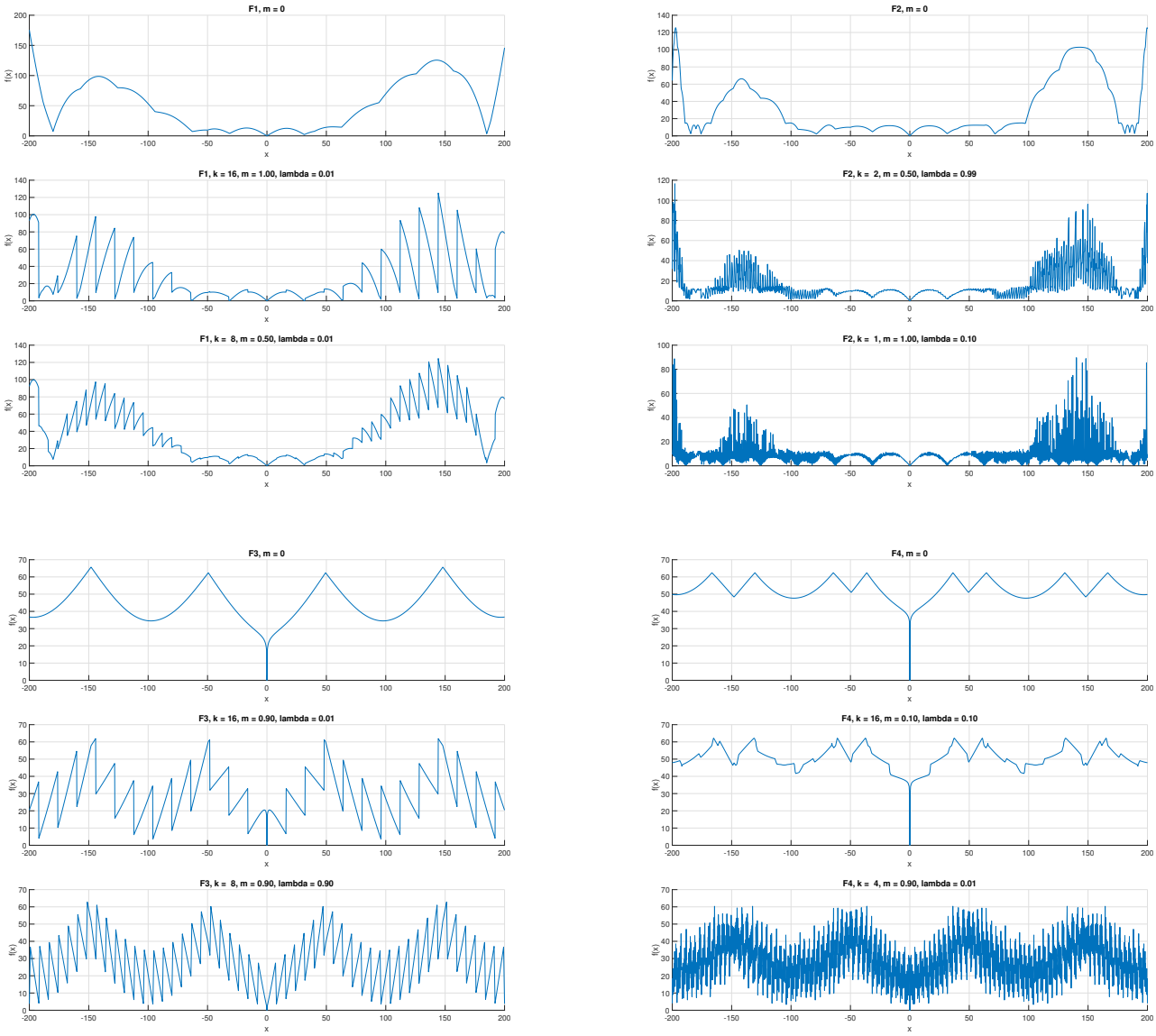


FIGURE 2. Shape of the benchmark functions F1-F4 for different values of the parameters.

(without any shift or scaling) for selected values of the parameters can be seen in Fig. 2, while their corresponding contour and surface plots for 2-D (this time, with shift and rotation/scaling) can be found in Appendix A. The individual parameters of the zigzag function, apart for their primary role in defining the shape of the zigzag, also serve secondary roles. The parameter k can be though of as a "ruggedness" parameter, that makes the zigzag more frequent. The parameter λ can be seen as a "deception" parameter, as low values of λ skew the function in such a way that its derivatives (if existing) point predominantly away from the global optimum (in the case of the 1-D examples in Fig. 2, away from $x = 0$). Higher values of the parameter m result in a "deeper" local optima. By setting these parameters to different values, we should be able to increase/decrease the difficulty of the resulting optimization problems [15], and to bring out the advantages and the disadvantages of different optimization methods.

III. COMPARISON ON ALGORITHMS

A. ALGORITHM SELECTION AND EXPERIMENTAL SETTING

In this section, we investigate the behaviour of selected EAs on the newly proposed benchmark functions for different values of the input parameters. We chose two main categories of algorithms for this investigation: a) generally known and used EAs, b) best performing algorithms from the CEC Competitions on Bound Constraint Single Objective Optimization [4]. The other unifying theme of the chosen algorithms was that they had their code publicly available for the MATLAB language. The eight selected algorithms were:

- Adaptive Gaining-Sharing Knowledge (AGSK) – the runner-up of the CEC'20 competition. The algorithm improves and extends upon original GSK [19] algorithm by adding adaptive settings to two main control parameters: the knowledge factor and the knowl-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

edge ratio, that control junior and senior gaining and sharing phases among the solutions during the optimization process [20].

- Covariance Matrix Adaptation Evolution Strategy (CMAES) – a canonical algorithm that adapts the covariance matrix of a mutation distribution [21].
- Differential Evolution (DE) – a canonical algorithm, one of the most widely used ones for continuous optimization [22]. The implementation and parameter settings for DE used in this paper was the one that was shipped alongside the benchmark suite for the CEC'21 Competition.
- Hybrid Sampling Evolution Strategy (HSES) – winner of the CEC'18 Competition. An evolution strategy optimization algorithm which combined CMAES and the univariate sampling method [23].
- Improved Multi-operator Differential Evolution (IMODE) – winner of the CEC'20 Competition. This algorithm employs multiple differential evolution operators and a sequential quadratic programming local search technique for accelerating its convergence [24].
- Linear Population Size Reduction SHADE (LSHADE) – one of the most popular variants of adaptive DE, that was used as a basis for many of the best performing algorithms in the CEC Competitions in past few years [25].
- Multiple Adaptation DE Strategy (MadDE) – one of the best performing algorithms in the CEC'21 competition. This is another modification of the DE algorithm that uses a multiple adaptation strategy for its search mechanisms and for adapting its control parameters at the same time [26].
- Particle Swarm Optimization (PSO) – another canonical algorithm that simulates swarm behavior of various social animals such as the fish schooling or bird flocking [27]. The implementation and parameter settings for PSO used in this paper was the one that was shipped alongside the benchmark suite for the CEC'20 Competition.

To investigate the impact of the different values of the parameters of the benchmark functions, we chose 100 parameter settings (for all functions F1-F4) for computational evaluations, which are reported in Table 1. For the individual computations, we used rules similar to the CEC'21 competition: the eight algorithms were evaluated on the four benchmark functions with 100 different parameter settings with $D = \{5, 10, 15\}$ dimensions, and a search space of $[-100, 100]^D$. The maximum number of function evaluations were set to 50,000, 200,000, 500,000 fitness function evaluations for problems with $D = \{5, 10, 15\}$, respectively. All algorithms were run 30 times to get representative results. For every run, if the objective function value of the resulting solution was less than or equal to 1E–8, it was considered as zero. For all algorithms, we used the same parameter settings that was used in the corresponding CEC competition [28].

TABLE 1. Different parameter settings used in the computations.

ID	k	m	λ	ID	k	m	λ	ID	k	m	λ	ID	k	m	λ
1	1	0.10	0.01	26	1	0.50	0.01	51	1	0.90	0.01	76	1	1.00	0.01
2	2	0.10	0.01	27	2	0.50	0.01	52	2	0.90	0.01	77	2	1.00	0.01
3	4	0.10	0.01	28	4	0.50	0.01	53	4	0.90	0.01	78	4	1.00	0.01
4	8	0.10	0.01	29	8	0.50	0.01	54	8	0.90	0.01	79	8	1.00	0.01
5	16	0.10	0.01	30	16	0.50	0.01	55	16	0.90	0.01	80	16	1.00	0.01
6	1	0.10	0.10	31	1	0.50	0.10	56	1	0.90	0.10	81	1	1.00	0.10
7	2	0.10	0.10	32	2	0.50	0.10	57	2	0.90	0.10	82	2	1.00	0.10
8	4	0.10	0.10	33	4	0.50	0.10	58	4	0.90	0.10	83	4	1.00	0.10
9	8	0.10	0.10	34	8	0.50	0.10	59	8	0.90	0.10	84	8	1.00	0.10
10	16	0.10	0.10	35	16	0.50	0.10	60	16	0.90	0.10	85	16	1.00	0.10
11	1	0.10	0.50	36	1	0.50	0.50	61	1	0.90	0.50	86	1	1.00	0.50
12	2	0.10	0.50	37	2	0.50	0.50	62	2	0.90	0.50	87	2	1.00	0.50
13	4	0.10	0.50	38	4	0.50	0.50	63	4	0.90	0.50	88	4	1.00	0.50
14	8	0.10	0.50	39	8	0.50	0.50	64	8	0.90	0.50	89	8	1.00	0.50
15	16	0.10	0.50	40	16	0.50	0.50	65	16	0.90	0.50	90	16	1.00	0.50
16	1	0.10	0.90	41	1	0.50	0.90	66	1	0.90	0.90	91	1	1.00	0.90
17	2	0.10	0.90	42	2	0.50	0.90	67	2	0.90	0.90	92	2	1.00	0.90
18	4	0.10	0.90	43	4	0.50	0.90	68	4	0.90	0.90	93	4	1.00	0.90
19	8	0.10	0.90	44	8	0.50	0.90	69	8	0.90	0.90	94	8	1.00	0.90
20	16	0.10	0.90	45	16	0.50	0.90	70	16	0.90	0.90	95	16	1.00	0.90
21	1	0.10	0.99	46	1	0.50	0.99	71	1	0.90	0.99	96	1	1.00	0.99
22	2	0.10	0.99	47	2	0.50	0.99	72	2	0.90	0.99	97	2	1.00	0.99
23	4	0.10	0.99	48	4	0.50	0.99	73	4	0.90	0.99	98	4	1.00	0.99
24	8	0.10	0.99	49	8	0.50	0.99	74	8	0.90	0.99	99	8	1.00	0.99
25	16	0.10	0.99	50	16	0.50	0.99	75	16	0.90	0.99	100	16	1.00	0.99

All algorithms were run in a MATLAB R2020b, on a PC with 3.2 GHz Core I5 processor, 16 GB RAM, and Windows 10. The particular values of matrices $\mathbf{M}$ and shifts $\mathbf{s}$, as well as the implementation of the benchmark functions in MATLAB, can be found in the authors github[1].

B. RESULTS

First, we investigate the "difficulty" of the four benchmark functions (with the 100 different parameter setting) by counting the number of unsolved instances, i.e., the number of times out of the 30 runs that any of the eight algorithms was not able to reach function value of at least 1E–8 within the specified number of function evaluations. These results are summarized in Fig. 3. The first thing to notice is that the basic forms of the benchmark functions with no zigzag (with $m = 0$) were not "impossible" to solve for at least some of the algorithms, regardless of the dimension. However, the instances were also not "too easy" so that the algorithms could reliably find the optimum – this, in our opinion, makes these benchmark functions worth investigating. Unexpectedly, increasing the dimension of the problem led to increase in the number of unsolved instances. The parameter m displayed the highest effect on the difficulty of the resulting problem – larger values of m (meaning a more pronounced zigzag) generated problems that were harder to solve. Similarly, lower values of the parameter $\lambda \in \{0.01, 0.1\}$ generally produced more difficult instances. The effects of different values of the parameter k were more subtle and function-dependent. For F1 and F2, increasing the period of the zigzag together with low values of λ produced the most difficult instances. For F3, increasing k exhibited a decrease in difficulty. Finally, for F4, the effect of k was mixed, with some instances being more or less difficult with increasing k.

More interesting result can be found in comparing the individual algorithms on the different benchmark functions. For this comparison, we used the IOHprofiler [29], which is a

[1]https://github.com/JakubKudela89/Zigzag

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

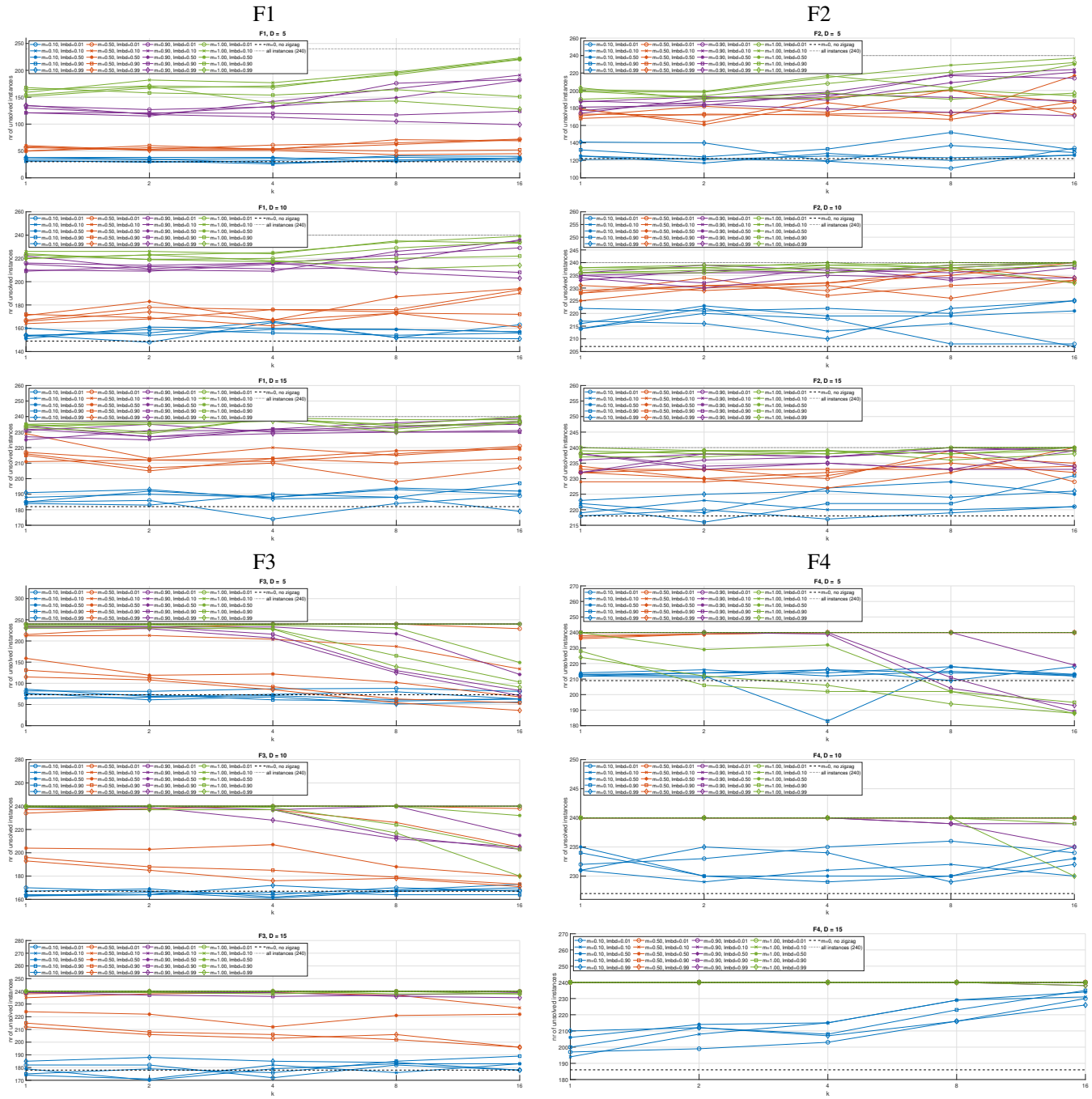


FIGURE 3. Number of unsolved instances for benchmark function F1-F4 for different values of parameters and dimensions.

benchmarking and profiling tool for optimization heuristics. Within the IOHprofiler, we chose the comparison based on a fixed-budget (defined by the maximum number of function evaluations) and compared the algorithms on each benchmark function and each dimension separately for all the 100 parameter settings. The ranking of the algorithms was obtained by using a Deep Statistical Comparison (DSC) analysis [30], which works by comparing distributions of the obtained solutions instead of using descriptive statistics. For the visualization of the results, we employed the PerformViz approach [31], which shows the algorithms that are most suited for a given problem and similarities among both problems and algorithms, within a single plot. The threshold for statistical significance was set to 0.05 for all comparisons. The results of the comparisons are shown in Fig. 4 and Fig. 5. For a particular problem, if a rectangle, that represents a certain problem (horizontal dimension) and a certain algorithm (vertical dimension), is bluer then the algorithm is better for the given problem (and, the redder it is, the worse is the algorithm). In the horizontal dimension, if the color of the rectangles remains the same, it signals that the ranking of the given algorithm remains the same across

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

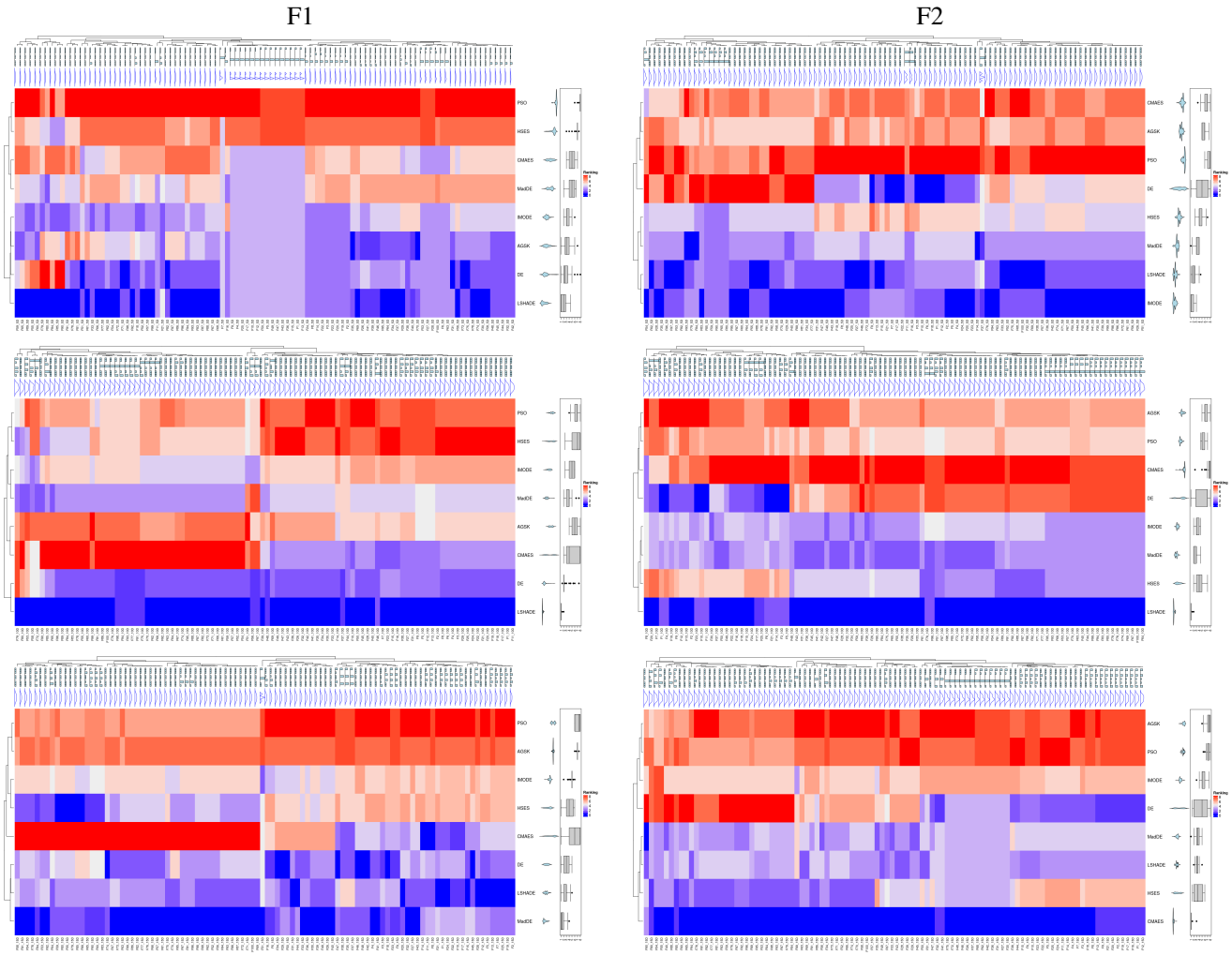


FIGURE 4. Comparison of the algorithms on different parameters settings for benchmark functions F1-F2, $D = \{5, 10, 15\}$.

different problems. On the other hand, if the rectangles have the same color in the vertical dimension, it means that there was no significant difference between the performance of the individual algorithms (and, probably, that this particular parameter settings is not very well suited to serve as a benchmark function).

For F1 in $D = 5$, there is a large section of parameter values that produced problems on which there was no significant difference among most of the algorithms. These were the parameter settings with either low $m = 0.1$, or lower $m \leq 0.5$ and higher $\lambda \geq 0.9$. The rest of the problems produced relatively stable rankings, where LSHADE performed the best, followed by DE and AGSK. The worst performing methods were PSO and HSES. Interestingly, there was a group of parameter settings for which otherwise well-performing DE struggled (and was ranked as the worst) – these settings corresponded to high values of $m \geq 0.9$ and $k \geq 8$ and lower values of $\lambda \leq 0.5$. For $D = 10$, there were only a few instances on which the algorithms were not well comparable. There were, however, two large groups of parameter settings on which two seemingly similar algorithms, CMAES and HSES, performed very differently. For the first group, that can be characterized by lower values of $m \leq 0.5$, CMAES ranked as a third best, while HSES ranked as the worst. For the second group ($m \geq 0.9$), the ranking of HSES became much better (between second and fifth), while CMAES became the worst ranking one. Overall, LSHADE remained the best performing algorithm, followed by DE, while PSO again ranked among the worst. A very similar pattern can be observed also for $D = 15$, where rankings of HSES and CMAES depended heavily on the value of m. A notable difference is that MadDE became the best performing algorithm, followed by LSHADE and DE.

For F2 in $D = 5$, there were no notable parameter settings that would result in an ambiguous ranking of the algorithms. The DE method experienced the largest variability in performance, while it ranked among the worst for problems with high values of $k \geq 8$ and $m \geq 0.9$, it was the best ranked algorithm for problems with low $m = 0.1$, and mediocre for the other problem settings. This time, the best performing algorithm overall was IMODE, followed by LSHADE and MadDE, while PSO performed the worst. For $D = 10$, there

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

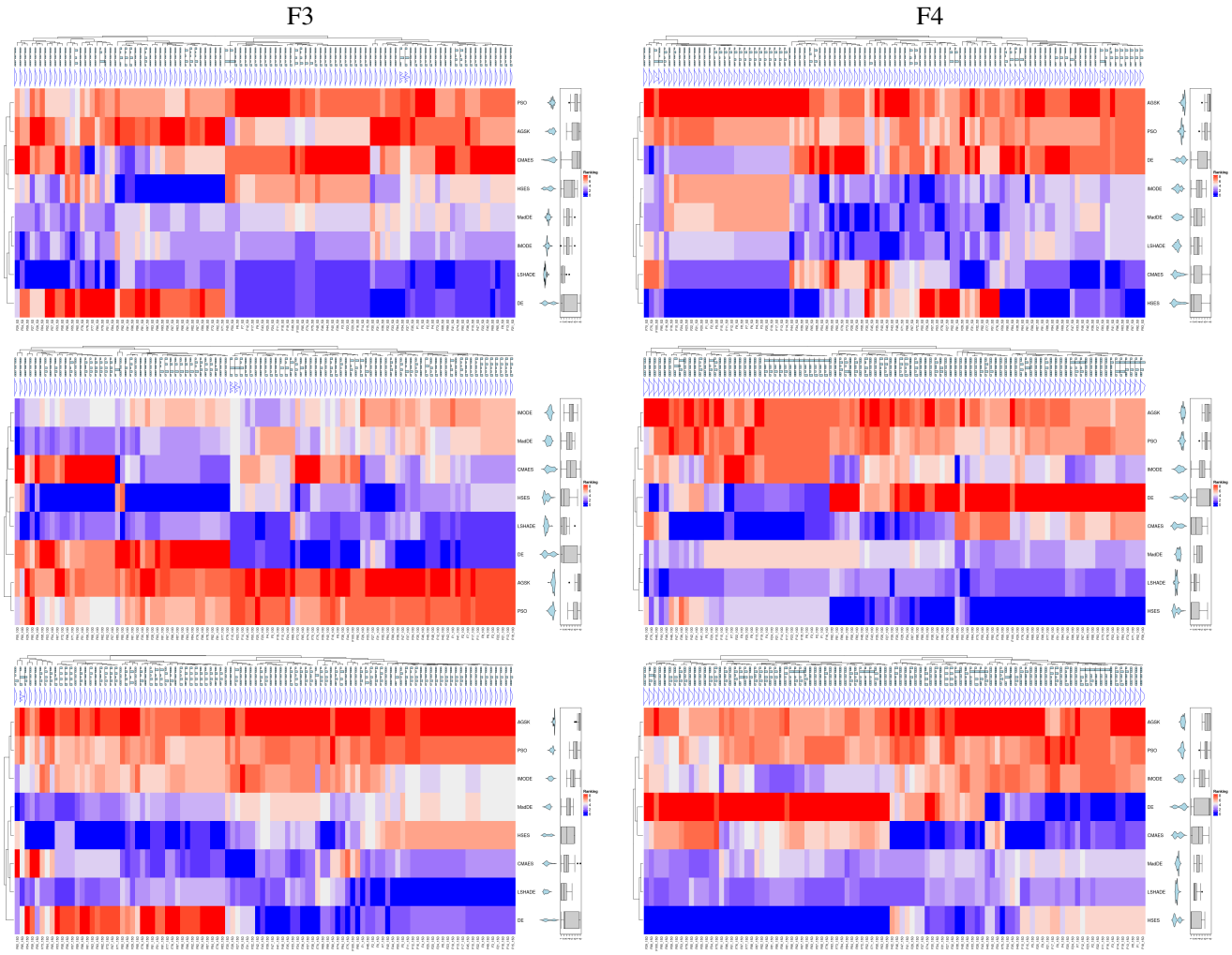


FIGURE 5. Comparison of the algorithms on different parameters settings for benchmark functions F3-F4, $D = \{5, 10, 15\}$.

were again instances for which the ranking was ambiguous, but this time they corresponded to the "difficult" instances with larger values of $m > 0.9$ and $k \geq 8$, and low $\lambda \leq 0.1$. Similarly to the previous case, DE performed well on the instances with low values of $m = 0.1$, but its performance degraded for larger m. The best ranked algorithm was LSHADE, followed by MadDE, IMODE and HSES. This time, the worst performing methods were CMAES and AGSK. For $D = 15$, the situation changed quite drastically. The parameter settings resulting in ambiguous ranking were the ones with low values of $k \leq 4$. Once again, the ranking of DE depended heavily on m. By far the best performing method was the canonical CMAES, while the worst were AGSK and PSO.

For F3 there were no parameter settings that would result in an ambiguous ranking in any of the investigated dimensions. For $D = 5$, the methods that were the most impacted by changes in parameters were HSES and CMAES, which both benefited from low values of $k \leq 4$ and high $m \geq 0.9$, and DE, which struggled for higher values of $m \geq 0.9$ but otherwise ranked among the best. A peculiar behaviour can be seen for AGSK, that performed relatively well for problems with either low values of $m = 0.1$, or for $m = 0.5$ with larger values of $\lambda \geq 0.9$, or for $m \geq 0.9$ but only with $\lambda = 0.9$ and $k = 16$. Otherwise, AGSK performed consistently among the worst, along with CMAES and PSO, while the best ranking algorithm was LSHADE, with HSES and DE also ranking high for some settings. For $D = 10$, the issues that DE had with high values of m remained. Meanwhile, HSES performed very well for problems with $m \geq 0.9$, while LSHADE and DE dominated the rest. Interestingly, CMAES also performed relatively well, apart from the instances with $m = 0.9$. The consistently worst performing algorithms were AGSK and PSO. A very similar patter also appeared in $D = 15$, where HSES and CMAES dominated the instances with $m \geq 0.9$, while LSHADE and DE dominated the rest.

For F4, there were a few instances that resulted in an ambiguous ranking, but we did not notice any clear pattern. For $D = 5$, there was a large variability in the best performing algorithm, as all algorithms apart from PSO and AGSK were ranked as the best one for some parameter settings. HSES

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

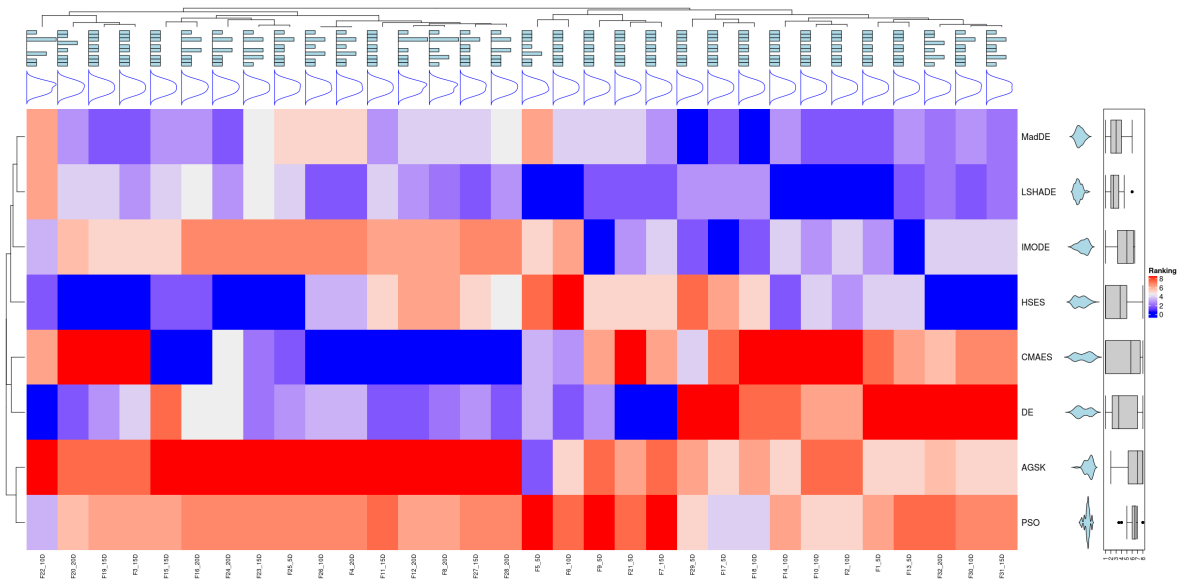


FIGURE 6. Comparison of the algorithms on the ambiguous benchmark set.

struggled with problems with both $m \geq 0.5$ and $\lambda \leq 0.1$, but otherwise performed the best. CMAES had a similar dependence on the parameters, apart from a few instances with lower values of $k \leq 2$, where it performed very well. LSHADE, IMODE, and MadDE performed consistently well on all instances, while DE performed well on instances with either low values of $m = 0.1$, or on instances that had simultaneously high values of $m \geq 0.9$, $\lambda \geq 0.9$, and $k = 16$. For $D = 10$, HSES and CMAES ranked the best in the majority of instances. What separated them was that CMAES was better ranked for problems with either $m = 0.1$, or with $m = 0.2$ together with $k \leq 2$, while HSES dominated the rest. IMODE showed mediocre performance, apart from instances with $m = 0.1$, where it performed significantly worse, and where, coincidentally, DE performed much better than on the other instances. LSHADE ranked consistently at the top, while PSO, AGSK and DE (for $m \geq 0.5$) ranked at the bottom. Finally, for $D = 15$, the best performing algorithm again depended mainly on m. For $m \geq 0.9$, HSES was by far the best one (while DE was the worst), while for $m \leq 0.5$, DE and CMAES were ranked as the best. Both LSHADE and MadDE performed consistently well across all instances, while PSO and AGSK were the worst.

Generally speaking, the performance of some algorithms, such as DE, HSES, and CMAES, exhibited high dependence on the parametrization of the benchmark functions, while others showed only little dependence. Among the three parameters, m had the most pronounced effect on the rankings, while λ and k showed significant effect mainly for particular combinations off all three parameters (in a similar fashion to the investigation of the number of unsolved instances). On the other hand, most of the parametrizations displayed good ability to differentiate between the various algorithms. The algorithm that performed consistently well across the different benchmarks was LSHADE – it is, then, unsurprising that it was chosen as the basis of many of the best performing algorithms for the CEC competitions in recent years. Naturally, studying the performance of the algorithms on problems in even higher dimensions, or with modified number of function evaluations, could bring more varied results.

IV. CREATING AMBIGUOUS BENCHMARK SET

In this section, we select a few parametrizations of the proposed benchmark functions to create a benchmark set. This selection had several goals. The first was to choose parametrizations that result in a clear ranking (for the specific problem). The second was to choose parametrizations across the range of possible values of the parameters. And the third was to select the parametrizations in such a way that the rankings for the resulting benchmark set are ambiguous. The last goal corresponds to having a benchmark set that is comprised of functions on which different algorithms perform differently, without a clear favourite. This selection was done by carefully examining the results of the comparison of the algorithms and choosing, for each benchmark function, two parametrization, that together resulted in the ambiguous benchmark set. We also decided to include additional dimension $D = 20$, with a maximum of 1,000,000 function evaluations for these parametrizations (this was excluded from the previous comparisons because of excessive computational requirements). This resulted in the creation of a benchmark set with 32 instances (four functions F1-F4, two parametrizations each, four different dimension), that are summarized in Table 2. The 1-D plots of the chosen functions can be seen in Fig. 2, while their 2-D contour and surface plots can be seen in the Appendix A.

Detailed results of the computations, including convergence analysis and detailed statistics, can be found in the Appendix B. The results of the comparison of the eight

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

TABLE 2. Parametrizations for the ambiguous benchmark set.

ID	function	D	k	m	λ	ID	function	D	k	m	λ
1	F1	5	16	1	0.01	17	F3	5	16	0.9	0.01
2	F1	10	16	1	0.01	18	F3	10	16	0.9	0.01
3	F1	15	16	1	0.01	19	F3	15	16	0.9	0.01
4	F1	20	16	1	0.01	20	F3	20	16	0.9	0.01
5	F1	5	8	0.5	0.01	21	F3	5	8	0.9	0.9
6	F1	10	8	0.5	0.01	22	F3	10	8	0.9	0.9
7	F1	15	8	0.5	0.01	23	F3	15	8	0.9	0.9
8	F1	20	8	0.5	0.01	24	F3	20	8	0.9	0.9
9	F2	5	2	0.5	0.99	25	F4	5	16	0.1	0.1
10	F2	10	2	0.5	0.99	26	F4	10	16	0.1	0.1
11	F2	15	2	0.5	0.99	27	F4	15	16	0.1	0.1
12	F2	20	2	0.5	0.99	28	F4	20	16	0.1	0.1
13	F2	5	1	1	0.1	29	F4	5	4	0.9	0.01
14	F2	10	1	1	0.1	30	F4	10	4	0.9	0.01
15	F2	15	1	1	0.1	31	F4	15	4	0.9	0.01
16	F2	20	1	1	0.1	32	F4	20	4	0.9	0.01

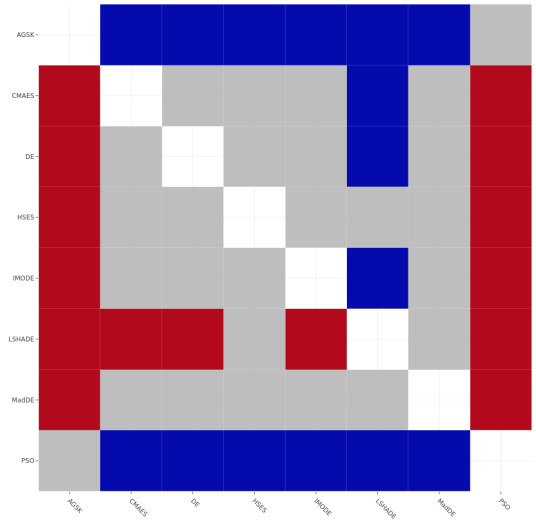


FIGURE 7. Results of the post-hoc test. For a given row: red cell – the algorithm in the row performs significantly better than the algorithm in the column; blue cell – performs significantly worse; grey cell – the ranking is ambiguous.

chosen algorithms on the ambiguous benchmark set are presented in Fig. 6. From this comparison, it can be seen that most of the algorithms, apart from PSO and AGSK, were best-ranked for at least one of the problems in the set. Also, for at least one problem, all algorithms placed in the bottom half of the rankings. This signals that the chosen parametrizations cover a wide range of the single-objective optimization problem space (measured by the performance of different algorithms). However, it is clear from the results that some of the selected algorithms performed consistently better than others. To quantify this relationship, we used the post-hoc test from the aforementioned DSC framework, the results of which are shown in Fig. 7. For a given row (one selected algorithm), if a column has a red color, it means that the selected algorithm performs significantly better (with a statistical significance threshold of 0.05) on the benchmark set that the algorithm in the column. If it has a blue color, it means that it performs significantly worse, and, if it has a grey color, the ranking is ambiguous. From Fig. 7, we can see that PSO and AGSK are ranked as the worst (but with unclear comparison between the two), and that LSHADE is ranked better that AGSK, CMAES, DE, IMODE, and PSO (but not better that HSES and MadDE). Other than that, the algorithms are mutually incomparable on our ambiguous benchmark set.

V. EXPLORATORY LANDSCAPE ANALYSIS

To better explore the problem space that is covered by the different parametrizations of the proposed benchmark functions, we used the method of exploratory landscape analysis (ELA) [32], within the flacco library [33]. We focus only on the landscape features that have been found to be invariant under shift and scale [10] and the ones that provide expressive results [8]. The 20 selected ELA measures were the following:

- cm_angle.angle.mean
- ela_distr.skewness
- ela_distr.kurtosis
- ela_distr.number_of_peaks
- ela_meta.lin_simple.adj_r2
- ela_meta.lin_simple.intercept
- ela_meta.lin_simple.coef.min
- ela_meta.quad_w_interact.adj_r2
- ela_meta.quad_simple.adj_r2
- ela_meta.lin_w_interact.adj_r2
- disp.ratio_mean_02
- disp.ratio_median_25
- nbc.nb_fitness.cor
- pca.expl_var_PC1.cov_init
- pca.expl_var.cov_init
- pca.expl_var.cor_init
- ic.eps.ratio
- ic.eps.s
- ic.h.max
- ic.m0

We chose only a single dimension $D = 10$ as the representative and used 8,000 function evaluations for robust computation of the features [8]. We performed additional analysis to find the effect of the parameters on the resulting ELA features and found that the main difference stemmed from varying the parameter m, which was the same parameter that had the highest impact on the number of unsolved instances and on the ranking of the selected optimization algorithms. The results of the ELA are summarized in Fig. 8, where the plotted lines are grouped based on the value of the parameter m. Although the results show high diversity in the values of the individual measures, it is hard to judge how diverse these values are compared to the commonly used benchmark functions. This is why we provide a comparison of all the parametrizations of F1-F4 and the parametrizations chosen for the ambiguous benchmark set with the benchmark functions from the BBOB suite [5], that are shown in Fig. 9. In general, the values of the ELA measures covered by the different parametrizations of the F1-F4 functions were

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

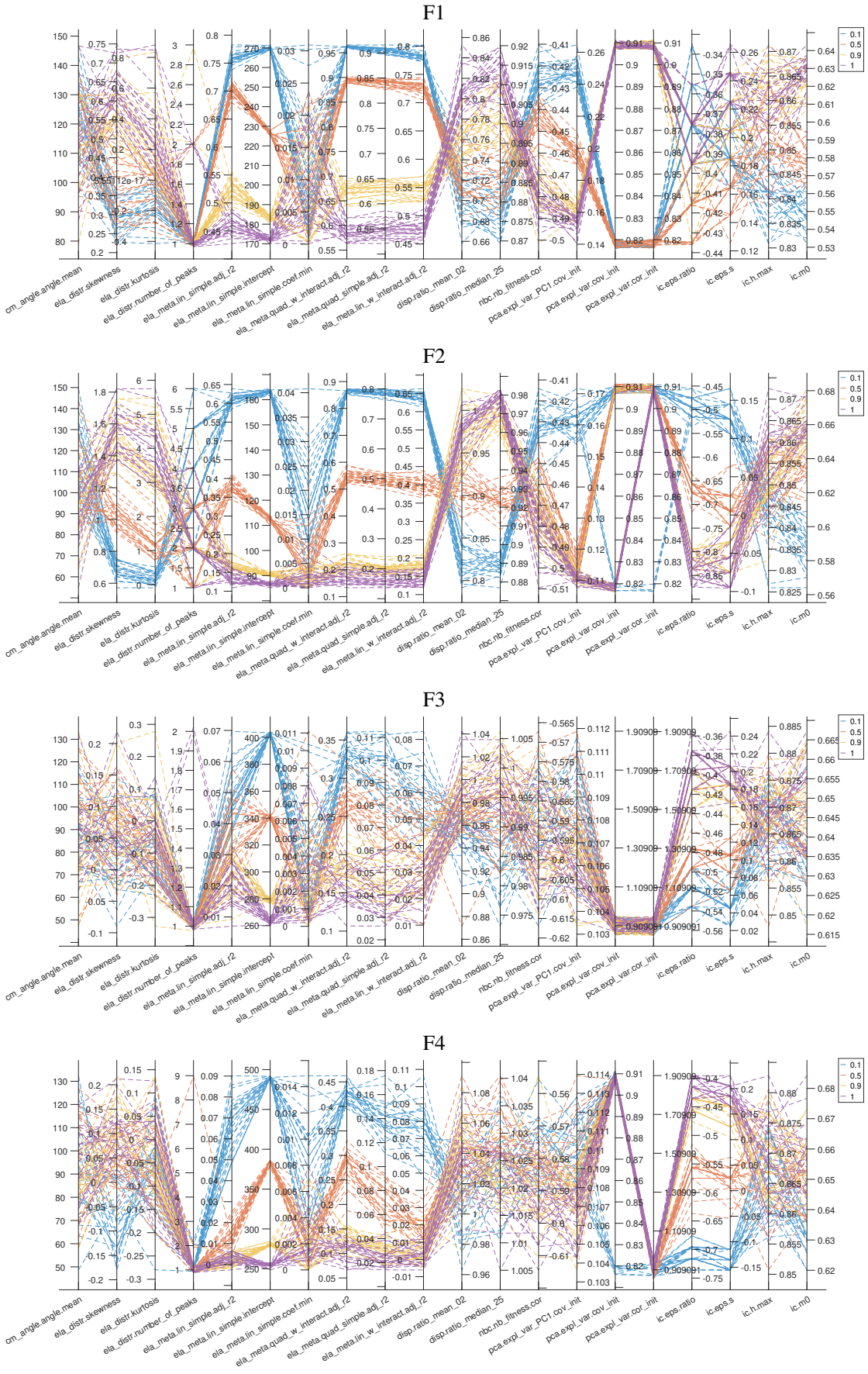


FIGURE 8. Parallel plots of the results of ELA for different parametrizations of F1-F4, $D = 10$, grouped by the value of the parameter m.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

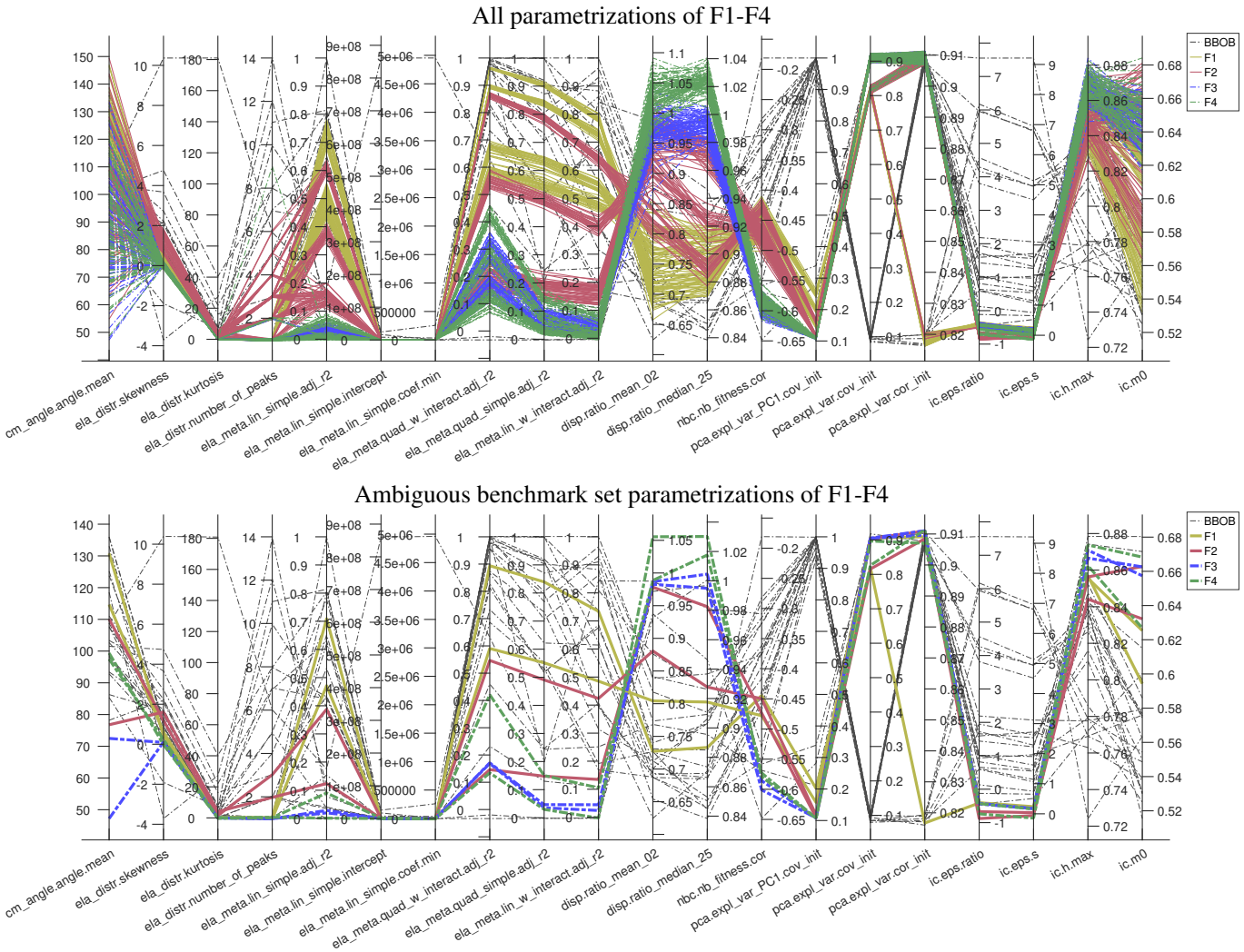


FIGURE 9. Comparison of ELA measures between different parametrizations of F1-F4 and BBOB, $D = 10$.

comparable to the ones covered by the BBOB benchmark functions. Notably, some F1-F4 parametrizations covered values that were outside the range of the BBOB functions for measures cm_angle.angle.mean, disp.ratio_mean_02, disp.ratio_median_25, pca.expl_var_PC1.cov_init, ic.eps.s, and ic.eps.ratio. When looking at the parametrizations of F1-F4 for the ambiguous benchmark set, we can see that particularly the ELA measures of the chosen F3 and F4 parametrizations fall into places where there are only a few BBOB counterparts. This suggests that either including these benchmark functions into the BBOB suite, or creating a new benchmark suite that includes them, could provide a better coverage of the problem space of bound-constrained single-objective optimization problems.

VI. CONCLUSION

In this paper, we introduced four new benchmark functions for bound constrained single objective optimization that are based on a highly parametrizable zigzag pattern. The construction of these benchmark functions was straightforward enough to allow for a broad range of extensions, variations, and further study. The extensive computational experiments showed that different parametrizations of these functions can serve as good benchmarks, and we were able to create an ambiguous benchmark set on which the ranking of the selected algorithms was unclear, although the ranking on the individual instances was clear-cut. These results, together with the conducted exploratory landscape analysis, suggest that the new benchmark functions are well suited for algorithmic comparisons. Future research directions will encompass comparing even broader selection of algorithms, particularly ones that were not very well represented by the studied methods (such as swarm intelligence algorithms). Another interesting path will be in investigating the role of bias in the optimal function value (in our experiments, all optima had the function value of 0). Finally, we plan on developing new multimodal benchmark functions [34] by utilizing the presented framework.

.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

APPENDIX A CONTOUR AND SURFACE PLOTS OF SELECTED FUNCTIONS

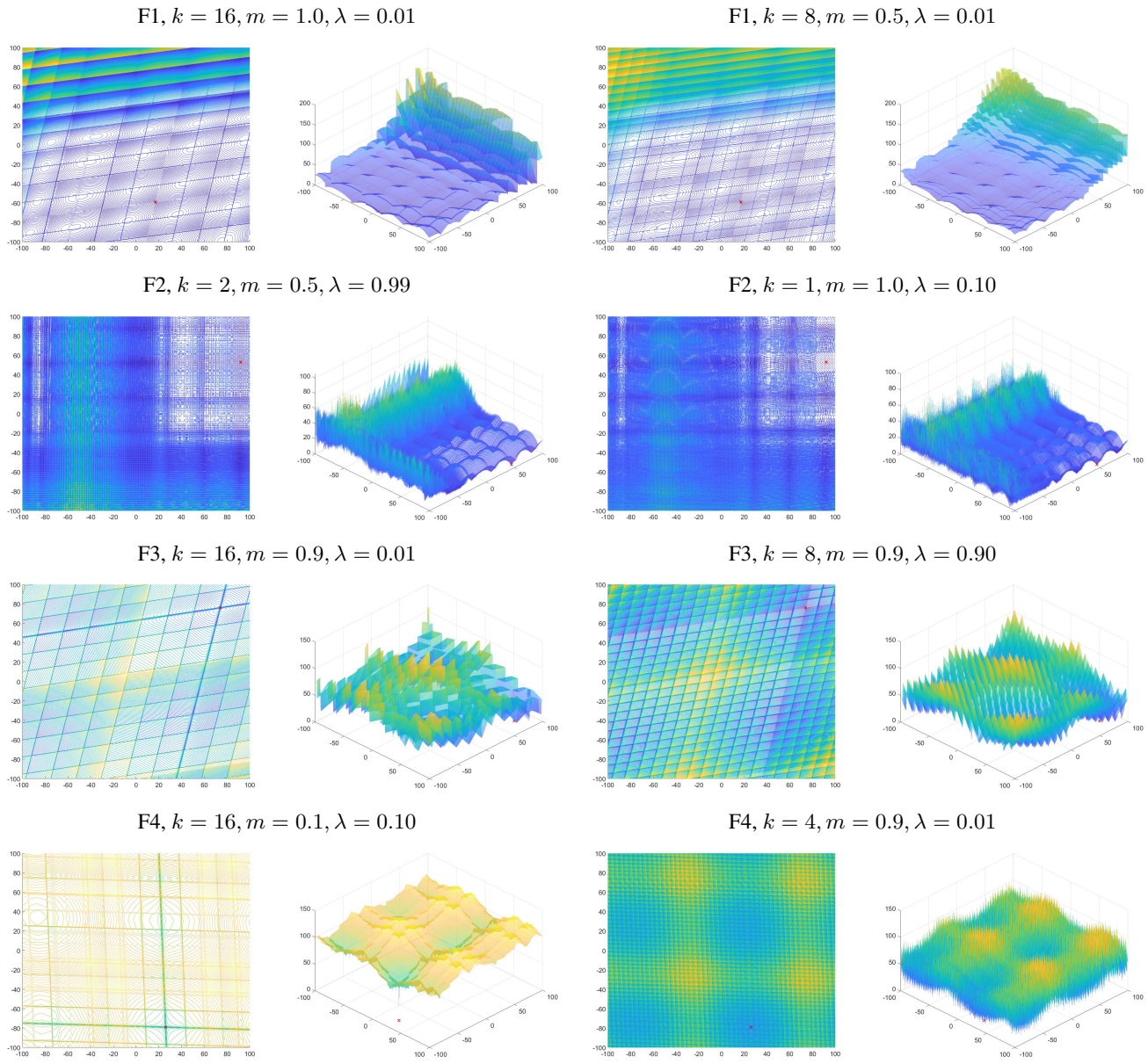


FIGURE 10. Contour and surface plots of the parametrization of F1-F4 used for the ambiguous benchmark set.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

APPENDIX B CONVERGENCE PLOTS AND DETAILED STATISTICS OF THE SELECTED ALGORITHMS ON THE AMBIGUOUS BENCHMARK SET

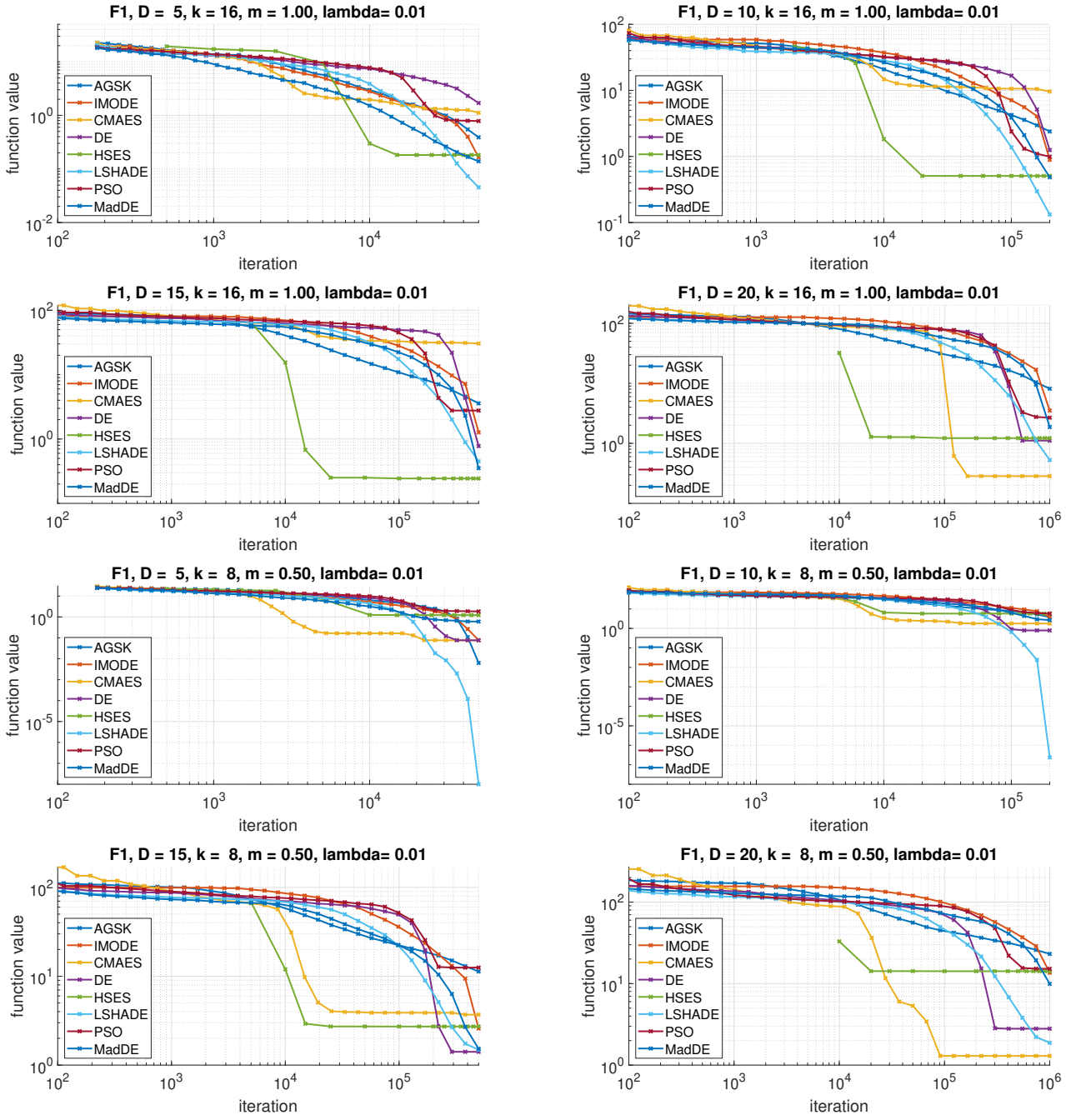


FIGURE 11. Convergence plots for F1 parametrizations (ID 1-8).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

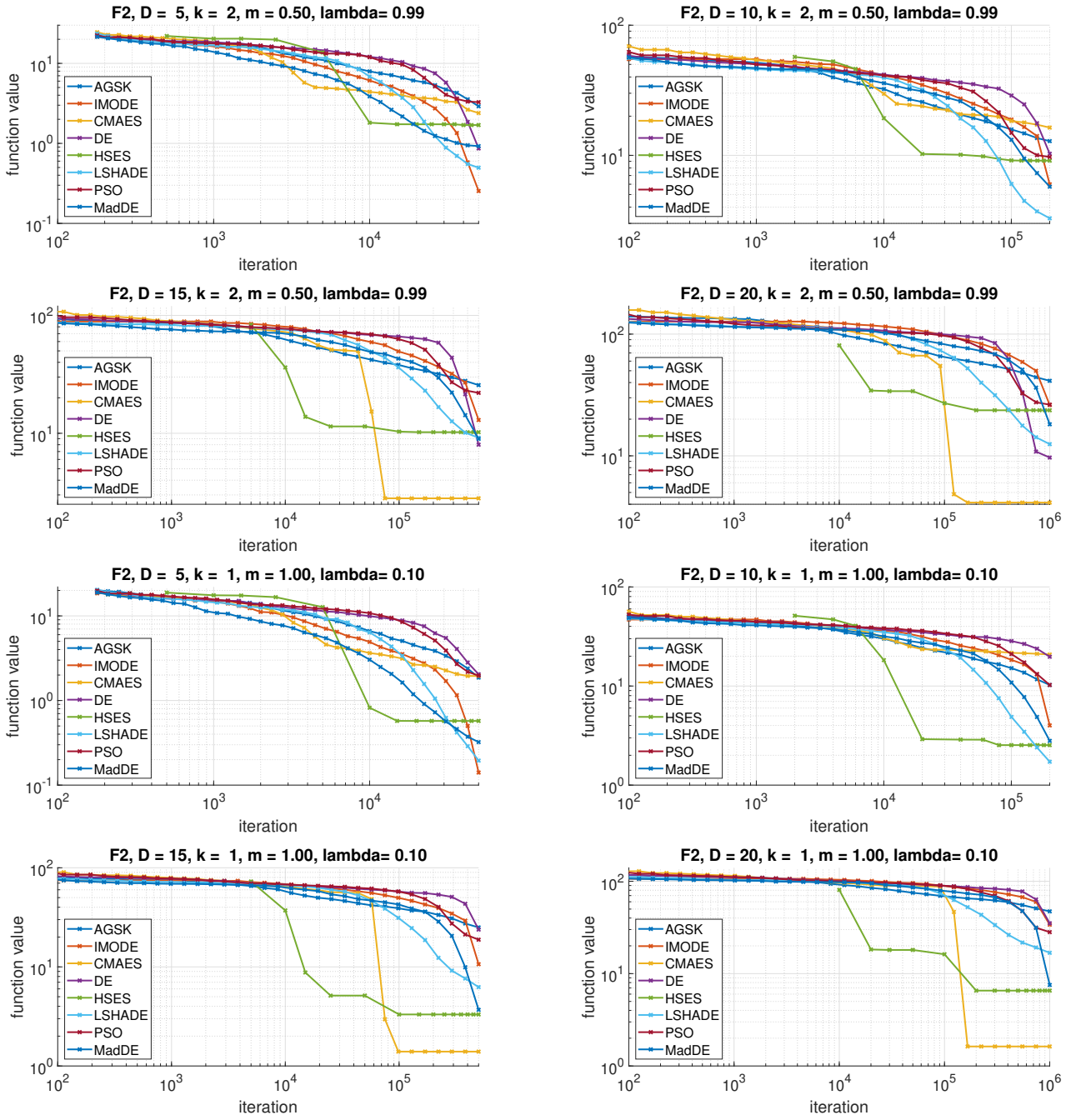


FIGURE 12. Convergence plots for F2 parametrizations (ID 9-16).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

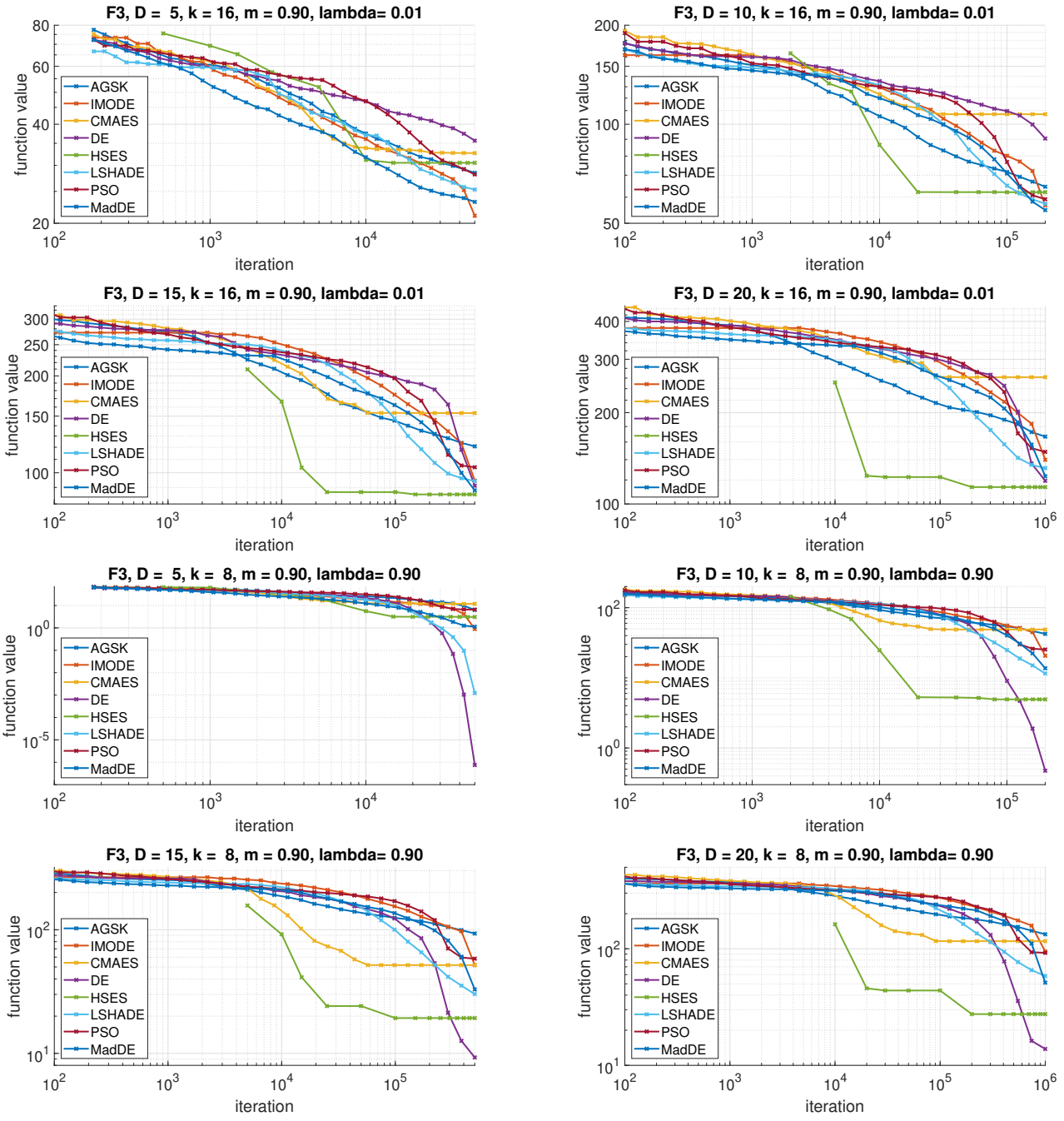


FIGURE 13. Convergence plots for F3 parametrizations (ID 17-24).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

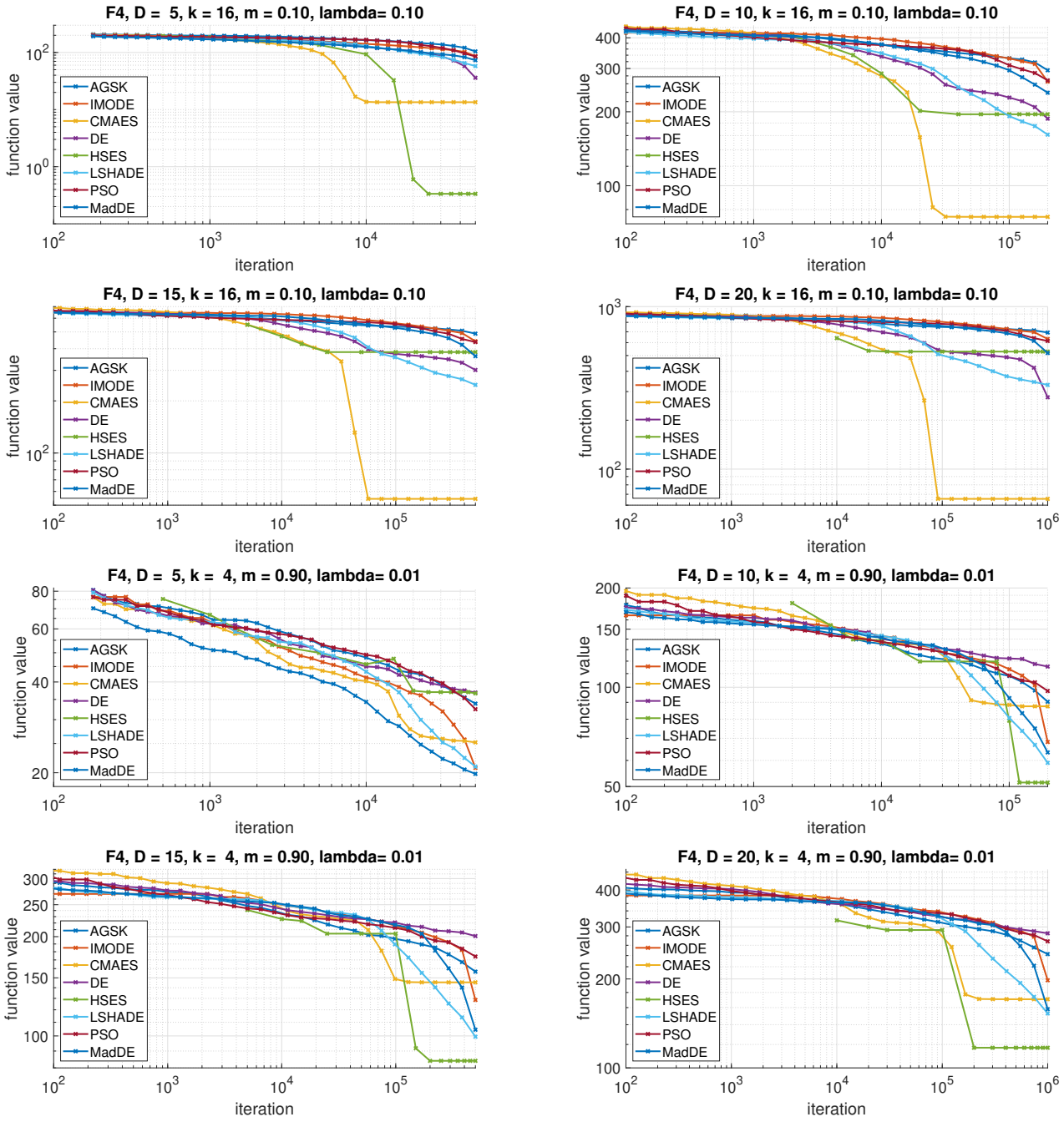


FIGURE 14. Convergence plots for F3 parametrizations (ID 25-32).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

TABLE 3. Detailed statistics of the 30 runs of the selected algorithms on the ambiguous benchmark set.

ID		AGSK	CMAES	DE	HSES	IMODE	LSHADE	MadDE	PSO
	min	0.079	0	4.7E-05	0.148	0	0	0	0.148
	median	0.382	0.079	1.810	0.148	0.192	0.016	0.098	0.690
1	mean	0.389	1.120	1.700	0.182	0.165	0.045	0.139	0.783
	max	0.629	6.674	2.978	0.533	0.375	0.227	0.377	1.822
	std	0.123	1.748	0.877	0.075	0.109	0.056	0.100	0.505
	min	1.257	0	0	0.306	0.237	0.002	0.080	0.306
	median	2.407	6.692	0.341	0.464	0.805	0.139	0.475	0.751
2	mean	2.389	9.639	1.250	0.507	0.889	0.132	0.485	0.993
	max	3.332	26.041	19.712	0.918	1.753	0.268	0.997	2.598
	std	0.486	9.996	3.616	0.119	0.407	0.087	0.163	0.606
	min	2.534	0	0.158	0.079	0.621	0.254	0.158	1.066
	median	3.643	39.538	0.582	0.237	1.228	0.482	0.380	2.587
3	mean	3.584	30.451	0.777	0.244	1.270	0.450	0.353	2.770
	max	4.641	49.380	2.816	0.464	2.680	0.649	0.533	6.899
	std	0.548	19.186	0.620	0.092	0.465	0.088	0.094	1.282
	min	6.394	0	0.227	0.909	1.835	0.200	1.136	0.988
	median	7.750	0.237	1.105	1.210	3.430	0.526	1.882	2.292
4	mean	8.132	0.284	1.110	1.221	3.512	0.527	1.871	2.679
	max	10.757	0.858	2.062	1.660	5.721	0.745	2.437	6.379
	std	1.144	0.187	0.482	0.156	1.049	0.122	0.277	1.340
	min	0	0	0	0	0	0	0	0
	median	0	0	0	2.159	0	0	0	1.148
5	mean	0.006	0.077	0.077	1.228	0.077	0	0.603	1.838
	max	0.189	1.148	1.148	3.308	1.148	0	2.296	7.452
	std	0.035	0.291	0.291	1.202	0.291	0	0.763	1.850
	min	0	0	0	4.456	2.159	0	1.148	2.319
	median	4.202	0	0	5.604	3.950	0	2.296	5.380
6	mean	3.818	1.779	0.790	5.796	3.870	2.4E-07	2.675	5.777
	max	6.568	13.668	4.456	7.901	5.798	6.1E-06	4.593	10.896
	std	1.645	3.678	1.252	0.803	1.212	1.1E-06	0.995	2.437
	min	3.445	0	0	1.148	5.0E-06	0.003	0	3.444
	median	11.628	1.148	1.148	2.296	2.296	1.164	1.148	13.075
7	mean	11.302	3.697	1.412	2.717	2.586	1.487	1.526	12.509
	max	14.614	35.888	5.741	5.741	4.456	3.455	3.445	20.485
	std	2.284	8.487	1.515	0.976	1.054	0.942	0.747	3.695
	min	15.209	0	0	10.934	8.912	0.002	2.296	8.912
	median	22.856	1.148	2.296	14.379	13.231	2.306	10.497	15.633
8	mean	23.114	1.297	2.797	14.180	13.583	1.877	9.913	15.192
	max	29.106	4.456	8.912	17.550	19.534	4.466	12.744	23.990
	std	3.309	1.258	2.218	1.731	2.636	1.186	2.273	4.007
	min	0.351	0	0	0	0	0	0	0
	median	2.876	1.886	6.1E-05	2.173	0	0.003	0.807	3.365
9	mean	2.908	2.396	0.865	1.695	0.254	0.497	0.925	3.274
	max	5.554	9.906	8.781	2.881	1.961	1.995	1.886	9.186
	std	1.355	2.719	2.037	0.882	0.659	0.832	0.942	2.023
	min	7.886	0	0	5.664	1.886	0.002	1.605	1.605
	median	12.487	21.548	3.966	9.116	6.085	3.715	5.757	9.497
10	mean	12.865	16.344	10.284	9.108	5.989	3.283	5.745	9.717
	max	17.286	37.279	28.568	14.860	9.641	6.727	9.033	17.369
	std	2.040	13.365	10.621	2.360	2.127	1.710	1.909	3.814
	min	19.195	0	1.886	6.519	8.714	4.348	5.185	12.838
	median	25.994	2.173	5.953	10.248	12.758	9.845	8.918	20.023
11	mean	25.629	2.807	8.006	10.219	12.961	9.169	8.968	22.022
	max	32.357	8.416	34.796	14.028	17.685	13.648	12.851	43.048
	std	3.492	2.678	6.217	1.690	2.222	2.560	1.848	7.163
	min	30.890	0	3.210	15.752	19.499	6.852	10.187	15.232
	median	41.565	3.327	8.426	23.186	26.333	12.499	18.971	25.418
12	mean	41.338	4.129	9.680	23.674	26.167	12.471	18.161	26.322
	max	50.257	10.308	21.103	30.395	36.428	18.686	24.438	43.032
	std	4.713	3.007	4.534	3.463	4.169	2.740	3.826	7.099
	min	0	0	5.6E-08	0	0	0	0	0.199
	median	1.842	0.637	1.917	0.637	0	0.014	0.469	1.620
13	mean	1.867	1.950	2.042	0.573	0.141	0.195	0.322	1.969
	max	3.528	8.261	6.146	1.274	0.470	0.695	0.640	6.072
	std	1.029	2.449	2.140	0.257	0.219	0.268	0.255	1.371
	min	5.961	0	4.0E-04	1.474	1.274	0.485	0	0.199
	median	10.284	26.534	22.105	2.549	4.134	1.661	2.551	9.113
14	mean	10.248	20.786	19.752	2.534	4.004	1.722	2.797	10.249
	max	14.187	30.922	30.011	3.385	7.986	3.130	9.179	26.090
	std	2.151	11.369	8.803	0.519	1.881	0.723	1.762	6.107
	min	17.448	0	0.637	2.274	3.728	2.806	1.408	3.382
	median	25.269	1.274	12.385	3.186	9.151	5.617	3.156	17.325
15	mean	25.000	1.396	23.789	3.311	10.639	6.248	3.687	18.817
	max	32.461	3.018	53.380	4.460	20.560	13.618	8.005	57.909
	std	3.840	0.807	22.321	0.605	4.464	3.025	1.503	12.130
	min	37.808	0.199	5.2E-07	4.993	14.945	4.194	1.878	16.267
	median	47.335	1.374	20.635	6.416	35.142	16.225	5.674	24.982
16	mean	47.233	1.626	35.135	6.549	33.864	16.808	7.541	28.150
	max	55.853	3.186	81.701	9.323	51.526	33.009	23.803	78.122
	std	4.450	0.889	33.104	1.095	9.050	7.673	4.969	12.278

ID		AGSK	CMAES	DE	HSES	IMODE	LSHADE	MadDE	PSO
	min	24.497	20.920	29.237	27.239	13.817	21.721	18.211	20.209
	median	28.568	30.398	35.520	30.398	20.559	25.862	22.916	28.718
17	mean	28.468	32.684	35.650	30.532	21.078	25.313	23.226	28.147
	max	31.597	51.998	41.569	33.557	27.393	27.175	27.811	34.698
	std	1.978	8.097	3.210	1.310	3.177	1.664	2.499	3.940
	min	55.092	54.477	60.309	57.636	50.114	51.925	46.091	52.587
	median	64.854	126.734	94.517	62.089	56.761	57.102	54.738	57.903
18	mean	64.553	107.265	90.495	62.136	56.653	57.323	54.832	59.166
	max	72.476	156.234	116.656	63.955	63.579	61.972	61.251	67.202
	std	3.653	38.127	15.928	1.761	3.729	2.392	2.863	4.328
	min	104.287	81.724	77.044	81.716	84.433	87.715	76.012	85.912
	median	120.277	95.646	92.118	84.875	95.491	94.035	88.989	105.300
19	mean	120.845	153.290	91.115	85.651	94.512	94.278	88.154	104.025
	max	137.601	264.754	111.408	90.047	102.123	102.945	93.989	127.015
	std	8.426	75.605	7.885	2.583	4.458	3.957	4.213	10.525
	min	148.185	102.866	111.148	105.221	116.583	121.006	112.764	115.995
	median	168.455	320.296	119.584	113.839	140.981	130.682	123.086	148.106
20	mean	166.773	261.881	119.236	113.669	139.884	131.311	123.438	148.521
	max	185.485	377.771	132.276	122.457	154.659	139.613	135.300	183.048
	std	10.307	105.028	4.649	4.272	10.055	5.071	5.052	15.519
	min	0	0	0	0	0	0	0	0
	median	6.490	4.090	0	3.552	0	0	0	5.213
21	mean	6.351	11.857	7.5E-07	3.078	0.893	0.001	1.130	6.553
	max	12.826	55.099	2.2E-05	3.552	5.138	0.037	3.762	15.020
	std	3.770	14.631	4.1E-06	1.228	1.668	0.007	1.669	4.667
	min	33.289	0	0	3.552	8.699	3.387	3.552	7.649
	median	42.885	14.208	0	3.552	21.159	11.961	13.212	25.586
22	mean	42.386	48.884	0.474	4.930	20.624	11.510	13.691	25.355
	max	50.719	128.454	3.552	8.690	33.462	17.541	23.428	45.074
	std	4.791	47.193	1.228	1.818	5.749	3.612	4.751	11.528
	min	65.592	0	3.552	17.760	27.757	20.753	12.042	26.357
	median	94.560	14.378	8.067	18.100	52.652	29.963	33.199	59.561
23	mean	93.180	51.745	9.269	19.290	51.787	30.153	32.884	58.324
	max	105.063	231.198	14.990	23.238	72.605	40.446	42.918	88.369
	std	8.714	75.848	4.243	1.974	10.185	4.553	7.000	16.076
	min	90.555	0	0	22.898	70.394	43.377	31.612	57.931
	median	135.687	20.019	14.548	26.450	91.950	59.616	50.779	90.813
24	mean	133.143	116.119	13.820	27.504	94.698	58.368	51.204	92.303
	max	152.926	361.609	22.216	37.065	122.852	68.916	65.893	137.916
	std	13.447	142.139	4.671	3.412	12.196	7.376	8.207	24.918
	min	60.686	0	0.183	0.120	53.576	0	26.523	29.946
	median	104.889	0	37.116	0.279	86.615	62.168	76.352	84.601
25	mean	104.439	13.471	36.067	0.335	86.386	57.876	73.217	85.128
	max	125.811	161.451	78.464	0.849	99.178	72.786	91.105	126.704
	std	14.153	41.309	20.464	0.208	10.621	17.322	13.625	18.336
	min	163.671	0	135.256	154.755	233.503	130.362	208.422	207.208
	median	314.126	41.748	194.938	195.832	267.512	164.385	242.386	270.424
26	mean	295.113	74.707	187.533	195.388	265.412	161.362	239.153	268.315
	max	331.847	329.923	222.756	237.361	300.395	187.579	261.766	330.771
	std	40.180	106.332	27.320	18.559	19.013	12.448	13.686	31.063
	min	348.402	0	193.006	317.022	374.203	221.197	304.657	341.376
	median	509.195	41.849	311.162	386.469	446.116	249.563	368.799	436.817
27	mean	488.725	54.362	301.460	382.444	441.602	247.384	362.159	436.950
	max	528.011	494.286	354.265	413.896	467.535	271.327	391.294	529.460
	std	50.179	89.360	43.729	18.832	24.525	13.193	25.341	38.330
	min	548.602	0	0	506.788	576.439	273.658	473.872	530.989
	median	711.098	41.795	279.976	525.713	635.228	334.567	521.129	606.926
28	mean	692.511	65.597	276.620	528.635	632.042	329.717	518.021	614.407
	max	727.464	580.101	527.866	554.467	672.459	364.781	563.720	715.581
	std	48.104	108.293	201.979	13.152	21.947	24.451	24.890	43.689
	min	23.854	17.386	28.283	18.695	16.092	16.002	16.246	23.265
	median	34.351	21.897	37.156	39.615	20.530	20.433	19.454	32.242
29	mean	33.905	25.213	36.893	36.677	20.750	20.953	19.828	32.490
	max	42.422	49.843	43.193	51.209	26.747	26.529	22.987	44.599
	std	4.011	9.400	3.599	9.305	2.768	2.137	1.618	5.791
	min	78.816	39.269	104.742	37.788	54.182	52.321	55.594	78.915
	median	91.075	98.698	118.756	49.973	68.264	58.426	63.994	98.324
30	mean	90.372	87.538	115.562	51.413	68.231	58.990	63.439	97.430
	max	102.035	123.785	122.909	77.245	83.318	65.581	73.942	114.089
	std	6.501	29.765	6.035	8.978	7.479	3.394	3.731	9.774
	min	144.170	62.598	174.725	68.015	98.105	82.713	90.469	125.414
	median	155.725	174.524	200.374	85.331	127.193	99.219	103.677	173.904
31	mean	156.636	145.368	200.810	84.046	128.558	99.454	104.418	174.087
	max	181.204	199.811	219.169	104.426	149.310	111.173	117.778	202.850
	std	9.036	51.971	10.301	10.350	12.461	5.827	7.043	17.711
	min	204.304	80.782	256.491	94.565	171.158	135.534	131.067	214.665
	median	242.529	112.234	286.675	114.071	198.139	153.238	161.119	274.342
32	mean	242.741	170.763	285.455	117.005	197.970	152.829	158.040	268.152
	max	275.858	271.524	306.145	138.448	223.666	164.808	177.022	309.456
	std	15.207	82.614	11.193	12.941	14.681	6.469	10.930	24.682

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3144067, IEEE Access

REFERENCES

[1] M. Hellwig and H.-G. Beyer, "Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – a critical review," Swarm and Evolutionary Computation, vol. 44, pp. 927–944, 2019.

[2] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, and C. Weihs, "Analyzing the bbob results by means of benchmarking concepts," vol. 23, no. 1, 2015.

[3] R. Rardin and R. Uzsoy, "Experimental evaluation of heuristic optimization algorithms: A tutorial," Journal of Heuristics, vol. 7, p. 261–304, 2001.

[4] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization," Cairo University, Egypt, Tech. Rep., 2020.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," Ph.D. dissertation, INRIA, 2009.

[6] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "Coco: a platform for comparing continuous optimizers in a black-box setting," Optimization Methods and Software, vol. 36, no. 1, pp. 114–144, 2021.

[7] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," International Journal of Mathematical Modelling and Numerical Optimisation, vol. 4, no. 2, pp. 150–194, 2013.

[8] R. D. Lang and A. P. Engelbrecht, "An exploratory landscape analysis-based benchmark suite," Algorithms, vol. 14, no. 3, 2021.

[9] M. A. Muñoz and K. Smith-Miles, "Generating New Space-Filling Test Instances for Continuous Black-Box Optimization," Evolutionary Computation, vol. 28, no. 3, pp. 379–404, 09 2020.

[10] U. Škvorc, T. Eftimov, and P. Korošec, "Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis," Applied Soft Computing, vol. 90, p. 106138, 2020.

[11] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," in 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1641–1649.

[12] Y.-W. Zhang and S. K. Halgamuge, "Similarity of continuous optimization problems from the algorithm performance perspective," in 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 2949–2957.

[13] L. A. Christie, A. E. Brownlee, and J. R. Woodward, "Investigating benchmark correlations when comparing algorithms with parameter tuning," in Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2018, pp. 209–210.

[14] I. Fister, J. Brest, A. Iglesias, A. Galvez, S. Deb, and I. Fister, "On selection of a benchmark by determining the algorithms' qualities," IEEE Access, vol. 9, pp. 51 166–51 178, 2021.

[15] T. Weise and Z. Wu, "Difficult features of combinatorial optimization problems and the tunable w-model benchmark problem for simulating them," ser. GECCO '18. New York, NY, USA: Association for Computing Machinery, 2018.

[16] J. Kudela, "Novel zigzag-based benchmark functions for bound constrained single objective optimization," in 2021 IEEE Congress on Evolutionary Computation (CEC), 2021, pp. 857–862.

[17] K. R. Opara, A. A. Hadi, and A. W. Mohamed, "Parametrized benchmarking: An outline of the idea and a feasibility study," ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020.

[18] P. Bujok and R. Polakova, "Eigenvector crossover in the efficient jso algorithm," MENDEL, vol. 25, no. 1, pp. 65–72, Jun. 2019.

[19] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," International Journal of Machine Learning and Cybernetics, vol. 11, pp. 1501–1529, 2020.

[20] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, and N. H. Awad, "Evaluating the performance of adaptive gainingsharing knowledge based algorithm on cec 2020 benchmark problems," in 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8.

[21] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in 2005 IEEE Congress on Evolutionary Computation, vol. 2, 2005, pp. 1769–1776 Vol. 2.

[22] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341–359, 1997.

[23] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–7.

[24] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," in 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8.

[25] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1658–1665.

[26] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through bayesian hyperparameter optimization," in 2021 IEEE Congress on Evolutionary Computation (CEC), 2021,

pp. 832–840.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[28] A. Kazikova, M. Pluhacek, and R. Senkerik, "Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison?" MENDEL, vol. 26, no. 2, pp. 9–16, Dec. 2020.

[29] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, "IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics," arXiv e-prints:1810.05281, Oct. 2018.

[30] T. Eftimov, G. Petelin, and P. Korošec, "Dsctool: A web-service-based framework for statistical comparison of stochastic optimization algorithms," Applied Soft Computing, vol. 87, p. 105977, 2020.

[31] T. Eftimov, R. Hribar, U. Skvorc, G. Popovski, G. Petelin, and P. Koroec, "Performviz: a machine learning approach to visualize and understand the performance of single-objective optimization algorithms," Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, 2020.

[32] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," ser. GECCO '11. New York, NY, USA: Association for Computing Machinery, 2011.

[33] P. Kerschke and H. Trautmann, "Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco," in Applications in Statistical Computing, 2019, pp. 93–123.

[34] B. Qu, J. Liang, Z. Wang, Q. Chen, and P. Suganthan, "Novel benchmark functions for continuous multimodal optimization with comparative results," Swarm and Evolutionary Computation, vol. 26, pp. 23–34, 2016.

JAKUB KUDELA received his M.S. degree in mathematical engineering from Brno University of Technology in 2014 and received his Ph.D. degree in Applied Mathematics from BUT in 2019.

From 2018, he works as a Research Assistant with the Institute of Automation and Computer Science, BUT. His research interest includes the development of computational methods for various optimization problems and engineering applications.

RADOMIL MATOUSEK received his M.S. degree in Applied Computer Science from Brno University of Technology in 1996 and received his Ph.D. degree in Technical Cybernetics from Brno University of Technology in 2004.

From 2012, he works as a assoc. Professor with the Institute of Automation and Computer Science, BUT. His research interest includes the development of AI and computational methods for engineering applications.

A7

Soft Computing (2022) 26:13709–13733
https://doi.org/10.1007/s00500-022-07362-8

APPLICATION OF SOFT COMPUTING

Recent advances and applications of surrogate models for finite element method computations: a review

Jakub Kudela[1] · **Radomil Matousek**[1]

Accepted: 7 June 2022 / Published online: 17 July 2022

Abstract
The utilization of surrogate models to approximate complex systems has recently gained increased popularity. Because of their capability to deal with black-box problems and lower computational requirements, surrogates were successfully utilized by researchers in various engineering and scientific fields. An efficient use of surrogates can bring considerable savings in computational resources and time. Since literature on surrogate modelling encompasses a large variety of approaches, the appropriate choice of a surrogate remains a challenging task. This review discusses significant publications where surrogate modelling for finite element method-based computations was utilized. We familiarize the reader with the subject, explain the function of surrogate modelling, sampling and model validation procedures, and give a description of the different surrogate types. We then discuss main categories where surrogate models are used: prediction, sensitivity analysis, uncertainty quantification, and surrogate-assisted optimization, and give detailed account of recent advances and applications. We review the most widely used and recently developed software tools that are used to apply the discussed techniques with ease. Based on a literature review of 180 papers related to surrogate modelling, we discuss major research trends, gaps, and practical recommendations. As the utilization of surrogate models grows in popularity, this review can function as a guide that makes surrogate modelling more accessible.

Keywords Surrogate model · Surrogate-assisted optimization · Sensitivity analysis · Uncertainty quantification · Finite element method

1 Introduction

The methods of numerical analysis, such as the finite-element method (FEM), computational fluid dynamics (CFD), or structural finite-element analysis (FEA), are routinely employed to perform analysis of complex systems and structures where obtaining an analytical solution may be either difficult or impossible. Such analyses are becoming ubiquitous in evaluating and optimizing design, reliability, and maintenance of complex systems and structures in a broad range of various industrial applications including aerospace (Yan et al. 2020), automotive (Berthelson et al. 2021), architecture (Westermann and Evins 2019), biomedical engineering (Putra et al. 2018), chemical engineering (Bhosekar and Ierapetritou 2018), and many others. However, these computer simulations tend to be very computationally demanding because of their intrinsically detailed description of the studied systems. These engineering problems based on computer models also require the computation of thousands of simulations in order to construct a suitable solution, requiring a large computational budget (Alizadeh et al. 2020). Additionally, because of their high fidelity, various issues in performing computer simulations can occur regardless of how much computer power can be used. Even the recent advance of parallel and pooling (Kudela and Popela 2020) computing methods, that carry out many calculations or executions of processes simultaneously, do not seem to be very helpful (Grama et al. 2003).

The principle purpose of using surrogate models (or metamodels) is to approximately emulate the expensive-to-evaluate high-fidelity models, such as a FEM-based model, employing computationally less costly statistical models. These surrogates are constructed based on a relatively low number of simulation input and output data, that are computed employing the high-fidelity expensive computations.

✉ Jakub Kudela
Jakub.Kudela@vutbr.cz

[1] Institute of Automation and Computer Science, Brno University of Technology, Technicka 2, Brno 616 00, Czech Republic

After the surrogate is validated to achieve a sufficient level of approximation of the FEM-based model, its utilization to predict the outputs of the high-fidelity model can be done almost instantly.

There are several features of a given problem, including its linearity/nonlinearity, the required accuracy level, the size (input dimensions) of the problem, the required amount of information, the speed of the computations, the number of samples, and the availability of convenient software tool that impact the appropriateness of a given surrogate (Alizadeh et al. 2020). It is possible to divide the use of surrogate models into three classes of problems. The first class contains the most fundamental utilization of surrogates—building and validating surrogate models and using them for prediction. The second class of problems deal with sensitivity analysis of the resulting models and different ways of quantifying the impact of uncertain parameters, that may influence the behaviour of the modelled systems. The third class is commonly called surrogate-assisted optimization, in which the objective function used for optimization is prohibitively expensive to compute and the information about its derivative is not available.

The primary motivation of this paper lies in investigating recent advances and applications of surrogate models for FEM-based computations. Although the utilization of surrogate models is growing in popularity, a text summarizing the state-of-the-art and recent developments (especially for FEM-based computations) was missing. The novelty of this paper is in encapsulating the state-of-the-art in surrogate modelling for FEM-based computations from both the theoretical and application perspectives. We also expect this work to function as a guide in the selection of the suitable approximation models for applications of computationally expensive high-fidelity FEM-based problems. This review emphasizes the differences between employing surrogates for the three above mentioned problem classes and gives a comprehensive overview of surrogate modelling and corresponding techniques. The main contributions of this paper, obtained by assessing 180 papers related to surrogate modelling, are the following:

- Analysis of the state-of-the-art in surrogate modelling techniques.
- Review of the recent advances in using surrogate models for FEM computations—highlighting both theoretical and application developments in model building and validation, sensitivity analysis and uncertainty quantification, and surrogate-assisted optimization.
- Description of available software tools.
- Identification of trends and research gaps.

The rest of the paper is organized as follows. Section 2 discusses the basics of surrogate modelling: sampling, model validation methods, and various model choices for surrogates and the underlying mathematical formulas. Section 3 describes the sensitivity analysis and uncertainty quantification approaches and their applications in FEM-based computations. In Sect. 4, we give a thorough overview of recent methodological advances and applications of surrogate-assisted optimization. Section 5 lists the most widely used as well as recently developed software tools. Section 6 summarizes the trends, research gaps, and recommendations extracted from the considered literature. Finally, conclusions and suggestions for further research are drawn in Sect. 7.

2 Surrogate modelling

First, we discuss the most frequently utilized approaches for building and validating the surrogates of the expensive-to-compute functions. We will deal with a surrogate $\hat{f}(x)$ of the function $f : R^d \rightarrow R$, where $x = (x_1, x_2, \ldots, x_d)$ is the input vector, d is the number of dimensions of the problem, and we have a single output y. The upper and lower bounds on the input (or design) vector are known and are denoted as $x_L \leq x \leq x_U$.

2.1 Sampling and model validation

After determining the input parameters (or the design space) of the model, we must select the data points (or designs) to evaluate for building the surrogate model. This process is usually called sampling, and sometimes is also referred to as design of experiments (DOE). The number of samples we choose to evaluate and their quality has a direct impact on the precision of the surrogate. As evaluating the data points comprises of the computation of the true function (in this case, running the expensive simulation), sampling is the source of significant computational costs. Maintaining the quality of the surrogate model without suffering prohibitive sampling cost can be achieved by using appropriate sampling strategies (Bhosekar and Ierapetritou 2018).

Sampling strategies can be divided into stationary sampling and adaptive sampling. The methods of stationary sampling are based upon geometry or patterns (such as full/half factorial design, or grid sampling) and approaches based on the DOE literature (orthogonal sampling, Box–Behnken design, etc.). Among the most frequently used used strategies of stationary sampling are the Latin Hypercube Sampling (LHS) (McKay et al. 1979), the maximin sampling (Johnson et al. 1990), and the sampling techniques of Morris and Mitchell (1995).

Conversely, adaptive sampling starts from fewer samples which are usually computed by using one of the stationary sampling strategies, but new sample points are determined sequentially. The goal of adaptive strategies is

to decrease sampling requirements by evaluating samples that are expected to increase the precision of the resulting surrogate. Most adaptive sampling approaches use various criteria for balancing the trade-offs between exploring the under-explored regions of the design space (exploration) and refining the regions close to the already evaluated samples for improved performance (exploitation). A frequent use of such an approach is within surrogate-assisted optimization in which exploration is applied to deal with local optima while exploitation aims at improving on the best design found so far. In the case of Kriging surrogates, one of the popular approaches is based on the Expected Improvement (EI) criterion (Jones et al. 1998), while for Radial Basis Function (RBF) surrogates, an analogous quantitative measure is achieved using a so called bumpiness criterion (Gutmann 2001).

In general, the approaches that address this exploration /exploitation trade-off have been demonstrated to attain better surrogate accuracy with a lower number of samples (Provost et al. 1999). The approaches based on a space-filling sequential design were also studied in Crombecq et al. (2011). Here, the authors developed a group of sequential sampling approaches which exhibit performance competitive with one-shot or stationary experimental designs. It is possible to reformulate the adaptive sampling problem as an optimization problem (Cozad et al. 2014), where the objective function measures the discrepancy between the expensive-to-compute function $f(x)$ and its surrogate $\hat{f}(x)$

$$\max \left(\frac{f(x) - \hat{f}(x)}{f(x)} \right)^2, \quad x_L \leq x \leq x_U.$$

Other contemporary methods make use of ranking the exploitation and the exploration and weighing them as needed. Garud et al. (2017) developed such a method, where a metric that consisted of two different measures for exploitation and exploration was employed. As the exploitation metric, they quantified the impact of the new sample point added near a point that was already sampled by the so called departure function. As the exploration metric, they used the sum of squares of the distances between the new sample and all the previously evaluated samples. By estimating the prediction variance of a surrogate model by the jackknifing technique, (Eason and Cremaschi 2014) suggest a different adaptive sampling method where the surrogate model is constructed using ANN and sample points that exhibit high prediction variance are selected. Such a method of adaptive sampling has the advantage of not being specific to the selection of particular surrogate models.

Evaluating the reliability of the constructed surrogate model constitutes one of the most important concerns—relying on a flawed surrogate model can result in a misuse of computational resources, and produce negative effects on prediction, optimization, or the resulting analyses. Validation of surrogate models is the procedure of evaluating the performance of the resulting surrogate models where, apart from measuring their accuracy, validation techniques are frequently utilized in the selection of a suitable surrogate model from a collection of candidate models and in tuning of its hyperparameters.

A common approach is to employ resampling techniques, such as bootstrapping or cross-validation (Forrester and Keane 2009). In the cross-validation approaches, the training data for a surrogate model are randomly partitioned into q subsets of roughly equal size. These subsets are then in turn removed from the set of the training data while the fitting of the model is performed on data that remain. Afterwards, the subset that was removed is predicted by the model that was fitted to the data that remained. Once every subset has been removed, the calculations of n predictions denoted by $\hat{y} = (\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(n)})$ of the n observed data points $y = (y^{(1)}, y^{(2)}, \ldots, y^{(n)})$ will be performed. The difference between the observations and the obtained predictions, i.e., the prediction error, is quantified by using various validation metrics. One of the most frequently used metrics for validation is the Mean Squared Error (MSE), calculated as

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2.$$

Using $q = n$, one can obtain an error estimate that is almost unbiased, but its variance is often very large. Hastie et al. (2001) suggested using a bit larger subsets, that have $q = 5$ or 10.

A similar method, but one which allows repeated samples in the training data is bootstrapping. By enabling the repetition of the samples in the set that is designated for building the model, we obtain a training set that has the same size as the actual data. Generally, for bootstrapping the number chosen of subsets q is larger than in the case of cross-validation. Apart from the already mentioned MSE, there are other commonly used validation metrics such as the explained variance score, the mean absolute error, the median absolute error, and the R^2 score. These metrics, along with the corresponding mathematical formulations, can be found in Table 1, where $\bar{y}$ denotes the mean predicted value. More details on various resampling approaches that are used for validation of surrogate models are discussed by Bischl et al. (2012).

Table 1 Conventionally used metrics for surrogate validation (Hastie et al. 2001)

Validation metric	Mathematical formula
R^2 score	$1 - \frac{\sum_{i=1}^{n}(y^{(i)}-\hat{y}^{(i)})^2}{\sum_{i=1}^{n}(y^{(i)}-\bar{y})^2}$
Mean absolute error	$\frac{1}{n}\sum_{i=1}^{n} \mid y^{(i)} - \hat{y}^{(i)} \mid$
Median absolute error	$\text{median}(\mid y^{(1)} - \hat{y}^{(1)} \mid, \ldots, \mid y^{(n)} - \hat{y}^{(n)} \mid)$
Explained variance score	$1 - \frac{\text{Var}(y-\hat{y})}{\text{Var}(y)}$

2.2 Model choice

2.2.1 Response surfaces and linear regression

The classic polynomial response surface model (RSM) (Box and Draper 1987) is the original and to this day one of the most frequently employed types of surrogate models in engineering design (Forrester and Keane 2009). In the RSM method, the surrogate is represented as a linear combination of polynomial functions (mostly linear and quadratic) of the input variables. A so called first-order RSM has the following form:

$$\hat{f}(x, a) = a_0 + a_1 x_1 + \cdots + a_m x_m$$

where the vector $a = (a_0, \ldots, a_m)^T$ is obtained by minimizing the sum of squared errors between the value from the expensive-to-compute function and the value predicted by the surrogate model. This (least squares) minimization problem is unconstrained and can be written as

$$\min \mid\mid Xa - y \mid\mid_2^2,$$

where the matrix X of size n by $m+1$ has all elements in its first column equal to 1 while the remaining columns contain the input vector. In the case of ordinary least squares, there is an analytical solution in the form $a = (X^T X)^{-1} X^T y$. If one or more of the input vector components x_i are perfectly correlated, the resulting matrix $X^T X$ can become singular (or can be very badly conditioned), which results in the coefficients a not being uniquely defined. Such an issue is usually approached by decreasing the number of input variables by prescreening, or by using regularization methods, such as ridge regression or lasso (Hastie et al. 2001).

Higher order polynomials are also commonly used, especially if there is a curvature in the problem. For example, a second-order model has the following form:

$$\hat{f}(x, a, \alpha) = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m} \alpha_{ii} x_i^2 + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \alpha_{ij} x_i x_j,$$

where the coefficients of a and α are again obtained by solving a least squares problem. Polynomial surrogates remain generally not well suited for the nonlinear, multi-dimensional, multi-modal design landscapes one deals with in engineering unless the ranges of the considered variables are made sufficiently small, e.g. in trust-region methods (Forrester and Keane 2009).

Also, in problems that are high-dimensional it may be impossible to sample enough data required for the estimation and construction of all except the low order polynomials. On the other hand, for problems that are not high-dimensional, display low modality (or unimodality), or where data are relatively inexpensive to compute, the use of polynomial surrogates may be an attractive (and correct) choice. Moreover, the individual terms of the polynomial expression computed by the methods mentioned above can give insight about the problem itself, e.g. the role of the individual inputs is quite easily judged by the value of the corresponding coefficient.

2.2.2 Kriging

Kriging refers to a surrogate model that is based on Gaussian process modelling (and is sometimes called Gaussian process regression Rasmussen and Williams 2006). The method first originated in geostatistics in a paper by Krige (1951) and became popular after its use for analysis and of various computer experiments (Sacks et al. 1989). Nowadays, it is among the most widely used methods for building surrogate models.

A Kriging surrogate model can be formulated as follows:

$$\hat{f}(x) = \sum_{i=1}^{m} a_j g_j(x) + \varepsilon(x), \tag{1}$$

where $g_j(x)$ denotes the m independent (and known) basis functions which describe the trend of prediction of the mean at the point x, a_j denotes the unknown parameters, and $\varepsilon(x)$ denotes the random error at the point x which follows a normal distribution with a zero mean. The Kriging predictor can be then formulated in the following way

$$\hat{f}(x) = g(x)^T a^* + r(x)^T \alpha^*,$$

where $g(x) = [g_1(x), \ldots, g_m(x)]^T$, a^* denotes the vector of the generalized least-square estimates of $a = [a_1, \ldots, a_m]^T$, $r(x)$ denotes the correlation vector of size $n \times 1$ between $\varepsilon(x)$ and $\varepsilon(x_i)$, and a^* and α^* are computed as

$$a^* = (G^T R^{-1} G)^{-1} G^T R^{-1} y,$$
$$\alpha^* = R^{-1}(y - Ga^*),$$

where R denotes the $n \times n$ covariance matrix whose (i, j) element describes the correlation between $\varepsilon(x^{(i)})$ and $\varepsilon(x^{(j)})$,

$G = [g(x^{(1)}), \ldots, g(x^{(n)})]^T$ is $n \times m$ matrix, and y denotes the observations at the available points. The variance of the process, denoted by σ^2, can be computed as

$$\sigma^2 = \frac{1}{n}(y - G^T a^*)^T R^{-1}(y - G^T a^*).$$

In the Kriging surrogate model, we assume that the random variables $\varepsilon(x)$ are correlated in accordance to the correlation model $R(\cdot, \cdot)$, which is parameterized by a collection of hyperparameters θ. These hyperparameters are usually computed by using maximum likelihood estimation (MLE) methods. One can find a comprehensive treatment of MLE in the context of Kriging in Kaymaz (2005).

Depending on the particular selection of the model for mean prediction $g(x)^T a$ in (1), there exist different modifications of Kriging: universal Kriging (also called Kriging with trend), ordinary Kriging, and simple Kriging. In simple Kriging we assume that the term $g(x)^T a$ is a known constant while in ordinary Kriging we assume it is an unknown constant. In universal Kriging we assume that $g(x)$ is any other (prespecified) function of x. Frequently, $g(x)$ has the form of a polynomial regression (of a lower order). Conventionally, the selection of the order of the polynomial is done empirically. On the other hand, this kind of a non-adaptive framework can make the modelling rather ineffective. To bypass this issue, blind Kriging (Joseph et al. 2008; Hung 2011) and similar methods (Kamiński 2015) are used.

Another important feature of Kriging is the selection of a suitable covariance function (Lirio et al. 2014). Usually, the covariance functions employed with Kriging surrogates are stationary ones which can be formulated as

$$R(x, x') = \prod_j \psi_j(\theta, m_j), \quad m_j = x_j - x'_j.$$

A correlation function expressed in this way enjoys two attractive properties. The first one is that it is possible to express the correlation function for multivariate functions by a product of several one-dimensional correlations. The second one is that the correlation is stationary, depending only on the distance m_j between the two points x and x'.

Frequently employed correlation models can be found in Table 2, where Γ is the Gamma function, K_{v_j} the modified Bessel function of order v_j, and the parameter $v_j > 0$ controls the differentiability of the Matern correlation model. Chen et al. (2016) compared several of the mentioned correlation models, showing a worse performance of the squared exponential correlation model in comparison to the exponential correlation one. On the other hand, an important note is that the generalized exponential correlation model needs twice as many hyper-parameters ($2d$) when compared to squared exponential correlation one. The authors also suggested using the Matern model (see Table 2) as a better alternative to the exponential correlation one.

2.2.3 Radial basis functions

RBFs (Broomhead and Lowe 1988) compute a weighted sum of prespecified simple functions to approximate complex design landscape. Sobester (2003) used an analogy of mimicking the characteristic timbre of a musical instrument by a synthesizer that uses a weighted combination of different tones. Given n different sample points, the RBF surrogates are written as

$$\hat{f}(x) = \sum_{i=1}^{n} w_i \psi(|| x - x^{(i)} ||_2),$$

where w_i denotes the weight which is computed using the method of least-squares, and ψ is the chosen basis function. There are several (symmetric) radial functions that can serve as a basis function, with the most widely ones summarized in Table 3.

An important note is that unlike response surface methods, RBF does not belong to the regression techniques. Conversely, RBF is broadly considered to be an interpolation method. This means that RBFs, in contrast with regression techniques, give exact result at the points that were already sampled. There is still no firm conclusion in the literature that would decisively show whether some of the mentioned basis functions are better than the others.

2.2.4 Support vector regression

SVR is based on the theory of support vector machines (SVM), that originated at AT&T Bell Laboratories (Vapnik 1995). In our context of surrogate-assisted engineering design, it is arguably more fitting to view SVR as the extension of the RBF techniques rather than of the SVMs (Forrester and Keane 2009).

One of the main attributes of the SVR methods is that they give us the possibility to prescribe a margin (δ) within which one can accept errors present in the sampled data without having negative effect on the prediction capabilities of the resulting surrogate models. This might be helpful in situations when the sample data contain random errors because of, for instance, a finite size of the mesh, because by performing an analysis of mesh sensitivity we can determine an appropriate value of δ.

The basic shape of an SVR prediction has the familiar form of a weighted sum of prescribed basis functions ψ, which have weights w, and are added to the "base" term μ. These are calculated in a different way to their Kriging and RBF counterparts, yet they contribute to the surrogate prediction in the exact same manner:

Table 2 Frequently employed correlation models for Kriging surrogates (Kleijnen 2017)

Name	Correlation model
Exponential	$\psi_j(\theta, m_j) = \exp(-\theta_j \mid m_j \mid)$
Generalized exponential	$\psi_j(\theta, m_j) = \exp(-\theta_j \mid m_j \mid^{\theta_{n+1}}), \quad 0 \le \theta_{n+1} \le 2$
Gaussian	$\psi_j(\theta, m_j) = \exp(-\theta_j m_j^2)$
Linear	$\psi_j(\theta, m_j) = \max\{0, 1 - \theta_j \mid m_j \mid\}$
Spherical	$\psi_j(\theta, m_j) = 1 - 1.5\xi_j + 0.5\xi_j^2, \quad \xi_j = \min\{1, \theta_j \mid m_j \mid\}$
Cubic	$\psi_j(\theta, m_j) = 1 - 3\xi_j^2 + 2\xi_j^3, \quad \xi_j = \min\{1, \theta_j \mid m_j \mid\}$
Spline	$\psi_j(\theta, m_j) = \begin{cases} 1 - 5\xi_j^2 + 30\xi_j^3, & 0 \le \xi < j \le 0.2 \\ 1.25(1 - \xi_j^3), & 0.2 < \xi_j \le 1 \\ 0, & \xi_j > 1 \end{cases}, \xi_j = \theta_j \mid m_j \mid$
Matern	$\psi_j(\theta, m_j) = \frac{1}{\Gamma(v_j)2^{(v_j-1)}}(\theta_j \mid m_j \mid)^{v_j} K_{v_j}(\theta_j \mid m_j \mid)$

Table 3 Commonly used radial basis functions (Bhosekar and Ierapetritou 2018)

Name	Radial basis function
Linear	$\psi(r) = r$
Cubic	$\psi(r) = r^3$
Multi-quadric	$\psi(r) = \sqrt{r^2 + \gamma^2}, \ \gamma > 0$
Inverse multi-quadric	$\psi(r) = \frac{1}{\sqrt{r^2+\gamma^2}}, \ \gamma > 0$
Thin plate spline	$\psi(r) = r^2 \ln(r)$
Gaussian	$\psi(r) = \exp(-\frac{r^2}{\gamma}), \ \gamma > 0$

$$\hat{f}(x) = \mu + \sum_{i=1}^{n} w_i \psi(x, x^{(i)}).$$

For a linear mapping, this can be rewritten using an inner product $\langle \cdot, \cdot \rangle$ as

$$\hat{f}(x) = \mu + \langle w, x \rangle. \tag{2}$$

Finding the weights that minimize the margin δ corresponds to solving the following convex optimization problem

$$\begin{aligned} \min \ & \frac{1}{2} \mid\mid w \mid\mid_2^2 \\ \text{subject to} \ & y^{(i)} - \langle w, x^{(i)} \rangle - \mu \le \delta, \quad i = 1, \ldots, n, \\ & \langle w, x^{(i)} \rangle + \mu - y^{(i)} \le \delta, \quad i = 1, \ldots, n. \end{aligned}$$

As there might not be a feasible solution to the optimization problem above, an extension using slack variables ξ^+ and ξ^- is used in practice

$$\begin{aligned} \min \ & \frac{1}{2} \mid\mid w \mid\mid_2^2 + \gamma \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) \\ \text{subject to} \ & y^{(i)} - \langle w, x^{(i)} \rangle - \mu \le \delta + \xi_i^+, \quad i = 1, \ldots, n, \\ & \langle w, x^{(i)} \rangle + \mu - y^{(i)} \le \delta + \xi_i^-, \quad i = 1, \ldots, n, \\ & \xi_i^+, \xi_i^- \ge 0, \quad i = 1, \ldots, n, \end{aligned}$$

where γ is a regularization parameter that controls the desired trade-offs between the complexity of the model and a level for which errors larger than δ are allowed. Using Lagrangian duality theory, a dual optimization model can be constructed

$$\begin{aligned} \max \ & -\frac{1}{2} \sum_{i,j=1}^{n} (\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-) \langle x^{(i)}, x^{(j)} \rangle \\ & - \delta \sum_{i=1}^{n} (\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^{n} y^{(i)} (\alpha_i^+ - \alpha_i^-) \qquad (3) \\ \text{subject to} \ & \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-) = 0 \\ & \alpha_i^+, \alpha_i^- \in [0, \gamma], \quad i = 1, \ldots, n, \end{aligned}$$

where the resulting weights are computed as

$$w = \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-) x^{(i)},$$

and, to compute μ the Karush–Kuhn–Tucker (KKT) conditions have to be used. Substituting into (2), we get

$$\hat{f}(x) = \mu + \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-) \langle x^{(i)}, x \rangle.$$

The advantage of the Lagrangian formulation is that we can extend the above mentioned model beyond the usual linear regression to different basis functions (which are known in the support vector literature as kernels), that are able to approximate more complex landscapes, by the so-called kernel trick. The procedure amounts to replacing the inner product $\langle \cdot, \cdot \rangle$ with a kernel function ψ, which needs to satisfy the conditions for a Mercer kernel. The most popular choices

Table 4 Commonly used kernel functions (Hastie et al. 2001)

Name	Mercer kernel function
Gaussian	$\psi(x^{(i)}, x^{(j)}) = \exp(-\frac{\|\|x^{(i)}-x^{(j)}\|\|^2}{\sigma^2}),\ \sigma > 0$
Linear	$\psi(x^(i), x^(j)) = \langle x^{(i)}, x^{(j)} \rangle$
d degree homogeneous polynomial	$\psi(x^{(i)}, x^{(j)}) = (\langle x^{(i)}, x^{(j)} \rangle)^d$
d degree inhomogeneous polynomial	$\psi(x^{(i)}, x^{(j)}) = (\langle x^{(i)}, x^{(j)} \rangle + c)^d$

for ψ are shown in Table 4. Whichever form of the kernel function is selected, the method for computing support vectors stays the same—as the properties of the Mercer kernel guarantee the optimization problem (3) is a convex quadratic one it can be readily solved by employing quadratic programming solvers.

2.2.5 Artificial neural networks

Artificial neural networks (ANNs) form a group of surrogate models that take inspiration from the biological functions found in the brain and in the nervous system. The structure of an ANN is usually represented by a system of mutually interconnected "neurons". These neurons in an ANN contain various primitive functions, and the connections between the neurons have numeric weights that are computed according to the input data. There are three elements that guide the construction of an ANN: a network topology (single or multiple layers), primitive functions related to the neurons, and a learning algorithm used to compute the corresponding weights. Most widely used paradigm of an ANN consists of a so-called multilayer perceptron (MLP) that is based on feed-forward (supervised) learning, and on the back-propagation algorithm (Sun and Wang 2019).

Among the various ANNs architectures, multilayer feed-forward neural network (MFFNN) are among the most frequently used ones (Chatterjee et al. 2019). In the MFFNN the neurons are organized into three layers: the input layer, the hidden layer, and the output layer. Note that the number of layers in MFFNN can be larger than just one and usually a convergence study is needed to find the right number of hidden layers. Similarly, the number of neurons in the hidden layers needs to be chosen based on a suitable convergence criterion.

The learning capabilities of an MLP architecture can be improved by using more hidden layers, and/or neurons in the hidden layers. On the other hand, there are trade-offs between the learning/prediction capabilities the network and its size. To address these issues, the approaches of "deep learning" have recently gained increased attention, although deep learning is only a subconcept under the ANNs. When compared to its conventional MLP counterparts, the deep learning approaches utilize more training layers and characterized layers (e.g. pooling and convolutional layers).

2.2.6 Polynomial chaos expansion

The polynomial chaos expansion (PCE) is a method for producing responses of stochastic systems which was developed by Wiener (1938). Afterwards, generalized results were introduced by Xiu and Karniadakis (2002) for different discrete and continuous system from the so-called Askey scheme.

Let $\mathbf{i} = (i_1, \ldots, i_n)$ be a vector of nonnegative integers (called a multi-index), with $|\ \mathbf{i}\ | = i_1 + \cdots + i_n$, and let N be a nonnegative integer. Them the Nth order PCE of $f(x)$ can be stated in the following way

$$\hat{f}(x) = \sum_{|\mathbf{i}|=0}^{N} a_{\mathbf{i}} \Phi_{\mathbf{i}}(x),$$

where $a_{\mathbf{i}}$ denotes the unknown coefficients to be determined and $\Phi_{\mathbf{i}}$ are n dimensional orthogonal polynomials that have the maximum order N and satisfy the following relation

$$\int_{x \in \Omega} \Phi_{\mathbf{i}}(x) \Phi_{\mathbf{j}}(x) d\Gamma(x) = \delta_{ij},\ 0 \leq |\ \mathbf{i}\ |, |\ \mathbf{j}\ | \leq N,$$

where δ_{ij} is the Kronecker delta , Ω is the support, and Γ is the chosen measure. Based on the selection of Ω and Γ, different orthogonal polynomials can be obtained (such as Hermite, Laguerre, Meixner, etc.). A review and a comparison of different sampling strategies for PCE was performed in Hadigol and Doostan (2018).

2.2.7 Boosted trees and random forests

Boosted trees (BTs), also known as gradient boosting machines, were introduced by Friedman (2001) as a supervised learning method. BTs are employed in problems of supervised learning where there are several features $x^{(i)}$ that are available in the training set and are used to estimate an output variable $y^{(i)}$ and to construct a prediction model that has a form of a collection of weaker prediction models which are usually decision trees (De'ath 2007).

Random forests (RFs) represent ensemble learning methods for classification, regression, and similar tasks. They are constructed by developing an aggregate of different decision trees from the training stage and producing as output the group which is corresponds to mode of the different groups (in classification) or to mean estimation (in regression) of individual trees (Ho 1998). RFs are used to correct the decision trees' issue of overfitting to the training dataset (Hastie et al. 2001).

2.2.8 Other approaches

There are also other approaches that are less common but still deserve to be mentioned:

- Adaptive learning/active learning—a semi-supervised machine learning method in which the algorithm refers to its information sources interactively to reach the favourable outcomes for the newly generated dataset (Gorissen et al. 2010).
- Lipschitz-based surrogates—the use of Lipschitz supporting hyperplanes for function approximation in the context of surrogate assisted optimization was proposed in Zhou et al. (2016) for the purpose of enhancing the exploitation capabilities of the optimization method, while in Kudela and Matousek (2022) the Lipschitz surrogate was used to enhance the exploration abilities of the proposed algorithm.
- Non-Uniform Rational B-splines (NURBs) based models—are characterized by a collection of control points (known as the NURBs orders and knot vectors), that result in a highly flexible and robust curve definition. They can be utilized to construct sequential sampling methods that are adaptive, and give the designer the possibility to efficiently search the interesting regions of the design space (Steuben and Turner 2014).
- Genetic Programming (GP) and Symbolic Regression (SR)—GP is a method of evolutionary computation that enables computers to solve problems automatically. GP is based on an automated learning of computer programs which is founded on the Darwinian principle of survival of the fittest (Koza 1992). In SR, GP is used to construct an empirical mathematical model of the available data. The key point of SR, unlike in the regression techniques discussed above, is not to compute the weights or coefficients in order to best fit a function, but instead to find the shape of the approximate model that is constructed by the evolutionary processes.

2.3 Ensembles of surrogates and multifidelity models

Quite often there is no single class of surrogates that performs better than the other classes for various problems and the issue of selecting the best class of surrogates for the given engineering problem can itself be a challenging problem. Also, there are situations when it is impossible to experiment with multiple classes of surrogate models and select the one which has the best performance.

Ensemble of surrogates (EoS) are being used to mitigate the drawbacks of using a single surrogate model (Acar 2015; Babaei and Pan 2016). For example, adaptive sampling methods which evaluate only a single design per cycle can be used. However, this addition of a single design at a time can become inefficient if it is possible to run the simulations in parallel (Kudela 2019). This issue was addressed in Viana et al. (2010) where the authors developed a method for adding numerous designs per one optimization cycle that is based on a concurrent utilization of EoS. The beneficiality of EoS was analyzed in Song et al. (2018), where the authors provided robustness, efficiency, and accuracy requirements for various specific problems.

A related approach is to employ a mixture (or a weighted combination) of several different surrogate models. Various approaches to compute the weights can be found in the related literature—variance of individual surrogates (Zerpa et al. 2005), a global cross-validation metric (Goel et al. 2007), and error metrics (Müller and Piché 2011) are among the most prevalent. Generally, using multiple surrogates can provide us with the possibility to put more emphasis on the good surrogates while putting less priority on the bad ones as needed.

Multi-fidelity (MF) surrogate models are constructed by a combination of different fidelity models which depend on the specifics of a given problem, with the goal of reducing the high computing cost while giving accurate solutions (Yoo et al. 2020). Multi-fidelity models usually utilize High-fidelity Models (HFMs) as well as Low-fidelity Models (LFMs) to give results of comparable accuracy to surrogates which are based only on HFMs whilst providing a notable reduction in the computation related costs.

In the uncertainty propagation methods, the input of the model is characterized by some random variable. Our interests then lie in the statistics of the output of the model. The use of Monte Carlo simulations for the estimation of these statistics frequently requires numerous evaluations to produce approximations that have sufficient levels of accuracy. The use of a MF model which integrates outputs from the computationally cheap LFMs with outputs from HFMs can result in significant reductions in the runtime and give unbiased estimates of the statistics of the HFM outputs. Similar improvements in computational costs by utilizing MF models can be achieved in optimization process by using LFMs to accelerate searching or using the LFMs in combination with various adaptive correction and trust-region model management schemes (Alizadeh et al. 2020).

2.4 Overview recent advances and applications in surrogate modelling for FEM-based models

The process of constructing surrogate models often becomes intractable in problems where the input space has high dimension, due to the numerous model responses that are needed to accurately estimate the parameters of the model. To alleviate this issue (Zhou and Lu 2020) developed a novel Kriging modeling method that was fused with a dimension reduction technique. To inspect the utility of the new method, the

authors used the pertinent to low cycle fatigue life of a aero-engine compressor disc, where the proposed Kriging-based method was shown to perform better than several well-known surrogate models (such as PCE and ordinary Kriging) when small sample sizes were used.

Sanchez et al. (2017) developed a new methodology for constructing surrogate models from FEM simulations, the so-called Variable Power Law meta-model. The methodology was based upon the response surface method and on a dimensional analysis, and was utilized to compute thermal models suitable for constructing preliminary designs of Multiphysics systems. Compared to traditional surrogate models, it showed an advantage by producing light, compact forms that had sufficient accuracy of prediction over a broad range of input variables (and over several orders of magnitude). Its efficiency was demonstrated on various classes of heat transfer problems, for which it gave an accurate and simple surrogate model every time, and on a aileron actuator application.

Gogu and Passieux (2013) proposed an approach for constructing efficient surrogates for high dimensional outputs combining a novel reduced basis model and a RSM. The goal of the proposed reduced basis modeling is to solve the computationally demanding problem by projecting it onto a reduced-dimensional basis which is build sequentially according to a DOE. Although this kind of a method of an order reduction may be employed as a surrogate by itself, they showed that faster response evaluations were obtained by combining this method with the RSM. The developed method constructs surrogate models based on coefficients which need only a lower number of expensive function evaluations which was enabled by the key points approach: the expensive problem (full scale) was evaluated only at a low number of important DOE points, whilst the reduced order model was employed at the other points. The strengths of the approach are illustrated on the identification problem considering orthotropic elastic constants which was based upon full field displacements and on an example of a surrogate model of a thermal field. Compared to standard surrogate models the proposed method had comparable accuracy while there was a decrease in the resolution time of the system in the DOE by almost an order of magnitude.

Jin and Jung (2016) developed a novel technique for flexible and robust surrogate modeling for finite element model updating (FEMU). They proposed a sequentially updated surrogate model that was based upon a statistical interpretation of a Kriging model. To evaluate the effectiveness of the proposed method when applied to FEMU, they performed experimental and numerical study by employing a five-story shear model of a building. They then showed both experimentally and numerically that a Kriging model utilizing the developed method could serve as a favourable substitute for the iterative FEA.

A new framework for solving high dimensional random partial differential equations (PDE) was proposed in Nabian and Meidani (2019). The considered random PDE was approximated by an ANN (deep, feed-forward, fully-connected), with either weak or strong enforcement of boundary and initial constraints. The proposed framework was mesh-free and could also deal with irregular computational domains. The correctness of the proposed method was shown on several heat conduction and diffusion problems for which numerical results were compared to the results computed by the Monte Carlo and FEM solutions. Schulz et al. (2019) introduced a numerical method for simulating Brownian polymer dynamics in a FEM framework, in which the Brownian polymer dynamics were described by stochastic PDEs driven by a white noise. To greatly improve the speed of the fitting process of the proposed method a Kriging surrogate model was used.

Nyshadham et al. (2019) compared several different surrogate models (ANNs, Kriging, response surfaces, etc.) for predicting the properties of different materials. They tested surrogate models which interpolates energies of various materials simultaneously on a data set of ten binary alloys and found that all the surrogate models agree (qualitatively) on prediction errors for the formation enthalpy, exhibiting relative errors of less than 2.5% for the considered systems. Another comparison of different surrogate models was carried out in Pavlíček et al. (2019). Kriging, ANN, and RF models were investigated on a problem of induction-assisted laser welding, which represented a rather complicated 3D problem. They also found that, if the models have properly tuned parameters, each of them can be successfully used as a surrogate for the FEM computations. Cernuda et al. (2020) presented a critical investigation of the appropriateness of different surrogate models for FEM application in the design of a suspension bushing. They compared linear SVR, RBF kernel SVR, nu SVR, generalized linear model with an elastic net, and RF surrogate models and found that RF is the most suitable among the compared surrogate models. Wang et al. (2021) devised a convolutional ANN surrogate model for accelerating screening materials and performance prediction. They showed that the proposed approach can predict the mechanical properties of a chosen collision problem with 98.63% accuracy, outperforming other techniques for surrogate modelling (SVM, RF, response surface, and ANN).

A computational method for simulations and monitoring-supported steering of tunnel boring machines in real time was introduced in Ninic et al. (2017). This strategy is combined a process-oriented 3D FEM with accompanying RNN surrogate model to recompute the parameters of the model based on the monitoring data and to give steering parameters that were continuously optimized. This hybrid FEM and surrogate model approach was compared with a strategy that was

entirely based on different surrogate models, and the hybrid method proved to be superior.

Bunnell et al. (2018) studied modeling different points on a FEM for compressor blades with unique surrogates and mesh morphing. Their results showed that mesh morphing gave good performance on various tested compressor blades—the surrogate model achieved error rates lower than 5%, while providing a 96% decrease in time needed for the structural iteration of the compressor blade.

An ANN-based technique for surrogate modeling suited for a nonlinear analysis of carbon-based nanotubes was presented in Papadopoulos et al. (2018). They proposed an ANN-based equivalent beam element which was able to accurately predict high order events that were the result of size-effects that outline the properties of carbon nanotubes at the nanoscale and could be estimated only by a micro-mechanical model. They conducted various numerical experiments for wavy and straight carbon nanotubes under compression and bending, and showed that the presented methodology was able to efficiently estimate the nonlinear responses of large-scale carbon nanotube structures needing only a fraction of time for the computation when compared to a full-scale problem.

Torkzadeh et al. (2016) used a cascade ANN (feed-forward) as the surrogate model for damage detection of plate-like structures. They developed a two-stage methodology. In stage one, location of the damages in plates were analyzed by the use of the concepts of curvature-moment and curvature-moment derivative. In stage two, in order to decrease the high computational cost of updating the FEM based model while detecting damage severity, multiple damage location criterion indexes that were grounded on the structural frequency change vector were computed by the ANN, whose structure was optimized using binary version of the bat algorithm.

In a medical applications, Liang et al. (2018) used an ANN (feed-forward, fully connected) with four hidden layers to construct a surrogate model of the zero-pressure geometry of human thoracic aortas. The surrogate model was validated by a cross-validation scheme on a data set of aorta shapes from 3125 virtual patients. They have found that the computed zero-pressure geometries gave good fit with the ones generated by the method based on FEA. While the FEA-based inverse method needed hours to days to finish the computations, the surrogate model was able to output the high precision zero-pressure geometry in a second. In a similar study, Liang et al. (2018) showed the ability of surrogates based on deep neural networks to predict the aortic wall stress distributions. The surrogate model was able to predict the stress distributions fairly accurately, exhibiting average errors of only 0.49 and 0.89% in the distribution of the Von Mises stress and the peak Von Mises stress, for a fraction of the computing costs of the full FEA. Vega and Todd (2020) applied a Bayesian ANN for cost-informed decision making and structural health monitoring in miter gates. They showed that a continuous monitoring via the proposed surrogate-assisted system could result in more economical decisions with respect to maintenance policies than using only the data from visual inspections. Yet another medical application of a Kriging-based surrogate model, this time for resonance frequency analyses of dental implants, was developed in Chu et al. (2019).

Berthelson et al. (2021) used Kriging and response surface surrogate models for investigating relative injury tendencies across various conditions of a vehicular impact. The surrogate modeling approach for injury analysis demonstrated great promise, requiring only 97 high fidelity FEM simulations to evaluate a wide range of motor vehicle collision scenarios exhibiting only a nominal prediction error, which saved substantial computational time and related costs, whilst enabling a thorough analyses of the influences of various collision conditions.

In the field of biomechanical engineering, Wee et al. (2016) developed response surface-based surrogate model of the biomechanics of a bone fracture fixation that was based on high numbers of FEM simulations. The developed surrogate models displayed a good fit with the results from FEA with R^2 score that ranged from 0.62 to 0.97.

The reviewed papers on surrogate modelling are summarized in Table 5, highlighting their choice of surrogates, and theoretical and application advancements.

3 Sensitivity analysis and uncertainty quantification

Sensitivity analysis (SA) is employed to determine the effect of the input parameters on a given outcome variable (Yang et al. 2016). It is often used as the preliminary step before an early design, analysis of uncertainty, or optimization to reduce problem complexity. Testing model sensitivity is an integral part of building any mathematical or simulation models. Different values of the parameters of a model as well as the initial (input) values of variables can be subject to various sources of uncertainties. Good comprehension of the sensitivity of the outputs of the model to various uncertainties in the values of the parameters and input variables is important for strengthening our confidence in our model and the resulting predictions (Alizadeh et al. 2020).

There are currently two complementary approaches for SA—local methods, and global methods. Local methods work by perturbing the inputs of one particular design in order to approximate its partial derivatives, which give sensitivities of inputs around the chosen design. Global methods aim to determine the effect of the parameters over the entirety of the design space. With the exception of methods for fast

Table 5 Considered literature on surrogate modelling, and properties of the models

References	Surrogates						Theoretical advancement	Application
	RSM	RBF	Kriging	ANN	SVR	Other		
Gogu and Passieux (2013)	×						Reduced basis model	Structural problems
Zhang and Au (2014)			×				N/A	Cable-stayed bridge
Jin and Jung (2016)			×				Finite element model updating	Structural problems
Torkzadeh et al. (2016)				×			N/A	Damage detection
Wee et al. (2016)	×						N/A	Bone fracture fixation
Ninic et al. (2017)				×			Hybrid FEM and surrogate model	Steering of tunnel boring machines
Sanchez et al. (2017)	×						Variable Power Law meta-model	Heat transfer problems
Bunnell et al. (2018)		×	×				N/A	Compressor blades
Liang et al. (2018)				×			N/A	Aortic wall stress distribution
Liang et al. (2018)				×			N/A	Human thoracic aortas
Papadopoulos et al. (2018)				×			N/A	Carbon nanotubes
Chu et al. (2019)			×				N/A	Analysis of dental implants
Ghorbel et al. (2019)				×			Adaptive run parameterization	Fluid dynamics
Nabian and Meidani (2019)				×			Mesh-free framework for solving PDEs	Heat conduction and diffusion problems
Schulz et al. (2019)			×				Novel numerical method	Brownian polymer dynamics
Nyshadham et al. (2019)	×		×	×			Benchmarking	Binary alloys
Pavlíček et al. (2019)			×	×		×	Benchmarking	Induction-assisted laser welding
Al Kajbaf and Bensi (2020)			×	×	×		Benchmarking	Estimation of storm surge
Cernuda et al. (2020)		×			×	×	Benchmarking	Suspension bushing
Lai et al. (2020)						×	Orthogonal decomposition	Parameter estimation
Vega and Todd (2020)				×			N/A	Structural health monitoring
Zhou and Lu (2020)			×				Dimension reduction technique	Cycle fatigue of compressor disc
Asteris et al. (2021)				×			N/A	Cement mortar materials
Brown et al. (2021)				×			N/A	Fluid dynamics
Berthelson et al. (2021)	×		×				N/A	Relative injury tendencies
Jin (2021)			×				Accelerated surrogate modeling	N/A
Wang et al. (2021)	×			×	×	×	Benchmarking	Material performance
Wang et al. (2022)			×				Annealing combinable Gaussian process	N/A

parameter prescreening, the methods of global analysis are generally more computationally demanding than the local methods. As both global and local methods are based upon simulating samples, faster evaluations of surrogate models can be used to speed up the process of sample generation, which is especially useful for variance-based techniques that require numerous samples.

On the other hand, SA also plays an important part in constructing surrogate models. By employing SA, promising inputs for the surrogate model can be found, which can

reduce the complexity of the model (Alizadeh et al. 2020). In problems where there is a rather complex surrogate model (such as a black-box FEM model), we can utilize SA alongside a surrogate model to get better knowledge about the behaviour of the model.

Whilst the goal of SA is the quantification of the effect of changing one input on the output, uncertainty quantification (UQ) (also called uncertainty analysis, uncertainty propagation, or reliability analysis) is used for investigating the likeliness of a change in the outputs that is induced by some uncertain inputs (Lee and Chen 2008). There has been a lot of work made to come up with methods of uncertainty quantification (UQ) in diverse fields such as stochastic mechanics, structural reliability, quality engineering, etc., and there is now a considerable number of techniques available.

The techniques for UQ can be categorized into five classes. The first class consists of the simulation-based techniques like Monte Carlo methods, and importance and adaptive sampling. The second class consists of the local expansion-based techniques such as the Taylor series expansion or perturbation techniques, which are generally weak when there is a large variability in the inputs and nonlinearities in the performance function (Lee and Chen 2008). The third class is the most probable point (MPP)-based techniques, which contain the first-order reliability methods and the second-order reliability methods. The fourth class comprises of techniques that are based on functional expansion. The PCE as well as the Neumann expansion method are found in this class. The PCE method has recently gained increased attention in stochastic mechanics, uncertainty representations, solution of stochastic differential equations, etc. The last class contains techniques that are based on numerical integration. One of the techniques in this classes is the dimension reduction technique (Rahman and Xu 2004), which works by approximating the (multidimensional) moment integrals by several (reduced-dimensional) integrals that are based upon additive decompositions of a performance function.

3.1 Overview of recent advances and applications in SA and UQ for FEM-based models

Eigel and Gruhlke (2021) developed an approach based on a domain decomposition scheme for high dimensional random PDEs which exploits the localization of the random parameters. The method uses hybrid local surrogates based on multielement generalized polynomial chaos expansion with the possibility to speed up the generation of samples by bypassing the assembly of operators and algebraic operations. They investigated the efficiency of the developed method on computational benchmark problems that illustrate the identification of nontrusted regions for sampling and trusted regions of the surrogate.

Gaspar et al. (2014) assessed the effectiveness of Kriging surrogates for problems in structural reliability which involve time-consuming nonlinear FEA models. The efficiency was investigates by systematically comparing the accuracy of the predictions of failure probability that were based on the first-order reliability methods that utilized the commonly used first-order and second-order polynomial RSM as well as Kriging models as the surrogates for the true expensive-to-compute limit state functions. They showed on a marine structures application that for structural reliability problems the Kriging models were efficient as surrogate models and, compared with the commonly used polynomial RSM, Kriging could provide substantially more accurate predictions of failure probability. This approach was further improved in Gaspar et al. (2017) by using a two-stage approach utilizing a trust region method.

In structural health monitoring, a technique for probabilistic prediction of the growth of a fatigue crack was developed in Leser et al. (2017). Prohibitive, time-consuming stress intensity factor computations were supplemented by efficient Kriging surrogate model that was trained on high-fidelity FEM simulations. Noisy visual assessments of the location history of the crack tip were utilized for the UQ of the model parameters. By utilizing the proposed surrogate modeling technique, they were able to reduce the simulation times by several orders of magnitude whilst maintaining high accuracy levels. Similar application of Kriging surrogates can be found in Su et al. (2017), where the authors developed a Monte Carlo method-based Dynamic Gaussian Process Regression surrogate for performing reliability analyses of complicated engineering structures.

Huang et al. (2016) applied a probabilistic method based on a surrogate model in the analyses of the effect of uncertainties in the process of deep drawing. They compared two types of surrogate models, quadratic response surface and Kriging, and found that Kriging models were more appropriate and accurate in modeling deep drawing processes.

The use of surrogate models has found numerous applications in the assessing the properties of composite structures. Omairey et al. (2019) developed a response surface-based multiscale surrogate model for efficient estimation of the stiffness properties of composite laminas, and also taking into account geometric and material uncertainty at laminate, meso, and micro-scale. Haeri and Fadaee (2016) proposed an advanced Kriging surrogate model for reliability analyses of laminated composites, employing a probabilistic classification function along with a metric for the refinement of the model. They demonstrated that the approach is significantly faster than using ANN-based surrogates, without sacrificing any predictive capabilities. Mukhopadhyay et al. (2016) presented the impact of noise on stochastic natural frequency analyses based on surrogate models of composite laminates. They developed an algorithm for exploring the

Table 6 Considered literature on sensitivity analysis and uncertainty quantification, and properties of the models

References	Surrogates						Theoretical advancement	Application
	RSM	RBF	Kriging	ANN	SVR	Other		
Gaspar et al. (2014)	×		×				Benchmarking	Structural reliability
Kersaudy et al. (2015)			×			×	Hybrid surrogate for UQ	Numerical dosimetry
Nobari et al. (2015)			×				N/A	Squeal noise of a real disc brake
Haeri and Fadaee (2016)			×	×			Benchmarking	Laminated composites
Huang et al. (2016)	×		×				Benchmarking	Deep drawing
Mukhopadhyay et al. (2016)			×				N/A	Composite laminate
Gaspar et al. (2017)	×		×				Two-stage approach	Structural reliability
Leser et al. (2017)			×				N/A	Structural health monitoring
Owen et al. (2017)			×			×	Benchmarking	N/A
Su et al. (2017)			×			×	Dynamic Gaussian process	Structural health monitoring
Tripathy and Bilionis (2018)				×			High dimensional UQ	Stochastic PDEs
Deng et al. (2020)						×	Drifted Wiener processes	Remaining useful lifetime prediction
Omairey et al. (2019)	×						Multiscale surrogate model	Composite laminas
Shi et al. (2019)	×						N/A	Vibration analysis
Slot et al. (2020)			×			×	N/A	Wind turbine reliability
Eigel and Gruhlke (2021)						×	Domain decomposition	N/A
Rocas et al. (2021)			×				Nonintrusive uncertainty quantification	Automotive crash problems
Wang et al. (2021)			×				Theory-guided neural network	N/A
Rocas et al. (2022)			×				Adaptive sampling methodology	Crashworthiness models
Ye et al. (2022)						×	N/A	In-stent restenosis

impact of noise in surrogate-based UQ methods and verified the approach for stochastic frequency analyses of spherical shallow shells using a surrogate model based on Kriging.

An application of Kriging-based surrogate models in vibration analyses of graphene sheets was proposed in Shi et al. (2019), conducting UQ for both Armchair and Zigzag graphene sheets. The LHS method was used to effectively propagate the uncertainty in material and geometrical features of the FEM-based model and the convergence and accuracy of the Kriging-based model were verified by comparing them with available references.

The reviewed papers on sensitivity analysis and uncertainty quantification are summarized in Table 6, highlighting their choice of surrogates, and theoretical and application advancements.

4 Surrogate-assisted optimization

The state-of-the-art in solving costly and complex problems that arise in real-world applications involves the utilization of surrogate models during optimization (Stork et al. 2020), i.e. in the search of the configuration of the design variables that produces the most desirable (optimal) outcome that is measured by a so-called objective function. The FEM-based models belong to a category of so-called black-box problems, in which the problem information that is available, such as mathematical equations and/or other exploitable knowledge of the problem is very limited, and the only method of extracting any information is the costly evaluation of the candidate designs.

The main goal of surrogate-assisted optimization (SAO) lies in the reduction of the resources, time, and the related costs by exploiting all information that is available efficiently to lower the number of objective function evaluations that are needed. Often, this is achieved by using only a few of the expensive true function evaluations to construct a "rough" surrogate model and running an optimization algorithm on this surrogate. The optimal solution from this computation is then used as a next point for another expensive true function evaluation and the refinement of the surrogate model. This process is then repeated until a stopping criterion (such as number of iteration, computational time, "good enough solution", precision of the surrogate, etc.) is met.

The optimization problems where the information about the derivative of the function is not symbolically nor numerically available are commonly categorized as derivative-free optimization (DFO) problems (Rios and Sahinidis 2013). Algorithms for DFO problems can be categorized into local search and global search methods. Local search algorithms are used to refine a solution or to reach a local optimum from an initial point. On the other hand, global search methods employ a mechanism that allows them to escape from local minima.

In the category of the local search methods are the direct search optimization algorithms that sequentially evaluate candidate points that are generated by a particular strategy (which often utilize geometric patterns), such as the Hooke and Jeeve's algorithm (Hooke and Jeeves 1961) and the Nelder-Mead (NM) method (Nelder and Mead 1965). Trust-region methods are also in the category of local search methods which use a surrogate model in a close neighbourhood of a given sample location. Another local search method is sequential quadratic programming (SQP), which at each iteration constructs a quadratic approximation of the optimization problem and finds the corresponding solution (Nocedal and Wright 2006).

In global search methods, we can find the partitioning methods such as the DIRECT algorithm (Jones et al. 1993), and stochastic algorithms. The stochastic algorithms have become especially popular for SAO in recent years (Jin et al. 2019), with methods such as simulated annealing (SiA) (Kirkpatrick et al. 1983), and evolutionary algorithms (EA) (Baeck et al. 1997) such as genetic algorithms (GA) (Goldberg and Holland 1988), particle swarm optimization (PSO) (Poli et al. 2007), differential evolution (DE) Storn and Price (1997) and many others (Matousek et al. 2022). One of the drawbacks of using stochastic algorithms is the need for proper tuning of their respective hyperparameters (Kazikova et al. 2020).

Highly challenging optimization problems are in many cases concerned with more than a single objective function. The multiple objectives (MO) are commonly aggregated into a single objective function by a weighted sum (or similar aggregation function) which makes it approachable by using ordinary (single objective) optimization methods. A different approach is to take all objectives into consideration in parallel which is particularly important if these objectives are conflicting, such as price and quality in production or drag and lift in airfoil design (Stork et al. 2020).

Although several algorithms for multi-objective SAO have already been developed, the field still lacks a repository where the different approaches could be collected and compared, because the development of these methods tends to be application-oriented and these methods have commonly been used to solve a particular real-word or industrial optimization task. Of the algorithm that are more widely used are the multiobjective genetic algorithm NSGA-II (Deb et al. 2002), the ParEGO algorithm (Knowles 2006), and the RASM method (Loshchilov et al. 2010).

As uncertainty plays a significant role in design, incorporating ways of dealing with uncertainty into the optimization process is often an important step that guarantees that the resulting optimal design can handle variations in input parameters. One possibility is to apply SA to the result obtained by the optimization process, and, if found inadequate, modifying the objective function or the constraints. Another possibility is to use robust optimization (RO), which gives a mathematical framework for optimization which is designed to minimize the propagation of the input uncertainties to the output responses (Chatterjee et al. 2019).

4.1 Overview of recent advances and applications in SAO for FEM-based models

In this subsection, we review the recent application of SAO. Table 7 gives a summary of the publications including surrogate model type, optimization methodology, and application.

4.1.1 Surrogate model types

The most ubiquitous choice of the surrogate model in applications that use surrogate-assisted optimization is Kriging, followed by RSM and ANN. Especially in the last few year, deep ANN are gaining increased popularity (Abueidda et al. 2020). Both kernel-based methods, RBF and SVR, were used only in a few applications, and PCE and RF were not used at all. GP was used by Mendes et al. (2013) and Easum et al. (2017) to build the surrogate, and in both instances the optimization was carried out by a GA. Li et al. (2019) developed a novel adaptive Singular Value Decomposition (SVD)-Krylov reduced order model as a surrogate for solving problems in structural optimization. Utilizing the SVD, they show that for a structural optimization problem the solution space of can be partitioned into a design subspace and a geometry subspace, which can be effectively approximated by a collection of different surrogate models. They show on a set of

Table 7 Considered literature on surrogate-assisted optimization, and properties of the models

References	Surrogates								Optimization			Application
	RSM	RBF	Kriging	ANN	SVR	Other	EoS	MF	MO	Uncertainty	Algorithm	
Mendes et al. (2013)						GP			×	RO	GA	Magnetostatics
Leifsson et al. (2015)	×										Trust-region	Multi-element trawl-doors
Lim et al. (2015)			×						×		Specialized	IPM synchronous motor
Tan et al. (2015)	×										PSO	Induction generator
Bramerdorfer and Zăvoianu (2017)	×			×			×		×	SA	EA	Electrical machine
Christelis et al. (2017)		×									EA	Coastal aquifers
Easum et al. (2017)	×	×				GP	×		×		GA	Patch antenna
Lal and Datta (2017)					×				×		GA	Coastal aquifers
Li et al. (2017)			×								EI	Stent and dilatation balloon
Shi et al. (2017)				×					×		GA	Ceramic heat exchanger
Shi et al. (2017)			×		×		×		×		SiA	Dental implant
Shi et al. (2017)	×										SQP	GEO satellites
Wu et al. (2017)			×								Specialized	Fluid-structure interaction
Bonfiglio et al. (2018)			×					×		UQ	EI	Supercavitating hydrofoils
Hassan et al. (2018)			×						×	SA	PSO	nanoantenna
Kaya and Hajimirza (2018)				×						SA	SiA	Organic solar cells
Liu et al. (2018)			×							UQ	PSO	Composite battery box
Park et al. (2018)			×							RO	Specialized	IPM motor
Pfrommer et al. (2018)				×							DE	Textile manufacturing
Putra et al. (2018)			×						×		EI	Stent geometry
Qin et al. (2018)			×							SA	PSO	Bridge structures
Qiu et al. (2018)	×									RO	PSO	Crashworthiness
Silber et al. (2018)				×					×		GA	Electric machines
Taran et al. (2018)			×						×		DE	Axial flux machines
Wang et al. (2018)			×						×	UQ	EI	Polymer nanocomposites
Benaouali and Kachel (2019)	×										GA	Aircraft wing
Fan et al. (2019)			×							RO	Specialized	Crane bridges
Li et al. (2019)						SVD					Specialized	Structural shape
Lin et al. (2019)		×								UQ	EA	Mooring system
Meng et al. (2019)	×									UQ	Specialized	Wind turbine blades
Rafiee and Faiz (2019)	×									RO	PSO	Outer rotor magnet

Table 7 continued

References	Surrogates								Optimization			Application
	RSM	RBF	Kriging	ANN	SVR	Other	EoS	MF	MO	Uncertainty	Algorithm	
Tan et al. (2019)			×								DE	Heat transfer coefficients
Watts et al. (2019)	×									SA	Specialized	Multiscale topology
White et al. (2019)				×							Specialized	Multiscale topology
Xu et al. (2019)			×								Specialized	Wind turbine blade
Yong et al. (2019)			×					×		SA	EI	Whole engine models
Zhao et al. (2019)				×					×		GA	Viscous damper system
Abueidda et al. (2020)				×						SA	Specialized	Topology of 2D structures
Bouhlel et al. (2020)	×			×			×				SQP	Airfoil shape
Dong et al. (2020)				×							NM	Elastic metamaterials
Qian et al. (2020)	×										GA	Engineering design
Tie et al. (2020)		×								RO	Specialized	CFPR laminates
Yan et al. (2020)					×						GA	Aero-engine turbine
Yoo et al. (2020)		×						×	×	UQ	GA	Composite structures
Fatahi (2021)			×						×	SA	GA	Welded plates
Ktari et al. (2021)	×								×		Specialized	Ring tensile specimen

numerical examples that the proposed methodology exhibits performance gains when compared to most already existing heuristic techniques.

The use of EoS and MF models is also not as wide-spread as one would expect, given their advantageous properties. EoS were used in conjunction with MO problems in Bramerdorfer and Zăvoianu (2017), where the authors used different global and local surrogate models for optimizing electric machine design, in Easum et al. (2017), where the EoS based method performed better than the traditional NSGA-II on a optimization of a patch antenna design, and in the design of dental implants in Shi et al. (2017). Bayesian optimization was used in maximizing the EI of MF surrogates for a realistic large-scale hydrostructural optimization of 3D supercavitating hydrofoils in Bonfiglio et al. (2018). MF surrogates were also used for solving realistic design problems concerning whole gas turbine engines Yong et al. (2019) and reliability-based optimization of various composite structures (Yoo et al. 2020).

4.1.2 Optimization models and algorithms

The consideration of multiple objective in FEM-based optimization problems is relatively widespread, appearing in around a third of the considered publications. The multiple objective have various forms depending on the particular application and are frequently connected to buckling load and structural mass in structural design applications (Yoo et al. 2020), response reduction ratio and value of a damping force in damper systems (Zhao et al. 2019), wall shear stress and initial average stress in stent geometry optimization (Putra et al. 2018), gain, front-to-back ratio, and ground plane area for patch antenna design (Easum et al. 2017), etc.

Almost a half of the considered publications included some form of dealing with uncertainty in the optimization model. UQ was frequently used together with the Kriging surrogate in Bonfiglio et al. (2018), Liu et al. (2018), and Wang et al. (2018). Applications utilizing SA frequently used ANN (Bramerdorfer and Zăvoianu 2017; Kaya and Hajimirza 2018; Abueidda et al. 2020) or Kriging (Hassan et al. 2018; Qin et al. 2018; Yong et al. 2019; Fatahi 2021) surrogates.

Stochastic optimization methods, be it SiA or evolutionary approaches (EA, GA, PSO), were the algorithm of choice for more than a half of the considered publications, which might be attributed to their relatively low implementation complexity and satisfactory performance. More interestingly, a growing number of authors used a specialized algorithm in their applications. Wu et al. (2017) used the surrogate management framework algorithm Booker et al. (1998), a mesh-based method that contains two strategies: a surrogate model as a tool for prediction to facilitate global exploration and to identify promising regions, and the use a local grid search or similar pattern-search techniques to guarantee convergence at least to local minima. Park et al. (2018) proposed a robust optimization method that uses a sub-domain Kriging to decrease the memory allocations during the optimization and a gradient-free sensitivity index, that measures the sensitivity of the fitness function value to the input variables. Meng et al. (2019) used collaborative optimization method based on uncertainties, which is a bi-level multidiscipline design optimization method, to approach the turbine blades design problem and to improve its aerodynamic performance.

4.1.3 Application areas

The employment SAO has permeated into a wide range of applied areas. In electrical engineering, these models are now quite routinely used to aid the design of IPM (Interior Permanent Magnet) motors (Lim et al. 2015; Park et al. 2018; Rafiee and Faiz 2019) and axial flux machines (Taran et al. 2018), induction generators (Tan et al. 2015), composite battery boxes (Liu et al. 2018), all-electric GEO satellites (Shi et al. 2017), or antennas (Easum et al. 2017; Hassan et al. 2018). Also ubiquitous are mechanical engineering applications, which include the design of wind turbine blades (Meng et al. 2019; Xu et al. 2019), aero-engine turbines (Yan et al. 2020), viscous damper systems (Zhao et al. 2019), welded plates (Fatahi 2021), ring tensile specimens (Ktari et al. 2021), or whole engine models (Yong et al. 2019), and civil engineering applications of topology optimization (Qin et al. 2018; White et al. 2019; Fan et al. 2019; Li et al. 2019; Abueidda et al. 2020).

Recently, SAO has been used in maritime applications. Leifsson et al. (2015) performed shape optimization of multi-element trawl-doors, which are for many fishing vessels the major contributors to their fuel expenditure, as they may be responsible for approximately 10–30% of the total drag of the vessel. Lin et al. (2019) presented a scantling optimization of a common internal turret area for mooring system of floating production storage and offloading units, which constitute some of the most widely used production platforms in offshore oil production. They report on achieving a weight reduction of 10.2%, computed by a SAO utilizing the RBF surrogate model and an EA. Lal and Datta (2017) investigated the viability and efficiency of utilizing artificial freshwater recharge in order to increase the pumping of fresh groundwater from wells. They used a SVR as a surrogate for the expensive simulations and a multi-objective GA for finding optimal solutions for recharge and integrated pumping, and for maintaining the levels of saltwater intrusion in the coastal aquifer within acceptable limits. Similar problem was investigated in Christelis et al. (2017), where the authors used the RBF surrogate and a single objective formulation solved by an EA instead. Their results demonstrated an outperformance of the SAO methods against the direct optimization, as their

method located the best solutions and demonstrated a robust performance for all optimization problems of coastal aquifer management.

Shi et al. (2017) developed a geometric model of a ceramic microchannel heat exchanger which included the heat transfer channel, the inlet part, and outlet part and conducted a numerical study to enhance the uniformity of the fluid flow. Then the authors constructed a radial basis neural network surrogate model and optimized the heat exchanger design using a GA. At the optimal design point, they report that the nonuniformity of the fluid flow was decreased by 68.2% and pressure drop has increased by 6.6%, which results in a significant improvement in the uniformity of the fluid flow in the heat exchanger with just a little cost of pressure drop. Another thermal engineering application can be found in Tan et al. (2019), where the authors used the Kriging surrogate and the DE algorithm to compute more accurate convective heat transfer coefficients in thermal analysis of spindle. They compared the results of the simulation temperatures using the optimized convective heat transfer coefficients with the experimental temperatures and found that they agreed well exhibiting the absolute error of the simulation not exceeding 0.5 °C and the relative error of the simulation not exceeding 2.34%.

Simple UQ method for evaluating numerical uncertainty as well as surrogate uncertainty in a crashworthiness optimization process was proposed in Qiu et al. (2018). They showed that the conventionally used 95% confidence interval was insufficient for obtaining robust results especially when encountering high levels of noise in the simulations because the commonly used number of samples for building surrogates is too small for reaching accurate estimates of the noise level.

Kaya and Hajimirza (2018) demonstrated that surrogates can be utilized for accurate predictions of the optical properties of thin solar cells and even for the optimization their structures. Instead of the time-consuming finite difference time domain methods for computing optical properties of arrangements at small sub-wavelength scales, they designed a two-layer ANN surrogate model for estimating the optical absorptivity of the cell. They then used a combination of steepest descend and SiA for the optimization of the cell parameters. The solutions found by SAO demonstrated enhancement factors that were higher than 270% for the optical absorptivity.

The use of SAO is also becoming more common in medical applications. Design optimization of stents and its dilation balloons, which are used in treating cardiovascular diseases, has investigated in Li et al. (2017). By using a Kriging surrogate and SAO, they were able to refine the fatigue life and expansion performance of both diamond-shaped and sv-shaped stents. Stent geometry was also the focus of Putra et al. (2018). SAO with expected hypervolume improvement and Kriging method were used to construct the surrogate model and to find the best configuration of parameters to the intravascular hemodynamics of the stent. In Shi et al. (2017) the authors used EoS and SAO in a dental application to reduce stress at the implant-bone interface to improve the implantation success rate by using the structure optimization of dental implant with other characteristics of dental implant only slightly deteriorating or optimizing simultaneously.

SAO also found recent applications in identifying interphase properties the properties in polymer nanocomposites (Wang et al. 2018), optimizing elastic metamaterial structures (Dong et al. 2020), optimizing manufacturing process parameters using deep ANN (Pfrommer et al. 2018), or maximizing the impact-resistance of patch repaired carbon fiber reinforced polymer laminates (Tie et al. 2020).

5 Software tools

As the employment of surrogates for analyzing and optimizing computationally expensive problems have become more prevalent, new software tools that provide an easy access to the needed technologies emerged. Few of the most used and recently developed software tools are listed below in alphabetical order:

- *ALAMO* (Cozad et al. 2014): ALAMO (Automated learning of algebraic models for optimization) is a methodology for classification and regression that aims to build accurate and simple surrogates based on a minimal collection of datapoints. It uses a technique based on integer programming to select from a high number of input variables and their possible transformations.
- *ARGONAUT* (Boukouvala and Floudas 2017): ARGONAUT is a framework designed to deal with DFO problems with either a total or a partial lack of closed form or analytical expressions for objective function or constraints. Pivotal feature of this framework is also surrogate model selection in which the surrogate models are selected from a library of regression models (including linear, quadratic and polynomial RSM) which are straightforward to optimize and various interpolation models (including RBF and Kriging) that have better data prediction accuracy.
- *Agros Suite and Ārtap* (Karban et al. 2021): Agros Suite is an environment for numerically solving second order PDEs by a higher-order FEM with a variety of other advanced features, including various optimization techniques and full adaptivity. Ārtap, a Python toolbox for robust design optimization provides an efficient and simple programming environment that encompases a broad-range of integrated as well as external PDE solvers,

optimization methods and well-known machine learning techniques for building surrogate models.

- *Eureqa* (Schmidt and Lipson 2009): Eureqa is a software (commercially available) for building surrogate models. Its procedures start with the initial data set and follow by computing symbolic regressions in which the search is not only bounded to the coefficients but also to regression model forms as well (similar to GP). It then constructs a collection of candidate regression models, where the precision of the models is evaluated by computing symbolic differentiation of the models and by comparing their derivatives with the initial dataset. On the basis of these computations, the models are sequentially recomputed until a convergence criterion is met.
- *FReET* (Novak and Novak 2018): FReET (Feasible Reliability Engineering Tool) is a multipurpose probabilistic software for reliability and sensitivity analysis of various problems in engineering, with incorporated PCE surrogate modelling enabling sensitivity and reliability analysis.
- *MATLAB toolboxes*: MATLAB has a dedicated Statistics and machine learning toolbox which supports subset selection employing goodness-of-fit measures, linear and nonlinear regression, regularization, and SVR. Validation metrics for studying the performance of classification algorithms, non-parametric regression algorithms and surrogates, are also included.
- *MATSuMoTo* (Mueller 2014): MATSuMoTo is the MATLAB Surrogate Model Toolbox for black-box, computationally expensive, global optimization problems, which offers various choices for initial experimental design strategies, sampling strategies, surrogate models and surrogate model mixtures. MATSuMoTo can also compute several function evaluations in parallel by utilizing the Parallel Computing Toolbox available in MATLAB.
- *Python toolboxes*: One of the options in Python is the SMT toolbox (Bouhlel et al. 2019), which contains several surrogate modelling methods, PyDOE toolbox offers a collection of different static sampling methods, and the well-known ScikitLearn (Pedregosa et al. 2011), Tensorflow (Abadi et al. 2016) and PyTorch (Paszke et al. 2017) all contain different model validation schemes and surrogate model types.
- *RBFOpt* (Costa and Nannicini 2018): RBFOpt is an open-source library for optimizing a black-box function over a mixed-integer box-constrained set. The algorithm is based on the RBF surrogate and performs a fast procedure for automatic model selection.
- *Surrogate Modeling (SUMO) Toolbox* (Gorissen et al. 2010): SUMO is a MATLAB toolbox for adaptive sampling and surrogate modeling which offers several surrogate model choices (Kriging, SVM, ANN, etc.), model selection algorithms, DOE methods, sample evaluation methods, and optimization algorithms (PSO, EI, SiA, GA, etc.).
- *Surrogates Toolbox* (Viana and Goel 2010): Another general purpose MATLAB toolbox for building surrogate models which contains several third-party software packages and has four primary capabilities: DOE, surrogate model construction (Kriging, RBF, etc.), model validation, and optimization along with sensitivity analysis.
- *SurroOpt* (Han 2016): SurroOpt is a research code developed for engineering designs and academic research guided by expensive numerical simulations. A notable feature is that constructing the surrogate and solving optimization problems that correspond to the infill-sampling criteria are looked at as new optimization mechanisms, the role of which is the same as any of the normal gradient-based techniques or stochastic optimization algorithms.

6 Trends, research gaps, and practical recommendations

The presented review affirms that surrogate modelling is a well grounded methodology in current research of FEM-based analysis and optimization of various systems. Performance and sensitivity analysis, uncertainty quantification, and also surrogate assisted design optimization become more accessible, mainly because of the huge reductions in computational costs. In the following text, we discuss the practical aspects and application trends that were extracted from the considered literature.

- The usefulness of surrogate models is mainly connected to the reduction in computational resources and time needed for certain analyses whilst keeping high accuracy levels. Even though there exist many examples showing advantageous use of surrogate models for UQ and SA, there still remains a gap in the knowledge on the extent of achievable time savings. The time savings were thoroughly analysed primarily in papers dealing with SAO.
- Researchers have been searching for more generalized surrogates that could be applied to various dissimilar problems. Multiple publications investigated the maximum capabilities of single surrogates, while the focus on the automation of the process of deriving surrogates (such as in ALAMO), and the employment of EoS or MF models is still underutilized.
- Most types of surrogate models have low interpretability of the underlying mathematical structures and, as such, are ill suited for answering any analytical questions.
- A common issue is the restricted number of design variables that surrogate models can deal with without suffering prohibitive computational cost. To this end, it

became popular to incorporate sensitivity analysis to the process of the derivation of the surrogate model to find the parameters that are most important. Dimension reduction techniques are also becoming popular for reducing the number of considered inputs.
- The proper choice of the initial sampling scheme is also uncertain. Most of the reviewed papers used LHS.
- In the cases where the size of the problem is higher, the required accuracy levels are lower, while the time for computations remains unchanged, RBF and RSM seem to be the appropriate choice of surrogates. Kriging is a method that displays in high accuracy levels and relatively good performance for larger problem sizes, but is not well suited for problems that have more than roughly 50 variables. For high-dimensional problems, SVM performs well even with high levels of nonlinearity.
- For SA and UQ, the most widely used surrogates in recent applications were Kriging and PCE.
- Using off-the-shelf optimization algorithms, such as NSGA-II, for finding the optimal surrogate-based design is a common practice, but building an application-oriented method can bring significant improvements.
- There now exists a wide range of readily available software tools for both building and validating surrogate models, and for SAO.

7 Conclusion

Surrogate models are becoming increasingly more and more utilized in multiple scientific and engineering disciplines and found applications in various fields. On the other hand, the selection of the appropriate surrogate for the given problem is not straightforward due to various trade-offs that are associated with using the different surrogates. This choice is a bit clearer if the given problem can be classified as model building/prediction, sensitivity analysis or uncertainty quantification, or optimization. The differences in the various techniques for every one of these categories from point of view of surrogate modeling were reviewed and relevant current advances and new applications in the different categories were investigated with the focus on surrogates. There now exist several software tools that give the user an easy entry to the techniques that were investigated in this paper. These tools were mentioned with a short explanation of their respective aims and capabilities.

We anticipate, that future research in surrogate models for FEM-based computations will focus more on developing automated tools for selection and construction of surrogates, as well as an efficient use of EoS and MF models, based on where in the three classes the particular application is located. We also expect future analyses to concentrate further on decreasing computational cost related to deriving surrogate models and on improving their interpretability.

Acknowledgements This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic project No. CZ.02.1.01/0.0/0.0/16_026/0008392 "Computer Simulations for Effective Low-Emission Energy" and by IGA BUT: FSI-S-20-6538.

Funding The Ministry of Education, Youth and Sports of the Czech Republic project No. CZ.02.1.01/0.0/0.0/16_026/0008392 "Computer Simulations for Effective Low-Emission Energy" and by IGA BUT: FSI-S-20-6538.

Availability of data and materials Not applicable.

Code Availability Not applicable.

Declarations

Conflict of interest We confirm that this manuscript is the authors' own original work, has not been published and is not under consideration for publication elsewhere. We declare that the authors have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abadi M, Barham P, Chen J et al (2016) Tensorflow: a system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp 265–283

Abueidda DW, Koric S, Sobh NA (2020) Topology optimization of 2d structures with nonlinearities using deep learning. Comput Struct 237(106):283. https://doi.org/10.1016/j.compstruc.2020.106283

Acar E (2015) Effect of error metrics on optimum weight factor selection for ensemble of metamodels. Expert Syst Appl 42(5):2703–2709. https://doi.org/10.1016/j.eswa.2014.11.020

Al Kajbaf A, Bensi M (2020) Application of surrogate models in estimation of storm surge: a comparative assessment. Appl Soft Comput 91(106):184

Alizadeh R, Allen JK, Mistree F (2020) Managing computational complexity using surrogate models: a critical review. Res Eng Des 31:275–298. https://doi.org/10.1007/s00163-020-00336-7

Asteris PG, Cavaleri L, Ly HB et al (2021) Surrogate models for the compressive strength mapping of cement mortar materials. Soft Comput 25(8):6347–6372

Babaei M, Pan I (2016) Performance comparison of several response surface surrogate models and ensemble methods for water injection optimization under uncertainty. Comput Geosci 91:19–32. https://doi.org/10.1016/j.cageo.2016.02.022

Baeck T, Fogel D, Michalewicz Z (1997) Handbook of evolutionary computation. Taylor & Francis, London

Benaouali A, Kachel S (2019) Multidisciplinary design optimization of aircraft wing using commercial software integration. Aerosp Sci Technol 92:766–776. https://doi.org/10.1016/j.ast.2019.06.040

Berthelson P, Ghassemi P, Wood JW et al (2021) A finite element-guided mathematical surrogate modeling approach for assessing occupant injury trends across variations in simplified vehicular impact conditions. Med Biol Eng Comput 59:1065–1079. https://doi.org/10.1007/s11517-021-02349-3

Bhosekar A, Ierapetritou M (2018) Advances in surrogate based modeling, feasibility analysis, and optimization: a review. Comput Chem

Eng 108:250–267. https://doi.org/10.1016/j.compchemeng.2017.09.017
Bischl B, Mersmann O, Trautmann H et al (2012) Resampling methods for meta-model validation with recommendations for evolutionary computation. Evol Comput 20:249–275. https://doi.org/10.1162/EVCO_a_00069
Bonfiglio L, Perdikaris P, del Águila J et al (2018) A probabilistic framework for multidisciplinary design: application to the hydrostructural optimization of supercavitating hydrofoils. Int J Numer Meth Eng 116(4):246–269. https://doi.org/10.1002/nme.5923
Booker A, Dennis J, Frank P et al (1998) A rigorous framework for optimization of expensive functions by surrogates. Struct Optim 17:1–13. https://doi.org/10.1007/BF01197708
Bouhlel M, He S, Martins J (2020) Scalable gradient-enhanced artificial neural networks for airfoil shape design in the subsonic and transonic regimes. Struct Multidiscip Optim 61:1363–1376. https://doi.org/10.1007/s00158-020-02488-5
Bouhlel MA, Hwang JT, Bartoli N et al (2019) A python surrogate modeling framework with derivatives. Adv Eng Soft. https://doi.org/10.1016/j.advengsoft.2019.03.005
Boukouvala F, Floudas CA (2017) Argonaut: Algorithms for global optimization of constrained grey-box computational problems. Optim Lett 11(5):895–913. https://doi.org/10.1007/s11590-016-1028-2
Box E, Draper N (1987) Empirical model building and response surfaces. Wiley, New York
Bramerdorfer G, Zăvoianu AC (2017) Surrogate-based multi-objective optimization of electrical machine designs facilitating tolerance analysis. IEEE Trans Magn 53(8):1–11. https://doi.org/10.1109/TMAG.2017.2694802
Broomhead D, Lowe D (1988) Multivariable functional interpolation and adaptive networks. Complex Syst 2:321–355
Brown A, Montgomery J, Garg S (2021) Automatic construction of accurate bioacoustics workflows under time constraints using a surrogate model. Appl Soft Comput 113(107):944
Bunnell S, Thelin C, Gorrell S et al (2018) Rapid visualization of compressor blade finite element models using surrogate modeling. p v07AT30A011. https://doi.org/10.1115/GT2018-77188
Cernuda C, Llavori I, Zăvoianu AC et al (2020) Critical analysis of the suitability of surrogate models for finite element method application in catalog-based suspension bushing design. In: 2020 25th IEEE international conference on emerging technologies and factory automation (ETFA), pp 829–836. https://doi.org/10.1109/ETFA46521.2020.9212166
Chatterjee T, Chakraborty S, Chowdhury R (2019) A critical review of surrogate assisted robust design optimization. Arch Comput Methods Eng 26:245–274. https://doi.org/10.1007/s11831-017-9240-5
Chen H, Loeppky JL, Sacks J et al (2016) Analysis methods for computer experiments: how to assess and what counts? Stat Sci 31(1):40–60. https://doi.org/10.1214/15-STS531
Christelis V, Regis RG, Mantoglou A (2017) Surrogate-based pumping optimization of coastal aquifers under limited computational budgets. J Hydroinf 20(1):164–176. https://doi.org/10.2166/hydro.2017.063
Chu L, Shi J, de Cursi ES (2019) Kriging surrogate model for resonance frequency analysis of dental implants by a Latin hypercube-based finite element method. Appl Bionics Biomech. https://doi.org/10.1155/2019/3768695
Costa A, Nannicini G (2018) RBFOpt: an open-source library for black-box optimization with costly function evaluations. Math Program Comput 10:597–629. https://doi.org/10.1007/s12532-018-0144-7
Cozad A, Sahinidis N, Miller D (2014) Learning surrogate models for simulation-based optimization. AIChE J 60:2211–2227. https://doi.org/10.1002/aic.14418
Crombecq K, Laermans E, Dhaene T (2011) Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. Eur J Oper Res 214:683–696. https://doi.org/10.1016/j.ejor.2011.05.032
Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: Nsga-II. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017
Deng Y, Di Bucchianico A, Pechenizkiy M (2020) Controlling the accuracy and uncertainty trade-off in RUL prediction with a surrogate wiener propagation model. Reliab Eng Syst Saf 196(106):727
De'ath G (2007) Boosted trees for ecological modeling and prediction. Ecology 88(1):243–51. https://doi.org/10.1890/0012-9658(2007)88[243:BTFEMA]2.0.CO;2
Dong J, Qin Q, Xiao Y (2020) Nelder–Mead optimization of elastic metamaterials via machine-learning-aided surrogate modeling. Int J Appl Mech 12(2050):011. https://doi.org/10.1142/S1758825120500118
Eason J, Cremaschi S (2014) Adaptive sequential sampling for surrogate model generation with artificial neural networks. Comput Chem Eng 68:220–232. https://doi.org/10.1016/j.compchemeng.2014.05.021
Easum JA, Nagar J, Werner DH (2017) Multi-objective surrogate-assisted optimization applied to patch antenna design. In: 2017 IEEE international symposium on antennas and propagation USNC/URSI national radio science meeting, pp 339–340. https://doi.org/10.1109/APUSNCURSINRSM.2017.8072212
Eigel M, Gruhlke R (2021) A local hybrid surrogate-based finite element tearing interconnecting dual-primal method for nonsmooth random partial differential equations. Int J Numer Methods Eng 122(4):1001–1030. https://doi.org/10.1002/nme.6571
Fan X, Wang P, Hao F (2019) Reliability-based design optimization of crane bridges using kriging-based surrogate models. Struct Multidisc Optim 59:993–1005. https://doi.org/10.1007/s00158-018-2183-0
Fatahi L (2021) Surrogate-based sensitivity analysis and finite element model updating of welded plates. Mech Adv Mater Struct. https://doi.org/10.1080/15376494.2021.1907006
Forrester A, Keane A (2009) Recent advances in surrogate-based optimization. Prog Aerosp Sci 45:50–79. https://doi.org/10.1016/j.paerosci.2008.11.001
Friedman J (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232
Garud S, Karimi I, Kraft M (2017) Smart sampling algorithm for surrogate model development. Comput Chem Eng 96:103–114. https://doi.org/10.1016/j.compchemeng.2016.10.006
Gaspar B, Teixeira A, Soares CG (2014) Assessment of the efficiency of kriging surrogate models for structural reliability analysis. Probab Eng Mech 37:24–34. https://doi.org/10.1016/j.probengmech.2014.03.011
Gaspar B, Teixeira A, Guedes Soares C (2017) Adaptive surrogate model with active refinement combining kriging and a trust region method. Reliab Eng Syst Saf 165:277–291. https://doi.org/10.1016/j.ress.2017.03.035
Ghorbel H, Zannini N, Cherif S et al (2019) Smart adaptive run parameterization (SARP): enhancement of user manual selection of running parameters in fluid dynamic simulations using bio-inspired and machine-learning techniques. Soft Comput 23(22):12031–12047
Goel T, Haftka R, Shyy W et al (2007) Ensemble of surrogates. Struct Multidiscip Optim 33:199–216. https://doi.org/10.1007/s00158-006-0051-9
Gogu C, Passieux JC (2013) Efficient surrogate construction by combining response surface methodology and reduced order modeling.

Struct Multidiscip Optim 47:821–837. https://doi.org/10.1007/s00158-012-0859-4
Goldberg D, Holland J (1988) Genetic algorithms and machine learning. Mach Learn 3:95–99. https://doi.org/10.1023/A:1022602019183
Gorissen D, Couckuyt I, Demeester P et al (2010) A surrogate modeling and adaptive sampling toolbox for computer based design. J Mach Learn Res 11:2051–2055
Grama A, Kumar V, Gupta A et al (2003) Introduction to parallel computing. Addison-Wesley, Pearson Education, Reading
Gutmann HM (2001) A radial basis function method for global optimization. J Glob Optim 19:201–227. https://doi.org/10.1023/A:1011255519438
Hadigol M, Doostan A (2018) Least squares polynomial chaos expansion: a review of sampling strategies. Comput Methods Appl Mech Eng 332:382–407. https://doi.org/10.1016/j.cma.2017.12.019
Haeri A, Fadaee MJ (2016) Efficient reliability analysis of laminated composites using advanced kriging surrogate model. Compos Struct 149:26–32. https://doi.org/10.1016/j.compstruct.2016.04.013
Han Z (2016) SurroOpt: a generic surrogate-based optimization code for aerodynamic and multidisciplinary design. In: Proceedings of the 30th congress of the international council of the aeronautical sciences, DCC, Daejeon, Korea, 25–30
Hassan AKS, Etman AS, Soliman EA (2018) Optimization of a novel nano antenna with two radiation modes using kriging surrogate models. IEEE Photonics J 10(4):1–17. https://doi.org/10.1109/JPHOT.2018.2848593
Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer, New York
Ho T (1998) The random subspace method for constructing decision forests. IEEE Trans Pattern Anal Mach Intell 20:832–844. https://doi.org/10.1109/34.709601
Hooke R, Jeeves TA (1961) "Direct search" solution of numerical and statistical problems. J ACM 8(2):212–229. https://doi.org/10.1145/321062.321069
Huang C, Radi B, Hami A (2016) Uncertainty analysis of deep drawing using surrogate model based probabilistic method. Int J Adv Manuf Technol 86:3229–3240. https://doi.org/10.1007/s00170-016-8436-4
Hung Y (2011) Penalized blind kriging in computer experiments. Stat Sin 21(3):1171–1190
Jin SS (2021) Accelerating gaussian process surrogate modeling using compositional kernel learning and multi-stage sampling framework. Appl Soft Comput 104(106):909
Jin SS, Jung HJ (2016) Sequential surrogate modeling for efficient finite element model updating. Comput Struct 168:30–45. https://doi.org/10.1016/j.compstruc.2016.02.005
Jin Y, Wang H, Chugh T et al (2019) Data-driven evolutionary optimization: an overview and case studies. IEEE Trans Evol Comput 23(3):442–458. https://doi.org/10.1109/TEVC.2018.2869001
Johnson M, Moore L, Ylvisaker D (1990) Minimax and maximin distance designs. J Stat Plann Inference 26:131–148. https://doi.org/10.1016/0378-3758(90)90122-B
Jones D, Schonlau M, Welch W (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13:455–492. https://doi.org/10.1023/A:1008306431147
Jones DR, Perttunen CD, Stuckman B (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79:157–181. https://doi.org/10.1007/BF00941892
Joseph V, Hung Y, Sudjianto A (2008) Blind kriging: a new method for developing metamodels. J Mech Des 130(031):102. https://doi.org/10.1115/1.2829873
Kamiński B (2015) A method for the updating of stochastic kriging metamodels. Eur J Oper Res 247(3):859–866. https://doi.org/10.1016/j.ejor.2015.06.070
Karban P, Pánek D, Orosz T et al (2021) Fem based robust design optimization with agros and Ārtap. Comput Math Appl 81:618–633. https://doi.org/10.1016/j.camwa.2020.02.010, development and Application of Open-source Software for Problems with Numerical PDEs
Kaya M, Hajimirza S (2018) Surrogate based modeling and optimization of plasmonic thin film organic solar cells. Int J Heat Mass Transf 118:1128–1142. https://doi.org/10.1016/j.ijheatmasstransfer.2017.11.044
Kaymaz I (2005) Application of kriging method to structural reliability problems. Struct Saf 27(2):133–151. https://doi.org/10.1016/j.strusafe.2004.09.001
Kazikova A, Pluhacek M, Senkerik R (2020) Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison? MENDEL 26(2):9–16. https://doi.org/10.13164/mendel.2020.2.009
Kersaudy P, Sudret B, Varsier N et al (2015) A new surrogate modeling technique combining kriging and polynomial chaos expansions-application to uncertainty analysis in computational dosimetry. J Comput Phys 286:103–117
Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680. https://doi.org/10.1126/science.220.4598.671
Kleijnen JP (2017) Regression and kriging metamodels with their experimental designs in simulation: a review. Eur J Oper Res 256(1):1–16
Knowles J (2006) ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans Evol Comput 10(1):50–66. https://doi.org/10.1109/TEVC.2005.851274
Koza J (1992) Genetic programming—on the programming of computers by means of natural selection. MIT Press, Cambridge
Krige D (1951) A statistical approach to some basic mine valuation problems on the Witwatersrand. J S Afr Inst Min Metall 52(6):119–139
Ktari Z, Leitão C, Prates PA et al (2021) Mechanical design of ring tensile specimen via surrogate modelling for inverse material parameter identification. Mech Mater 153(103):673. https://doi.org/10.1016/j.mechmat.2020.103673
Kudela J (2019) Minimum-volume covering ellipsoids: Improving the efficiency of the Wolfe-Atwood algorithm for large-scale instances by pooling and batching. MENDEL 25(2):19–26. https://doi.org/10.13164/mendel.2019.2.019
Kudela J, Matousek R (2022) Lipschitz-based surrogate model for high-dimensional computationally expensive problems. arXiv preprint arXiv:2204.14236
Kudela J, Popela P (2020) Pool & discard algorithm for chance constrained optimization problems. IEEE Access 8:79397–79407. https://doi.org/10.1109/ACCESS.2020.2990726
Lai X, Wang X, Nie Y et al (2020) An efficient parameter estimation method for nonlinear high-order systems via surrogate modeling and cuckoo search. Soft Comput 24(22):17065–17079
Lal A, Datta B (2017) Modelling saltwater intrusion processes and development of a multi-objective strategy for management of coastal aquifers utilizing planned artificial freshwater recharge. Model Earth Syst Environ 4:111–126. https://doi.org/10.1007/s40808-017-0405-x
Lee S, Chen W (2008) A comparative study of uncertainty propagation methods for black-box-type problems. Struct Multidiscip Optim 37:239–253. https://doi.org/10.1007/s00158-008-0234-7
Leifsson L, Hermannsson E, Koziel S (2015) Optimal shape design of multi-element trawl-doors using local surrogate models. J Comput Sci 10:55–62. https://doi.org/10.1016/j.jocs.2015.01.006
Leser PE, Hochhalter JD, Warner JE et al (2017) Probabilistic fatigue damage prognosis using surrogate models trained via

three-dimensional finite element analysis. Struct Health Monit 16(3):291–308. https://doi.org/10.1177/1475921716643298
Li H, Liu T, Wang M et al (2017) Design optimization of stent and its dilatation balloon using kriging surrogate model. BioMedical Eng OnLine 16. https://doi.org/10.1186/s12938-016-0307-6
Li S, Trevelyan J, Wu Z et al (2019) An adaptive SVD–Krylov reduced order model for surrogate based structural shape optimization through isogeometric boundary element method. Comput Methods Appl Mech Eng 349:312–338. https://doi.org/10.1016/j.cma.2019.02.023
Liang L, Liu M, Martin C, et al (2018a) A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. J R Soc Interface 15(138):20170844. https://doi.org/10.1098/rsif.2017.0844
Liang L, Liu M, Martin C et al (2018b) A machine learning approach as a surrogate of finite element analysis-based inverse method to estimate the zero-pressure geometry of human thoracic aorta. Int J Numer Methods Biomed Eng 34(8):e3103. https://doi.org/10.1002/cnm.3103, e3103 CNM-Dec-17-0318, https://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/cnm.3103
Lim DK, Woo DK, Yeo HK et al (2015) A novel surrogate-assisted multi-objective optimization algorithm for an electromagnetic machine design. IEEE Trans Magn 51(3):1–4. https://doi.org/10.1109/TMAG.2014.2359452
Lin Y, Yang Q, Guan G (2019) Scantling optimization of FPSO internal turret area structure using RBF model and evolutionary strategy. Ocean Eng 191(106):562. https://doi.org/10.1016/j.oceaneng.2019.106562
Lirio RB, Camejo D, Loubes JM et al (2014) Estimation of covariance functions by a fully data-driven model selection procedure and its application to Kriging spatial interpolation of real rainfall data. Stat Methods Appl 23(2):149–174. https://doi.org/10.1007/s10260-013-0250-7
Liu Z, Zhu C, Zhu P et al (2018) Reliability-based design optimization of composite battery box based on modified particle swarm optimization algorithm. Compos Struct 204:239–255. https://doi.org/10.1016/j.compstruct.2018.07.053
Loshchilov I, Schoenauer M, Sebag M (2010) Dominance-based pareto-surrogate for multi-objective optimization. In: Simulated evolution and learning. SEAL 2010. Lecture notes in computer science, vol 6457. Springer, Berlin. https://doi.org/10.1007/978-3-642-17298-4_24
Matousek R, Dobrovsky L, Kudela J (2022) How to start a heuristic? Utilizing lower bounds for solving the quadratic assignment problem. Int J Ind Eng Comput 13(2):151–164
McKay M, Beckman R, Conover W (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21:239–245. https://doi.org/10.2307/1268522
Mendes MHS, Soares GL, Coulomb JL et al (2013) A surrogate genetic programming based model to facilitate robust multi-objective optimization: a case study in magnetostatics. IEEE Trans Magn 49(5):2065–2068. https://doi.org/10.1109/TMAG.2013.2238615
Meng D, Yang S, Zhang Y et al (2019) Structural reliability analysis and uncertainties-based collaborative design and optimization of turbine blades using surrogate model. Fatigue Fract Eng Mater Struct 42(6):1219–1227. https://doi.org/10.1111/ffe.12906
Morris M, Mitchell T (1995) Exploratory designs for computational experiments. J Stat Plan Inference 43:381–402. https://doi.org/10.1016/0378-3758(94)00035-T
Mueller J (2014) MATSuMoTo: the MATLAB surrogate model toolbox for computationally expensive black-box global optimization problems. arXiv:1404.4261
Mukhopadhyay T, Naskar S, Dey S et al (2016) On quantifying the effect of noise in surrogate based stochastic free vibration analysis of laminated composite shallow shells. Compos Struct 140:798–805. https://doi.org/10.1016/j.compstruct.2015.12.037
Müller J, Piché R (2011) Mixture surrogate models based on Dempster-Shafer theory for global optimization problems. J Glob Optim 51:79–104. https://doi.org/10.1007/s10898-010-9620-y
Nabian MA, Meidani H (2019) A deep learning solution approach for high-dimensional random differential equations. Probab Eng Mech 57:14–25. https://doi.org/10.1016/j.probengmech.2019.05.001
Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313. https://doi.org/10.1093/comjnl/7.4.308
Ninic J, Freitag S, Meschke G (2017) A hybrid finite element and surrogate modelling approach for simulation and monitoring supported TBM steering. Tunn Undergr Space Technol 63:12–28. https://doi.org/10.1016/j.tust.2016.12.004
Nobari A, Ouyang H, Bannister P (2015) Uncertainty quantification of squeal instability via surrogate modelling. Mech Syst Signal Process 60:887–908
Nocedal J, Wright S (2006) Numerical optimization. Springer Series in Operations Research and Financial Engineering. Springer, New York
Novak L, Novak D (2018) Polynomial chaos expansion for surrogate modelling: theory and software. Beton- und Stahlbetonbau 113(S2):27–32. https://doi.org/10.1002/best.201800048
Nyshadham C, Rupp M, Bekker B et al (2019) Machine-learned multi-system surrogate models for materials prediction. npj Comput Mater 5:51. https://doi.org/10.1038/s41524-019-0189-9
Omairey SL, Dunning PD, Sriramula S (2019) Multiscale surrogate-based framework for reliability analysis of unidirectional FRP composites. Compos B Eng 173(106):925. https://doi.org/10.1016/j.compositesb.2019.106925
Owen NE, Challenor P, Menon PP et al (2017) Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators. SIAM/ASA J Uncertain Quantif 5(1):403–435
Papadopoulos V, Soimiris G, Giovanis D et al (2018) A neural network-based surrogate model for carbon nanotubes with geometric nonlinearities. Comput Methods Appl Mech Eng 328:411–430. https://doi.org/10.1016/j.cma.2017.09.010
Park HJ, Yeo HK, Jung SY et al (2018) A robust multimodal optimization algorithm based on a sub-division surrogate model and an improved sampling method. IEEE Trans Magn 54(3):1–4. https://doi.org/10.1109/TMAG.2017.2755073
Paszke A, Gross S, Chintala S et al (2017) Automatic differentiation in pytorch. In: NIPS 2017 workshop autodiff decision program chairs
Pavlíček K, Kotlan V, Doležel I (2019) Applicability and comparison of surrogate techniques for modeling of selected heating problems. Comput Math Appl 78(9):2897–2910. https://doi.org/10.1016/j.camwa.2019.02.013, applications of Partial Differential Equations in Science and Engineering
Pedregosa F, Varoquaux G, Gramfort A et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12(85):2825–2830
Pfrommer J, Zimmerling C, Liu J et al (2018) Optimisation of manufacturing process parameters using deep neural networks as surrogate models. Procedia CIRP 72:426–431. https://doi.org/10.1016/j.procir.2018.03.046, 51st CIRP Conference on Manufacturing Systems
Poli R, Kennedy J, Blackwell T (2007) Particle swarm optimization. Swarm Intell 1:33–57. https://doi.org/10.1007/s11721-007-0002-0
Provost F, Jensen D, Oates T (1999) Efficient progressive sampling. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, pp 23–32. https://doi.org/10.1145/312129.312188
Putra N, Palar PS, Anzai H et al (2018) Multiobjective design optimization of stent geometry with wall deformation for triangular

and rectangular struts. Med Biol Eng Comput 57:15–26. https://doi.org/10.1007/s11517-018-1864-6
Qian J, Yi J, Cheng Y et al (2020) A sequential constraints updating approach for kriging surrogate model-assisted engineering optimization design problem. Eng Comput 36:993–1009. https://doi.org/10.1007/s00366-019-00745-w
Qin S, Zhang Y, Zhou YL et al (2018) Dynamic model updating for bridge structures using the kriging model and PSO algorithm ensemble with higher vibration modes. Sensors. https://doi.org/10.3390/s18061879
Qiu N, Gao Y, Fang J et al (2018) Crashworthiness optimization with uncertainty from surrogate model and numerical error. Thin-Walled Struct 129:457–472. https://doi.org/10.1016/j.tws.2018.05.002
Rafiee V, Faiz J (2019) Robust design of an outer rotor permanent magnet motor through six-sigma methodology using response surface surrogate model. IEEE Trans Magn 55(10):1–10. https://doi.org/10.1109/TMAG.2019.2923160
Rahman S, Xu H (2004) A univariate dimension-reduction method for multi-dimensional integration in stochastic mechanics. Probab Eng Mech 19(4):393–408. https://doi.org/10.1016/j.probengmech.2004.04.003
Rasmussen C, Williams C (2006) Gaussian processes for machine learning. MIT Press, US
Rios LM, Sahinidis N (2013) Derivative-free optimization: a review of algorithms and comparison of software implementations. J Glob Optim 56:1247–1293. https://doi.org/10.1007/s10898-012-9951-y
Rocas M, García-González A, Zlotnik S et al (2021) Nonintrusive uncertainty quantification for automotive crash problems with VPS/Pamcrash. Finite Elem Anal Des 193(103):556
Rocas M, García-González A, Larrayoz X et al (2022) Adaptive surrogates of crashworthiness models for multi-purpose engineering analyses accounting for uncertainty. Finite Elem Anal Des 203(103):694
Sacks J, Welch WJ, Mitchell TJ et al (1989) Design and analysis of computer experiments. Stat Sci 4(4):409–423
Sanchez F, Budinger M, Hazyuk I (2017) Dimensional analysis and surrogate models for the thermal modeling of multiphysics systems. Appl Therm Eng 110:758–771. https://doi.org/10.1016/j.applthermaleng.2016.08.117
Schmidt M, Lipson H (2009) Distilling free-form natural laws from experimental data. Science 324(5923):81–85. https://doi.org/10.1126/science.1165893
Schulz M, Dittmann J, Böl M (2019) Modeling the mechanical behavior of semi-flexible polymer chains using a surrogate model based on a finite-element approach to Brownian polymer dynamics. J Mech Phys Solids 130:101–117. https://doi.org/10.1016/j.jmps.2019.05.016
Shi H, Ma T, Chu W et al (2017) Optimization of inlet part of a microchannel ceramic heat exchanger using surrogate model coupled with genetic algorithm. Energy Convers Manag 149:988–996. https://doi.org/10.1016/j.enconman.2017.04.035
Shi J, Chu L, Braun R (2019) A kriging surrogate model for uncertainty analysis of graphene based on a finite element method. Int J Mol Sci. https://doi.org/10.3390/ijms20092355
Shi M, Li H, Liu X (2017) Multidisciplinary design optimization of dental implant based on finite element method and surrogate models. J Mech Sci Technol 31:5067–5073. https://doi.org/10.1007/s12206-017-0955-x
Shi R, Liu L, Long T et al (2017) Surrogate assisted multidisciplinary design optimization for an all-electric geo satellite. Acta Astronaut 138:301–317. https://doi.org/10.1016/j.actaastro.2017.05.032, the Fifth International Conference on Tethers in Space
Silber S, Koppelstätter W, Weidenholzer G et al (2018) Reducing development time of electric machines with symspace. In: 2018 8th international electric drives production conference (EDPC), pp 1–5. https://doi.org/10.1109/EDPC.2018.8658312
Slot RM, Sørensen JD, Sudret B et al (2020) Surrogate model uncertainty in wind turbine reliability assessment. Renew Energy 151:1150–1162
Sobester A (2003) Enhancements to global design optimization techniques. PhD thesis, University of Southampton
Song X, Lv L, Li J et al (2018) An advanced and robust ensemble surrogate model: extended adaptive hybrid functions. J Mech Des 140(041):402. https://doi.org/10.1115/1.4039128
Steuben J, Turner C (2014) Adaptive surrogate-model fitting using error monotonicity
Stork J, Friese M, Zaefferer M et al (2020) Open issues in surrogate-assisted optimization. Springer, Cham, pp 225–244. https://doi.org/10.1007/978-3-030-18764-4_10
Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359. https://doi.org/10.1023/A:1008202821328
Su G, Peng L, Hu L (2017) A gaussian process-based dynamic surrogate model for complex engineering structural reliability analysis. Struct Saf 68:97–109. https://doi.org/10.1016/j.strusafe.2017.06.003
Sun G, Wang S (2019) A review of the artificial neural network surrogate modeling in aerodynamic design. Proc Inst Mech Eng Part G J Aerosp Eng 233(16):5863–5872. https://doi.org/10.1177/0954410019864485
Tan F, Wang L, Yin M et al (2019) Obtaining more accurate convective heat transfer coefficients in thermal analysis of spindle using surrogate assisted differential evolution method. Appl Therm Eng 149:1335–1344. https://doi.org/10.1016/j.applthermaleng.2018.12.124
Tan Z, Song X, Cao W et al (2015) DFIG machine design for maximizing power output based on surrogate optimization algorithm. IEEE Trans Energy Convers 30(3):1154–1162. https://doi.org/10.1109/TEC.2015.2411153
Taran N, Ionel DM, Dorrell DG (2018) Two-level surrogate-assisted differential evolution multi-objective optimization of electric machines using 3-d FEA. IEEE Trans Magn 54(11):1–5. https://doi.org/10.1109/TMAG.2018.2856858
Tie Y, Hou Y, Li C et al (2020) Optimization for maximizing the impact-resistance of patch repaired CFRP laminates using a surrogate-based model. Int J Mech Sci 172(105):407. https://doi.org/10.1016/j.ijmecsci.2019.105407
Torkzadeh P, Fathnejat H, Ghiasi R (2016) Damage detection of plate-like structures using intelligent surrogate model. Smart Struct Syst 18:1233–1250. https://doi.org/10.12989/sss.2016.18.6.1233
Tripathy RK, Bilionis I (2018) Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. J Comput Phys 375:565–588
Vapnik V (1995) The nature of statistical learning theory. Springer, New York
Vega MA, Todd MD (2020) A variational Bayesian neural network for structural health monitoring and cost-informed decision-making in miter gates. Struct Health Monit. https://doi.org/10.1177/1475921720904543
Viana FA, Goel T (2010) Surrogates toolbox user's guide. In: Gainesville, FL, USA
Viana FA, Picheny V, Haftka R (2010) Using cross validation to design conservative surrogates. AIAA J 48:2286–2298. https://doi.org/10.2514/1.J050327
Wang B, Yan L, Duan X et al (2022) An integrated surrogate model constructing method: Annealing combinable gaussian process. Inf Sci
Wang N, Chang H, Zhang D (2021) Efficient uncertainty quantification for dynamic subsurface flow with surrogate by theory-guided neural network. Comput Methods Appl Mech Eng 373(113):492

Wang T, Shao M, Guo R et al (2021b) Surrogate model via artificial intelligence method for accelerating screening materials and performance prediction. Adv Funct Mater 31(8). https://doi.org/10.1002/adfm.202006245

Wang Y, Zhang Y, Zhao H et al (2018) Identifying interphase properties in polymer nanocomposites using adaptive optimization. Compos Sci Technol 162:146–155. https://doi.org/10.1016/j.compscitech.2018.04.017

Watts S, Arrighi W, Kudo J et al (2019) Simple, accurate surrogate models of the elastic response of three-dimensional open truss micro-architectures with applications to multiscale topology design. Struct Multidiscip Optim. https://doi.org/10.1007/s00158-019-02297-5

Wee H, Reid J, Chinchilli V et al (2016) Finite element-derived surrogate models of locked plate fracture fixation biomechanics. Ann Biomed Eng 45:668–680. https://doi.org/10.1007/s10439-016-1714-3

Westermann P, Evins R (2019) Surrogate modelling for sustainable building design—a review. Energy Build 198:170–186. https://doi.org/10.1016/j.enbuild.2019.05.057

White DA, Arrighi WJ, Kudo J et al (2019) Multiscale topology optimization using neural network surrogate models. Comput Methods Appl Mech Eng 346:1118–1135. https://doi.org/10.1016/j.cma.2018.09.007

Wiener N (1938) The homogeneous. Am J Math 60:897–936

Wu MC, Kamensky D, Wang C et al (2017) Optimizing fluid-structure interaction systems with immersogeometric analysis and surrogate modeling: application to a hydraulic arresting gear. Comput Methods Appl Mech Eng 316:668–693. https://doi.org/10.1016/j.cma.2016.09.032, special Issue on Isogeometric Analysis: Progress and Challenges

Xiu D, Karniadakis G (2002) The Wiener–Askey polynomial chaos for stochastic differential equations. SIAM J Sci Comput 24:619–644. https://doi.org/10.1137/S1064827501387826

Xu J, Han Z, Yan X et al (2019) Design optimization of a multi-megawatt wind turbine blade with the NPU-MWA airfoil family. Energies. https://doi.org/10.3390/en12173330

Yan C, Yin Z, Shen X et al (2020) Surrogate-based optimization with improved support vector regression for non-circular vent hole on aero-engine turbine disk. Aerosp Sci Technol 96(105):332. https://doi.org/10.1016/j.ast.2019.105332

Yang S, Tian W, Cubi E et al (2016) Comparison of sensitivity analysis methods in building energy assessment. Proc Eng 146:174–181. https://doi.org/10.1016/j.proeng.2016.06.369, the 8th international cold climate HVAC Conference

Ye D, Zun P, Krzhizhanovskaya V et al (2022) Uncertainty quantification of a three-dimensional in-stent restenosis model with surrogate modelling. J R Soc Interface 19(187):20210864

Yong H, Wang L, Toal D et al (2019) Multi-fidelity kriging-assisted structural optimization of whole engine models employing medial meshes. Struct Multidisc Optim 60:1209–1226. https://doi.org/10.1007/s00158-019-02242-6

Yoo K, Bacarreza O, Aliabadi MHF (2020) A novel multi-fidelity modelling-based framework for reliability-based design optimisation of composite structures. Eng Comput. https://doi.org/10.1007/s00366-020-01084-x

Zerpa L, Queipo N, Pintos SA et al (2005) An optimization methodology of alkaline-surfactant-polymer flooding processes using field scale numerical simulation and multiple surrogates. J Petrol Sci Eng 47:197–208. https://doi.org/10.1016/j.petrol.2005.03.002

Zhang J, Au F (2014) Calibration of initial cable forces in cable-stayed bridge based on kriging approach. Finite Elem Anal Des 92:80–92

Zhao Z, Dai K, Lalonde ER et al (2019) Studies on application of scissor-jack braced viscous damper system in wind turbines under seismic and wind loads. Eng Struct 196(109):294. https://doi.org/10.1016/j.engstruct.2019.109294

Zhou X, Zhang G, Hao X et al (2016) Enhanced differential evolution using local Lipschitz underestimate strategy for computationally expensive optimization problems. Appl Soft Comput 48(C):169–181. https://doi.org/10.1016/j.asoc.2016.06.044

Zhou Y, Lu Z (2020) An enhanced kriging surrogate modeling technique for high-dimensional problems. Mech Syst Signal Process 140(106):687. https://doi.org/10.1016/j.ymssp.2020.106687

A8

Information Sciences 619 (2023) 457–477

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Combining Lipschitz and RBF surrogate models for high-dimensional computationally expensive problems

Jakub Kůdela *, Radomil Matoušek

Institute of Automation and Computer Science, Brno University of Technology, Technicka 2, Brno, Czech Republic

ARTICLE INFO

Article history:
Received 13 July 2021
Received in revised form 11 October 2022
Accepted 12 November 2022
Available online 17 November 2022

Keywords:
Lipschitz surrogate model
Differential evolution
Radial basis function
Surrogate assisted evolutionary algorithms
High-dimensional expensive optimization

ABSTRACT

Standard evolutionary optimization algorithms assume that the evaluation of the objective and constraint functions is straightforward and computationally cheap. However, in many real-world optimization problems, these evaluations involve computationally expensive numerical simulations or physical experiments. Surrogate-assisted evolutionary algorithms (SAEAs) have recently gained increased attention for their performance in solving these types of problems. The main idea of SAEAs is the integration of an evolutionary algorithm with a selected surrogate model that approximates the computationally expensive function. In this paper, we propose a surrogate model based on a Lipschitz underestimation and use it to develop a differential evolution-based algorithm. The algorithm, called Lipschitz Surrogate-assisted Differential Evolution (LSADE), utilizes the Lipschitz-based surrogate model, along with a standard radial basis function surrogate model and a local search procedure. The experimental results on seven benchmark functions of dimensions 30, 50, 100, and 200 show that the proposed LSADE algorithm is competitive compared with the state-of-the-art algorithms under a limited computational budget, being especially effective for the very complicated benchmark functions in high dimensions.

1. Introduction

Many real-world optimization problems involve expensive computations, such as computational fluid dynamics and finite element analysis, or executions of physical experiments. In such situations, the evaluation of objective functions or constraints can take an excessively long time, prohibiting the use of conventional optimization methods [1]. To mitigate the computational costs, surrogate models (sometimes called metamodels [2]) have been widely used in combination with evolutionary algorithms (EAs), which are known as surrogate-assisted EAs (SAEAs) [3].

SAEAs execute only a limited number of real objective function (or constraint) evaluations and use these evaluations to train surrogate models. The surrogate models then serve as approximations of the real functions [4], and their evaluation should have negligible computational costs compared to evaluating the real functions. Many standard machine learning models, such as polynomial response surface [5], Kriging (or Gaussian processes) [6], artificial neural networks [7], radial basis functions (RBFs) [8] or support vector regression [9] have been employed in SAEAs. The performance of different surrogate models under multiple criteria was investigated in [10].

* Corresponding author.
E-mail address: Jakub.Kudela@vutbr.cz (J. Kůdela).

https://doi.org/10.1016/j.ins.2022.11.045
0020-0255/

EAs are effective metaheuristics used for global optimization, which are inspired by the processes of biological evolution, such as reproduction, mutation, and natural selection. The most widely known examples of these techniques are genetic algorithms (GA), differential evolution (DE), evolutionary strategy (ES), or particle swarm optimization (PSO). These methods were successfully used in the optimization of various complex problems such as the hyperparameter optimization in deep learning [11], difficult assignment problems [12], design of quantum operators [13], dynamical systems prediction [14], or solving boundary value problems [15].

Surrogate models are being employed in a variety of real-world problems, including protein structure prediction [16], elastic actuator design [17], structural optimization design of truss topology [18] or robust optimization of large scale networks [19]. A review of recent advances and applications of surrogate models for finite element method computations can be found in [20].

Based on the current surrogate model, the SAEAs typically choose two types of solutions for real function evaluation: promising samples around the optimum of the surrogate model, and uncertain samples with a large expected approximation error. For example, in [21] the authors designed multiple trial positions for each particle and then used an RBF model to select a position with the minimum predicted fitness value. A global and a local surrogate-assisted PSO algorithm for computationally expensive problems was developed in [22]. Here, the particle with a smaller predicted fitness value than its personal historical best was exactly evaluated. The uncertain samples were used to guide the search into some sparse and not yet well-explored areas, while the promising samples were used to guide a local search in the most promising areas. Many combinations of the two types are used to keep a good balance of global exploration and local exploitation. For instance, [23] developed a dimension reduction method to construct a Kriging surrogate model in a lower-dimensional space and chose the offspring with better lower confidence bound (LCB) values for real function evaluation. The LBC values were also used in [24], where the authors employed two different surrogate models. Here, the weight coefficient of the two models was changed to control the evolutionary progress. Another approach utilizing a trust region method for the interleaved use of exact models with computationally inexpensive RBF surrogates during a local search was developed in [25].

Surrogate models can guide the search of EAs to promising directions by using optima of these models, as was demonstrated in [21,26], and many others. It has also been shown that evaluating the uncertain samples can strengthen the exploration capabilities of SAEAs and effectively improve the approximation accuracy of the surrogate [2,4], and different methods for estimating the degree of uncertainty in function prediction have been proposed [27].

In recent years, there has been a multitude of SAEAs proposed in the literature. These algorithms usually employ a metaheuristic algorithm to be the primary optimization framework and use the surrogates as additional tools to accelerate the convergence of the underlying metaheuristic algorithm. In general, it is difficult for EAs to search for global optima in high-dimensional spaces because of the curse of dimensionality. SAEAs also encounter the same challenge when the dimension of a problem is high. Although current SAEAs can handle high-dimensional expensive problems relatively well, most of these algorithms still need many function evaluations (usually more than several thousands) to obtain good optimization results. Also, these algorithms are developed for optimizing problems whose dimensions are usually less than 30. For instance, the generalized surrogate single-objective memetic algorithm proposed in [28] needs 8000 function evaluations for 30D problems. The surrogate-assisted DE algorithm introduced [29] needs more than 10000 function evaluations for 30D problems. A similarly high number of required function evaluations were utilized by Lipschitz-based algorithm in [30]. A framework combining particle swarm optimization and RBF global surrogate was developed in [21], where the proposed method first generates multiple candidate solutions for each particle in each generation, and then the surrogate is employed to select the promising positions to form the new population. The Gaussian process model was utilized in [23] with the lower confidence bound to prescreen solutions in a differential evolution (DE) algorithm and a dimensional reduction technique was used to enhance the accuracy of the model. The maximum dimension of the test problems used in [23] was 50 and the dimension was reduced to 4 before the surrogate was constructed. An alternative approach for this issue is the use of multiple swarms, that can enhance population diversity, explore different search spaces simultaneously to efficiently find promising areas, and combine the advantage of different swarms if heterogeneous swarms are used. For computationally expensive problems, multiple swarms were used in the surrogate-assisted multiswarm optimization (SAMSO) algorithm [31]. The SAMSO algorithm takes advantage of the good global searchability of the teaching learning-based optimization algorithm and the fast convergence ability of the PSO algorithm.

Multiple surrogates have been shown to perform better than single ones in assisting EAs, typically utilizing a global surrogate model to smooth out the local optima, and local surrogate models to capture the local details of the fitness function around the neighborhood of the current best individuals. In [32] an ensemble surrogate-based model management method for surrogate-assisted PSO was proposed. This method searches for the promising and most uncertain candidate solutions to be evaluated using the expensive fitness function. Their results were outstanding on medium-scale test functions with a limited number of function evaluations. Surrogate-assisted cooperative swarm optimization (SA-COSO) for high-dimensional expensive problems, developed in [26], combined two PSO methods to solve problems with dimension up to 200. Another algorithm for high dimensional expensive problems, called evolutionary sampling assisted optimization (ESAO), which utilized a global RBF model and a local optimizer, was developed in [33].

A generalized surrogate-assisted evolutionary algorithm (GSGA) based on the optimization framework of the genetic algorithm was proposed in [34]. This algorithm uses a surrogate-based trust region local search method, a surrogate-guided GA updating mechanism with a neighbor region partition strategy, and a prescreening strategy based on the expected improvement infilling criterion of a simplified Kriging in the optimization process. A multi-objective infill criterion for a

Gaussian process assisted social learning particle swarm optimization (MGP-SLPSO) algorithm was proposed in [35]. The multi-objective infill criterion considers the approximated fitness and the approximation uncertainty as two objectives and uses non-dominated sorting for model management. Surrogate-assisted grey wolf optimization (SAGWO) algorithm was introduced in [36], where RBF is employed as the surrogate model. SAGWO conducts the search in three phases, initial exploration, RBF-assisted meta-heuristic exploration, and knowledge mining on RBF.

In this paper, we propose a novel Lipschitz-based surrogate model, that is designed to increase the exploration capabilities of SAEAs. We also develop a new Lipschitz surrogate-assisted differential evolution (LSADE) algorithm that uses the Lipschitz-based surrogate in combination with a standard RBF surrogate and a local optimization procedure. The rest of this paper is organized as follows. Section 2 briefly introduces the related techniques, including surrogate models, Lipschitz-based underestimation, and DE. Section 3 describes the proposed LSADE algorithm in detail. In Section 4, we provide a computational analysis of the individual components of the LSADE algorithm, the frequency of the utilization of said components, the choice of an RBF, and a comparison with other state-of-the-art SAEAs, namely with SA-COSO, ESAO, SAMSO, GSGA, MGP-SLPSO, and SAGWO. The conclusions and future research directions are described in Section 5.

2. Related Techniques

2.1. Surrogate Models

Kriging models and RBFs are the most widely applied methods for generating surrogate models [37]. It has been shown that the Kriging model outperforms other surrogate models in solving low-dimensional optimization problems, and RBF is the most efficient method among surrogates for solving high-dimensional optimization problems [38]. A disadvantage of Kriging is that the training of the model is time-consuming when the number of samples is large. Since this paper focuses on high-dimensional problems, we will adopt the RBF methodology for building the surrogate model, which has been successfully used in several other SAEAs [26].

RBFs compute a weighted sum of prespecified simple functions to approximate complex design landscape. Given t different sample points $X_1, \ldots, X_t$, the RBF surrogates are written as [20]

$$f_{\mathrm{RBF}}(x) = \sum_{i=1}^{t} w_i \psi(||x - X_i||_2),$$

where w_i denotes the weight which is computed using the method of least squares, and ψ is the chosen basis function. There are several (symmetric) radial functions that can serve as a basis function, such as Gaussian function, thin-plate splines, linear splines, cubic splines, and multiquadrics splines [20].

2.2. Lipschitz-based Underestimation

The use of a Lipschitz constant in optimization was first proposed in [39,40] and initiated a line of research within global optimization that is active to this day [41]. We assume that the unknown or expensive to compute objective function f has a finite Lipschitz constant k, i.e.

$$\exists k \geqslant 0 \text{ s.t. } |f(x) - f(x\prime)| \leqslant k||x - x\prime||_2 \ \forall (x, x\prime) \in \mathscr{X}^2,$$

which is among the weakest regularity assumptions we can ask for. Based on a sample of t evaluations of the function f at points $X_1, \ldots, X_t$, we can construct a global underestimator f_L of f by using the following expression [41]

$$f_L(x) = \max_{i=1,\ldots,t} f(X_i) - k||x - X_i||_2. \tag{1}$$

A visual representation of this Lipschitz-based surrogate function in 1D is depicted in Fig. 1, where each already evaluated point has two lines (one to the left and the other to the right) emanating from it under an angle that depends on the Lipschitz constant k. Then the surrogate is constructed as the pointwise maximum of the individual lines. A 2D visualization is shown in Fig. 2. This surrogate has two important properties – it assigns low values to points that are far from previously evaluated points and combines it with the information (objective value and "global" Lipschitz constant) from the closest evaluated point. Therefore, it can serve as a good "uncertainty measure" of prospective points for evaluation, as points with low values of f_L are either far from any other evaluated solution, or relatively close to a good one.

Naturally, since we do not know the objective function f itself, we can hardly expect to know the Lipschitz constant k. We will approach this issue by estimating k from the previously evaluated points. We will use the approach described in [41], which utilizes a nondecreasing sequence of Lipschitz constants $k_{i\in\mathbf{Z}}$ that defines a meshgrid on $\mathbf{R}^+$. The estimate $\hat{k}_t$ of the Lipschitz constant is then computed as

$$\hat{k}_t = \inf\left\{k_{i\in\mathbf{Z}} : \max_{l\neq j} \frac{|f(X_j) - f(X_l)|}{||X_j - X_l||_2} \leqslant k_i\right\}. \tag{2}$$

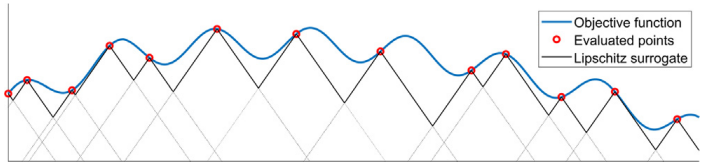


Fig. 1. Visual representation of the Lipschitz-based surrogate in 1D.

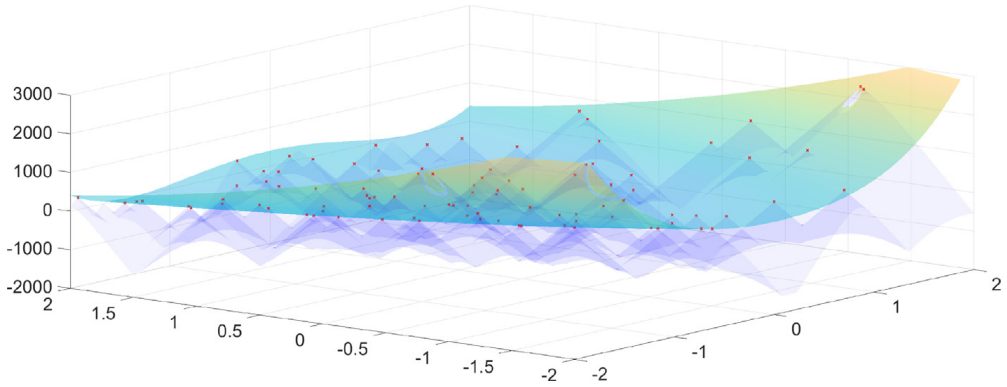


Fig. 2. Visual representation of the Lipschitz-based surrogate on the Rosenbrock function in 2D. Sampled points are highlighted in red and the Lipschitz-based surrogate in light blue.

Sequences of different shapes could be considered – we utilize a sequence $k_i = (1+\alpha)^i$ that uses a parameter $\alpha > 0$. For this sequence, the computation (2) of the estimate is simplifies into $\hat{k}_t = (1+\alpha)^{i_t}$, where

$$i_t = \left\lceil \ln\left(\max_{l \neq j} \frac{|f(X_j) - f(X_l)|}{||X_j - X_l||_2} \right) / \ln(1+\alpha) \right\rceil. \tag{3}$$

2.3. Differential Evolution

EAs are powerful methods for solving complex engineering optimization problems, that are difficult to approach with standard optimization methods. In this work, DE is employed as the optimization solver due to its straightforward structure and its global optimization capabilities. Several variants of DE have been developed to improve its performance [42]. In general, there are four stages of DE: initialization, mutation, crossover, and selection. We assume we have a population at the current generation, $x = [x_1, \ldots, x_t]$, where each individual has dimension $D, x_i = (x_i^1, \ldots, x_i^D)$. In this work, we utilize the DE/best/1 strategy for the mutation process of DE which, can be expressed as

$$v_i = x_b + F \cdot (x_{i_1} - x_{i_2}), \tag{4}$$

where x_b is the current best solution, x_{i_1} and x_{i_2} are different randomly selected individuals from the population, and F is a scalar number typically within the interval [0.4, 1] [42]. The crossover stage of DE is conducted after mutation and has the following form:

$$u_i^j = \begin{cases} v_i^j, & \text{if } (U_j(0,1) \leqslant C_r \,|\, j = j_{rand}), \\ x_i^j, & \text{otherwise}, \end{cases} \tag{5}$$

where u_i^j the jth component of ith offspring, x_j^i and v_j^i are the jth component of ith parent individual and the mutated individual, respectively. The crossover constant C_r is between 0 and 1, $U_j(0,1)$ indicates a uniformly distributed random number, and $j_{rand} \in [1, \ldots, D]$ is a randomly chosen index that ensures u_i has at least one component of v_i. The interested reader can find more information about the intricacies of DE in [42].

3. Proposed LSADE Method

The proposed LSADE method has four distinct parts: 1) the DE-based generation of prospective points, 2) the global RBF evaluation of the prospective points, 3) the Lipschitz surrogate evaluation of the prospective points, and 4) the local optimization within a close range of the best solution found so far. The execution of parts 2) – 4) of the algorithm can be con-

trolled based on chosen conditions, i.e., we may sometimes skip RBF surrogate evaluation, Lipschitz surrogate evaluation, or local optimization, if deemed advantageous.

At the beginning of the process, Latin hypercube sampling [37] is used to generate the initial population of t individuals, whose objective function is evaluated [43]. The best individual is found, a parent population of size p is randomly selected from the evaluated points and a new population is constructed based on the DE rules (4) and (5). If the *RBF evaluation condition* is true, the new population is evaluated based on the RBF surrogate model. Then the best individual based on the RBF model has its objective function evaluated and is added to the whole population. This step constitutes a global search strategy.

Algorithm 1: Pseudocode of the LSADE.

1: Generate an initial population of t points $X_1, \ldots, X_t$ and evaluate their objective function values. Denode the best solution as X_b.
2: Set $iter = 0$ (iteration counter), $NFE = t$ (number of function evaluations).
3: Use the evaluated points so far to estimate k by (3) and to construct the RBF surrogate.
4: Sample p points from the population as parents for DE.
5: Based on the DE rules (4) and (5), generate children.
6: Increase *iter* by 1.
7: **if** *RBF condition* **then**
8: Evaluate the children on the RBF surrogate.
9: Find the child with the minimum RFB surrogate value, and add it to the population and evaluate its objective function value. Increase *NFE* by 1.
10: **if** *Lipschitz condition* **then**
11: Evaluate the children on the Lipschitz surrogate (1).
12: Find the child with the minimum Lipschitz surrogate value, and add it to the population and evaluate its objective function value. Increase *NFE* by 1.
13: **if***Local Optimization condition* **then**
14: Construct a RBF local surrogate model using the best c solutions found so far.
15: Find the bounds in each dimension for the local optimization (6).
16: Minimize the local RBF surrogate model within the bounds. Denote the minimum as $\widehat{X}_m$ and, if it is not already in the population, add it to the population and evaluate its objective function value. Increase *NFE* by 1.
17: Find the best solution so far and denote it as X_b.
18: **if** $NFE < NFE_{max}$ **then**
19: **goto** 3.
20: **else**
21: **terminate**.

If the *Lipschitz evaluation condition* is true, the Lipschitz constant k is estimated based on (3) and the new population is evaluated on the Lipschitz surrogate model (1). The best individual based on the Lipschitz surrogate model has its objective function evaluated and is added to the whole population.

If the *Local optimization condition* is true, we construct a local RBF surrogate model using the best c solutions found so far, which we denote by $\widehat{X}_1, \ldots, \widehat{X}_c$. Additionally, we find the bounds for the local optimization procedure within those c points:

$$\begin{aligned} lb(i) &= \min_{j=1,\ldots,c} \widehat{X}_j(i), \quad i = 1, \ldots, D, \\ ub(i) &= \max_{j=1,\ldots,c} \widehat{X}_j(i), \quad i = 1, \ldots, D, \end{aligned} \tag{6}$$

and perform a local optimization of the local RBF model within the bounds $[lb, ub]$. For local optimization we adapt a sequential quadratic programming strategy, which was also used by the winner of the 2020 CEC Single Objective Bound Constrained Competition [44]. We find the local optimum and check, if it is not already in the population, before evaluating it and adding it to the population.

The evaluation of points based on the Lipschitz-based surrogate model can be thought of as an exploration step in the algorithm (and should increase our ability to find the regions of good solutions), whereas the evaluation of points based on the local optimization procedure can be thought of as an exploitation step of the algorithm (and should give us the means to improve the best solutions we have found so far).

The cycle of generating new population, evaluating it on the RBF and Lipschitz surrogate models and conducting the local optimization is carried out until a maximum number of objective function evaluations is reached. The pseudocode[1] for the LSADE method is described in Algorithm 1.

[1] The MATLAB code can be found at the authors github: https://github.com/JakubKudela89/LSADE

4. Results and Discussion

To examine the effectiveness of the proposed method, we compare it with six other state-of-the-art algorithms on a testbed of standard benchmark functions [45] that are summarized in Table 1. Although there are more recent benchmark sets, such as [46], these were not yet used for benchmarking SAEAs. The dimensions for the comparison are $D = 30, 50, 100, 200$ for all of the benchmark functions. We also investigate the advantages of the individual components of the LSADE method, the choice of the conditions for using the different components, and the choice of basis functions for the RBF surrogates. The algorithm is implemented in MATLAB R2020b and runs on an Intel(R) Core(TM) i5-4460 CPU @ 3.20 GHz desktop PC.

4.1. Experiment Setting

For constructing both the local and the global RBF surrogate models we used the SURROGATES toolbox [47] with default settings (multiquadric RBF with parameter $c = 1$). The DE coefficients were set to $F = 0.5$ and $C_r = 0.5$ [48]. The number of initial points were set to 100 for $D = [30, 50]$ and 200 for $D = [100, 200]$. The number of children was set to D. The local optimization uses the best $c = 3 \cdot D$ points found so far (or less if there are not enough points yet evaluated), and utilizes the sequential quadratic programming algorithm implemented in the FMINCON function with default parameters. The Lipschitz approximation parameter was set to $\alpha = 0.01$. The maximum number of function evaluations was set to 1000 for all problems. For all benchmark functions, 20 independent runs are conducted to get statistical results. Finally, some of the more in-depth results regarding the sensitivity of the parameters of the LSADE algorithm are studied in the Appendix.

4.2. Comparison of Individual Components

Firstly, we assess the effectiveness of the individual components of the LSADE: the RBF surrogate, the Lipschitz surrogate, the local optimization procedure, and their combinations. This corresponds to setting the *RBF condition, Lipschitz condition*, and *Local Optimization condition* to true or false (1 or 0) for every iteration of the algorithm. We denote the 8 possible variations as a triplet (R – RBF, Li – Lipschitz, Lo – Local Optimization) R#| Li#| Lo#, where the "#" indicates if the condition was true or false. The R0| Li0| Lo0 variation does not use any optimization (as there is no rule to add points for evaluation) and instead just evaluates 1000 randomly selected points, using the entire computational budget. The results (mean of the best-found objective function values over the 20 runs) for the different variations in dimensions $D = [30, 50]$ are reported in Table 2. Not surprisingly, the R0| Li0| Lo0 variant comes out being substantially worse than the other ones and is the only one that has its cells in the table colored in grey. The remaining variations are color-coded in the following way: the variant with the best (lowest) mean objective function value for a given problem instance has the corresponding cell in the table colored in a dark shade of green, the one with the worst (highest) mean objective function value has a dark red color, and the ones in between are ordered from green (better) to red (worse). This paradigm is also used in the subsequent tables for making straightforward comparisons. From Table 2 we can see that the "usefulness" of the individual components of LSADE is very problem-dependent, as there are instances, where adding either component may be beneficial or detrimental. However, based on the results, it seems advantageous to have the *RFB condition* be true, as the majority of the best results (11 of the 14 instances) were achieved by the R1 variants. As for the other two components, the situation is more nuanced – it is clear that they are both beneficial (the best results are always in a variant with either Li1 or Lo1), but the trade-off between adding one or the other needs to be be investigated in more detail.

4.3. Tuning the Lipschitz and Local Optimization Conditions

As LSADE allows controlling the addition of points for evaluation for the individual surrogates, we use it for tuning the balance between the exploration via the *Lipschitz condition* and the exploitation via the *Local Optimization condition* (from this point onward, the *RBF condition* is always true). We start by using static rules for both conditions to be true, which will be based on the current iteration number. We consider 5 possibilities: 1 – iteration number divisible by 1 (i.e., every iteration); 2 – iteration number divisible by 2 (every other iteration); 4 – iteration number divisible by 4; 8 – iteration number

Table 1
Benchmark functions used for the comparison

Problem	Description	Property	Optimum
F1	Ellipsoid	Unimodal	0
F2	Rosenbrock	Multimodal with narrow valley	0
F3	Ackley	Multimodal	0
F4	Griewank	Multimodal	0
F5	F10 in [45]	Very complicated multimodal	−330
F6	F16 in [45]	Very complicated multimodal	120
F7	F19 in [45]	Very complicated multimodal	10

Table 2
Comparison of the individual components of LSADE on $D = [30, 50]$.

D	F	R0 | Li0 | Lo0	R0 | Li0 | Lo1	R0 | Li1 | Lo0	R0 | Li1 | Lo1	R1 | Li0 | Lo0	R1 | Li0 | Lo1	R1 | Li1 | Lo0	R1 | Li1 | Lo1
30	F1	1898	124.3	222.1	7.787	3.660	0.0041	7.237	0.010
	F2	4641	193.2	379.9	60.79	38.55	30.32	46.32	29.79
	F3	20.35	16.94	13.55	12.96	12.63	16.99	5.399	13.37
	F4	467.0	109.1	53.63	9.595	1.234	10.69	2.030	0.431
	F5	434.7	-126.9	33.95	-114.6	-133.2	-153.7	-133.4	-216.9
	F6	1154	814.7	587.6	488.4	608.8	603.3	490.8	440.7
	F7	1348	1194	987.9	959.2	1062	1086	976.7	973.0
50	F1	6365	148.5	1131	6.645	285.5	3.727	69.96	2.352
	F2	10279	283.5	1070	79.83	214.4	65.41	161.8	65.12
	F3	20.59	17.45	15.48	13.38	18.36	17.98	10.58	15.56
	F4	926.6	307.0	162.5	21.87	79.97	191.9	9.117	6.463
	F5	1185	30.64	396.1	-122.9	272.9	20.02	161.9	-138.0
	F6	1276	880.6	679.3	368.9	787.4	752.7	567.8	410.4
	F7	1460	1296	1086	1019	1229	1238	1047	1077

Table 3
Comparison of the static rules for the Lipschitz and Local Optimization conditions, $D = [30, 50]$.

F1 $[0.0036, 0.3671]$

Li | Lo	1	2	4	8	0
1	0.0102	0.0342	0.1328	0.3671	7.2373
2	0.0063	0.0085	0.0223	0.041	1.5932
4	0.0041	0.0048	0.0087	0.0129	0.6838
8	0.0047	0.0036	0.0062	0.0107	0.2917
0	0.0041	0.0046	0.0179	0.0063	3.6604

F2 $[25.90, 32.59]$

Li | Lo	1	2	4	8	0
1	29.79	29.82	27.85	27.78	46.32
2	29.05	32.59	29.61	28.69	44.16
4	30.93	29.42	29.02	26.79	33.70
8	30.21	25.90	31.71	26.38	36.15
0	30.32	27.80	28.55	31.25	38.55

F3 $[1.67, 15.78]$

Li | Lo	1	2	4	8	0
1	13.37	7.85	2.30	1.67	5.39
2	13.94	10.95	6.28	3.48	4.74
4	15.22	13.79	9.24	6.42	7.40
8	15.78	14.62	9.80	10.19	9.69
0	16.99	16.17	14.99	13.39	12.63

F4 $[0.0035, 1.107]$

Li | Lo	1	2	4	8	0
1	0.431	0.290	0.508	0.595	2.030
2	0.586	0.144	0.109	0.197	1.253
4	0.702	0.120	0.075	0.078	1.082
8	1.107	0.160	0.035	0.061	1.025
0	10.69	2.193	0.615	0.139	1.234

F5 $[-222.1, -168.5]$

Li | Lo	1	2	4	8	0
1	-216.9	-222.1	-212.1	-217.3	-133.4
2	-192.1	-206.8	-191.6	-168.5	-137.5
4	-189.3	-182.8	-185.3	-177.0	-140.4
8	-173.0	-178.5	-184.1	-177.0	-140.0
0	-153.7	-157.6	-167.8	-167.0	-133.2

F6 $[418.7, 558.4]$

Li | Lo	1	2	4	8	0
1	440.7	423.8	438.0	437.4	490.8
2	476.1	440.5	462.8	418.7	493.7
4	492.1	466.4	471.9	463.9	511.6
8	558.4	529.0	509.1	495.0	543.8
0	603.3	592.8	597.7	587.1	608.9

F7 $[958.1, 1036.4]$

Li | Lo	1	2	4	8	0
1	973.1	986.9	968.9	960.1	976.7
2	971.4	968.5	974.3	958.1	972.8
4	1010	998.7	994.3	981.9	1005
8	1036.4	1013	987.1	1014	1029
0	1086.8	1070	1040	1033	1062

Min and max mean values from the static rules for $D = 50$ (disregarding rules with Li0 or Lo0)

	F1	F2	F3	F4	F5	F6	F7
min	0.445	47.45	6.459	1.010	-138.0	363.2	1019
max	6.003	65.37	16.78	71.78	0.750	615.8	1195

divisible by 8; 0 – never. For example, Li2 | Lo0 means that points for real function evaluation based on the *Lipschitz condition* are added every two iterations and the *Local Optimization* is not used at all. In this setting, there were 25 variations in total. The results of the computations (mean over the 20 independent runs) for all 25 variations of the considered static rules for $D = 30$ are reported in Table 3. In the table, next to the benchmark function identifier is the best and worst results in square brackets, where we chose to omit the rules with Li0 or Lo0 (as these were often quite a lot worse than the other ones). Also in Table 3 are the aggregate results for $D = 50$, while the detailed results can be found in the Appendix. These results suggest that using both the Lipschitz surrogate and the local optimization procedure is beneficial for every benchmark problem. The Lispchitz surrogate is especially well suited for problems F3 and F5-F7 (which are the ones with the complicated multimodal structure). However, none of the variations performed very well for all the considered problems, and the difference between the best and the worst variation for a given problem (even with disregarding rules with Li0 or Lo0) was quite high.

Since the Lipschitz surrogate should serve as an exploration-enhancing part of the algorithm, it is only natural that the frequency of its use should diminish as the iterations progress, to make space for the parts of the algorithm that focus on the exploitation of prospective areas. Hence, we devised several dynamic rules that decrease the frequency of using the Lipschitz surrogate, and increase the frequency of the local optimization, both in a linear manner. For instance, the variant Li1–4 | Lo8–1 starts with the Lipschitz surrogate being used every iteration and the local optimization procedure being used every 8 iterations, and ends with the Lipschitz surrogate being used every 4 iterations and the location optimization procedure being used every iteration. The individual conditions for the 9 considered variations can be found in the Appendix. The results of the computations with the dynamic rules for $D = [30, 50, 100]$ are summarized in Table 4. When comparing the results from the dynamic and the static rules, two important observations can be made. First, the dynamic rules have a much smaller interval between the best and the worst variation for the given problem instance, while the values of the best instances remain comparable. Second, there is one variation that stands out as having good results across many problem instances, particularly in higher dimensions.

The Li1–4 | Lo8–1 variant of the algorithm was selected as the best-performing one and will be used as the default variation for the subsequent modifications. It would probably be advantageous to devise a scheme that automatically decides on the frequency of using the Lipschitz surrogate or the local optimization procedure based on the past improvements and to tailor it for each problem separately. This is a research topic we plan to investigate in the future.

Table 4
Comparison of the dynamic rules for the Lipschitz and Local Optimization conditions, $D = [30, 50, 100]$.

D	F	[min,max]	1-4 | 8-1	1-6 | 8-1	1-8 | 8-1	1-4 | 6-1	1-6 | 6-1	1-8 | 6-1	1-4 | 4-1	1-6 | 4-1	1-8 | 4-1
30	F1	[0.0061, 0.0113]	0.0113	0.0087	0.0069	0.0103	0.0107	0.0079	0.0082	0.0066	0.0061
	F2	[26.63, 27.06]	27.06	26.69	26.83	27.04	27.01	26.63	26.95	26.86	26.75
	F3	[1.122, 3.496]	1.308	1.152	1.480	1.235	1.279	1.122	2.546	3.496	3.327
	F4	[0.013, 0.051]	0.051	0.033	0.012	0.037	0.040	0.019	0.040	0.013	0.014
	F5	[-218.7, -196.3]	-218.7	-213.4	-214.5	-196.3	-197.0	-198.5	-213.1	-211.3	-215.5
	F6	[402.8, 439.6]	433.7	439.6	436.5	402.8	406.3	412.2	425.1	434.6	434.6
	F7	[964.8, 978.9]	965.7	967.5	975.0	964.8	969.0	970.6	978.9	978.3	974.8
50	F1	[0.839, 1.686]	1.358	1.686	1.126	1.400	1.339	0.839	1.499	1.248	1.144
	F2	[47.65, 58.89]	47.65	47.73	49.71	50.12	50.25	51.67	58.15	58.69	57.93
	F3	[6.876, 12.42]	6.876	7.469	8.467	8.995	8.858	9.344	11.02	12.03	12.42
	F4	[0.749, 1.097]	0.819	0.789	0.749	0.887	0.898	0.879	1.031	1.045	1.097
	F5	[-136.4, -97.26]	-98.78	-97.26	-99.96	-108.7	-107.5	-123.9	-136.4	-132.3	-131.1
	F6	[367.2, 405.2]	370.3	379.2	405.2	384.8	375.1	388.8	367.2	384.1	380.7
	F7	[1015, 1068]	1016	1027	1053	1015	1025	1033	1037	1051	1068
100	F1	[88.13, 125.3]	112.8	105.2	94.59	97.99	106.3	88.13	125.3	110.3	112.0
	F2	[123.6, 147.5]	140.6	135.7	132.8	138.0	141.1	123.6	147.5	129.0	138.4
	F3	[12.05, 15.11]	12.05	12.77	13.41	13.25	13.48	14.06	14.78	14.90	15.11
	F4	[6.517, 18.74]	6.517	7.434	12.47	7.574	8.522	11.37	10.64	14.74	18.74
	F5	[34.52, 117.6]	60.28	117.6	82.19	92.60	96.94	94.94	34.52	44.79	82.18
	F6	[332.7, 363.0]	332.7	343.6	360.0	343.4	345.0	354.1	333.7	351.5	363.0
	F7	[1144, 1193]	1144	1162	1185	1162	1176	1193	1160	1184	1192

4.4. Comparison of Different RBFs

Next, we investigate the effect of using different basis functions for the two RBF surrogate models (one global and one local). We use the Li1–4 | Lo8–1 rule for the *Lipschitz* and *Local Optimization conditions* that was tuned for the multiquadratic (MQ) basis function and run the algorithm with cubic, thin plate spline (TPS), linear, and Gaussian basis function for the two RBFs instead. The results of the computations can be found in Table 5. From these results, it is apparent that the choice of the basis function has a substantial effect on the performance of the algorithm. Both the multiquadratic and the cubic basis functions performed very well on most of the problem instances, the TPS function was consistently mediocre, the Gaussian function performer mostly poorly (apart from the F1 problem) and the linear function performed the worst. The convergence histories of these variations can be found in the Appendix. Once again, it would very likely be beneficial to devise a scheme that would automatically choose the "appropriate" basis function for each problem separately. In the same vein, using different RBFs for the local and global models could also improve the performance of the algorithm.

4.5. Comparison with Other Algorithms

The proposed LSADE method is compared with six SAEAs, namely, SA-COSO [26], ESAO [33], SAGWO [36], GSGA [34], MGP-SLPSO [35], and SAMSO [31], which are all methods for high-dimensional expensive problems that can be compared on the same testbed (although some of the problems have not been evaluated by some of the algorithms). SA-COSO is a surrogate-assisted cooperative swarm optimization algorithm, in which a surrogate-assisted particle swarm optimization algorithm and a surrogate-assisted social learning based particle swarm optimization algorithm cooperatively search for the global optimum. ESAO is an evolutionary sampling-assisted optimization method that combines global and local search to balance exploration and exploitation, and employs DE as the optimization method. SAGWO utilizes the grey wolf optimization algorithm and conducts the search in three phases, initial exploration, RBF-assisted meta-heuristic exploration, and knowledge mining on RBF. GSGA uses a surrogate-based trust region local search method, a surrogate-guided GA updating mechanism with a neighbor region partition strategy, and a prescreening strategy based on the expected improvement infilling criterion of a simplified Kriging in the optimization process. MGP-SLPSO employs a multi-objective infill criterion

Table 5
Comparison of different basis functions, $D = [30, 50, 100]$.

D	F	MQ	Cubic	TPS	Linear	Gaussian
30	F1	0.011	0.011	0.509	6.276	0.003
	F2	27.06	27.77	31.86	93.31	28.10
	F3	1.308	0.256	0.418	4.946	4.164
	F4	0.051	0.176	0.577	1.883	0.944
	F5	-218.7	-172.6	-155.9	-143.6	-9.30
	F6	433.7	426.2	437.2	448.1	526.5
	F7	965.7	938.8	944.4	965.8	951.7
50	F1	1.358	0.434	7.556	54.78	0.191
	F2	47.65	47.98	62.20	221.8	47.71
	F3	6.876	0.695	1.822	10.56	5.161
	F4	0.819	0.380	0.801	5.668	0.930
	F5	-98.78	-10.03	2.45	82.64	274.9
	F6	370.3	481.6	464.5	521.6	585.6
	F7	1016	976.3	979.6	1054	985.7
100	F1	112.8	30.94	279.5	766.6	20.39
	F2	140.6	106.4	331.7	714.5	165.1
	F3	12.05	4.622	9.089	16.65	8.965
	F4	6.517	0.816	2.190	69.61	0.946
	F5	60.28	646.8	527.1	701.2	1012
	F6	332.7	550.4	522.9	572.5	596.3
	F7	1144	1056	1146	1248	1112

that considers the approximated fitness and the approximation uncertainty as two objectives for a Gaussian process assisted social learning particle swarm optimization algorithm. SAMSO is a a surrogate-assisted multiswarm optimization algorithm for high-dimensional problems, which includes two swarms: the first one uses the learner phase of teaching–learning-based optimization to enhance exploration and the second one uses the particle swarm optimization for faster convergence. The data for the comparison were obtained from the corresponding papers, with the exception of the data for SA-COSO and ESAO, which were obtained from [31].

The average objective function value for the considered algorithms and for the LSADE algorithm with multiquadratic and cubic RBFs are reported in Table 6. More detailed results, including the best results, worst results, and standard deviations of the independent runs for all the considered algorithms can be found in the Appendix. Looking at $D = 30$ first, we can see that there is no one algorithm that is strictly better than all the others on all the benchmark functions. The less complicated functions F1-F4 are dominated by MGP-SLPSO, GSGA, SAGWO, and EASO, while for the more complicated functions F5-F7 SAMSO seems to be the best. Both of the LSADE variants come out somewhere in the middle for all problems. In a direct comparison with LSADE, the best ones are SAMSO (better in 5/7 than LSADE-MQ) and GSGA (better in 5/7 than LSADE-C). For $D = 50$ the situation is quite similar: the best algorithms for the less complicated problems are MGP-SLPSO, SAGWO, and ESAO, while SAMSO dominates the more complicated problems again. Both of the LSADE variants are, once again, somewhere in the middle. In a direct comparison with LSADE, the SAMSO is the best (better in 6/7 than LSADE-MQ). However, the situation

Table 6
Comparison with other algorithms, average objective function value.

D	F	SAMSO	MGP-SLPSO	GSGA	SAGWO	ESAO	SA-COSO	LSADE-MQ	LSADE-C
30	F1	0.0053	0	0.073	0.00007	0.027	3.85	0.0113	0.0115
	F2	28.3	100	27.60	28.30	25.04	59.9	27.06	27.77
	F3	0.628	6.58	0.023	0	2.521	5.01	1.308	0.256
	F4	0.538	0.013	0.228	0.015	0.953	1.44	0.051	0.176
	F5	-239	-220	-203.0	-128.8	6.325	-57.4	-218.7	-172.6
	F6	372	N/A	424.7	489.8	N/A	528	433.7	426.2
	F7	922	952	927.2	973.2	931.6	969	965.7	938.8
50	F1	0.513	0	0.621	0.004	0.740	46.6	1.358	0.434
	F2	50.1	120	48.21	49.06	47.39	253	47.65	47.98
	F3	1.53	9.31	0.022	0	1.431	8.86	6.876	0.695
	F4	0.666	0.154	0.346	0.025	0.94	5.63	0.819	0.380
	F5	-169	33	-75.82	98.39	198.6	235	-98.78	-10.03
	F6	326	N/A	403.3	502.0	N/A	613	370.3	481.6
	F7	970	1060	970.7	1044.1	975.3	1080	1016	976.3
100	F1	72.1	0.00005	12.33	0.139	1283	985	112.8	30.94
	F2	286	612	109.1	123.4	578.8	2500	140.6	106.4
	F3	6.12	14.3	1.31	0	10.36	15.9	12.05	4.622
	F4	1.06	0.715	0.706	0.023	57.34	63.5	6.517	0.816
	F5	737	885	672.5	800.1	713.4	1420	60.28	646.8
	F6	513	N/A	447.2	518.6	N/A	807	332.7	550.4
	F7	1290	1390	1256	1350	1372	1410	1144	1056
200	F1	1520	N/A	N/A	N/A	17616	16382	3959	793.6
	F2	1150	N/A	N/A	N/A	4318	16411	927.2	576.3
	F3	12	N/A	N/A	N/A	14.69	17.86	15.20	14.58
	F4	9.03	N/A	N/A	N/A	572.9	577.7	135.6	2.892
	F5	4960	N/A	N/A	N/A	5389	3927	1416	2305
	F6	684	N/A	N/A	N/A	N/A	N/A	578.7	722.7
	F7	1340	N/A	N/A	N/A	1456	1347	1276	1222

changes substantially for higher dimensions. For $D = 100$, MGP-SLPSO, LSADE-C, and SAGWO dominate the less complicated functions, while LSADE-MQ and LSADE-C have the best results for the more complicated function. In direct comparison with LSADE, the best ones are GSGA and SAGWO (both 4/7 for both variants). For $D = 200$, only three of the six considered algo-

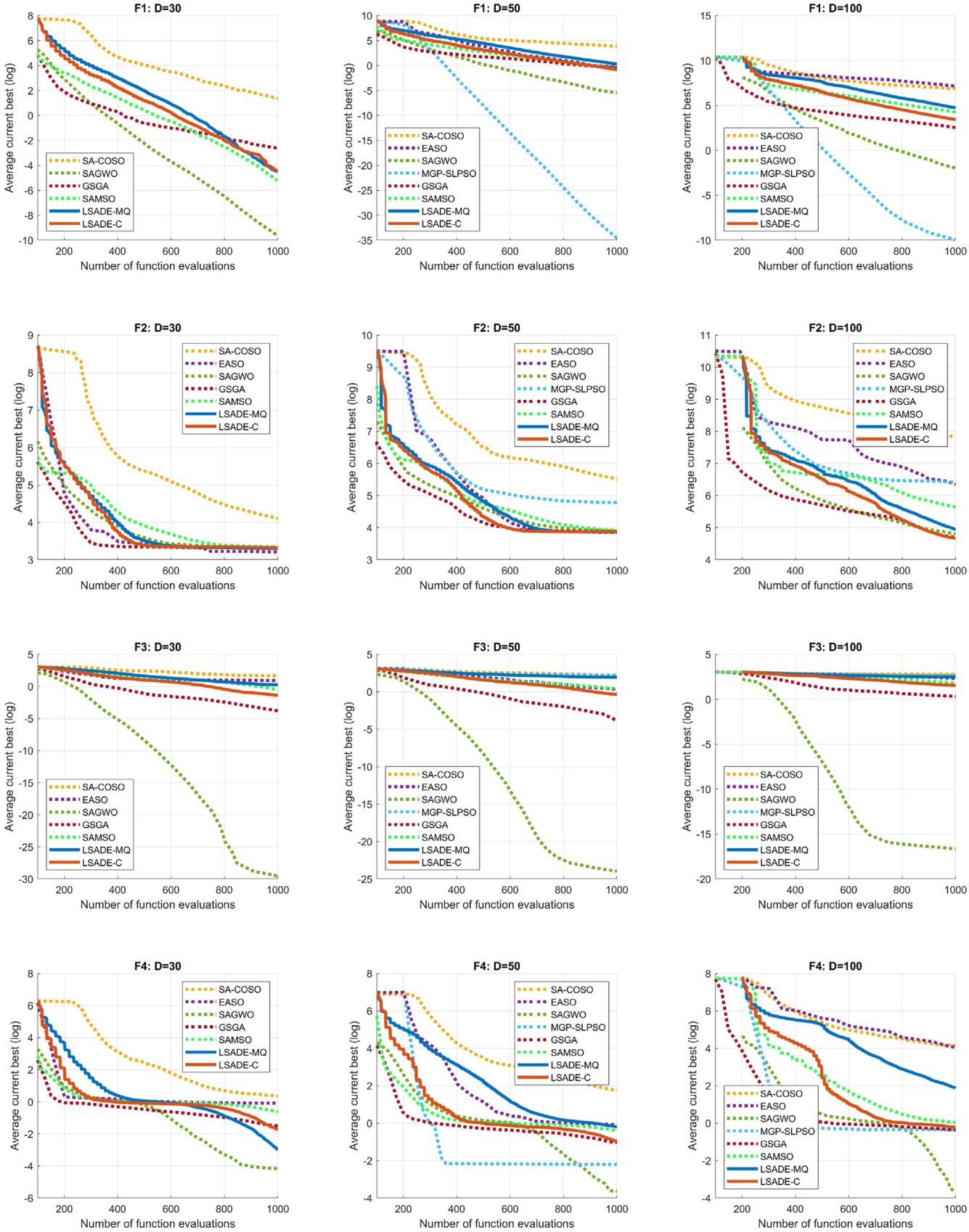


Fig. 3. Convergence history of the considered algorithms on the benchmark functions F1–F4 in dimensions $D = [30, 50, 100]$.

rithms reported results (possibly because of prohibitively large computational times as will be investigated in the following section). In these largest instances, LSADE-MQ and LSADE-C were the best choices for all problems with the exception of F3 for which SAMSO was the best.

The convergence histories of the considered algorithms for $D = [50, 100, 200]$ are depicted in Figs. 3 and 4, where on the y axis are not the objective function values, but the difference between the objective function value and the corresponding optimum (otherwise, the log operator would fail for F5). For $D = 200$, the convergence histories of the six compared algorithms were not available, and the convergence history of LSADE can be found in the Appendix. From these results, it is quite clear that the LSADE algorithm with properly tuned rules for using the newly proposed Lipschitz surrogate model and local optimization procedure compares well to the state-of-the-art SAEAs, especially for the high-dimensional highly complicated benchmark problems.

4.6. Computational Complexity

For LSADE the computational complexity mainly consists of five parts: the computation time for initial search, creating and evaluating the local and global RBF surrogate models, creating and evaluating the Lipschitz model, local optimization, and real function evaluations. In the following, we focus on empirical analysis of the computational time for the surrogates

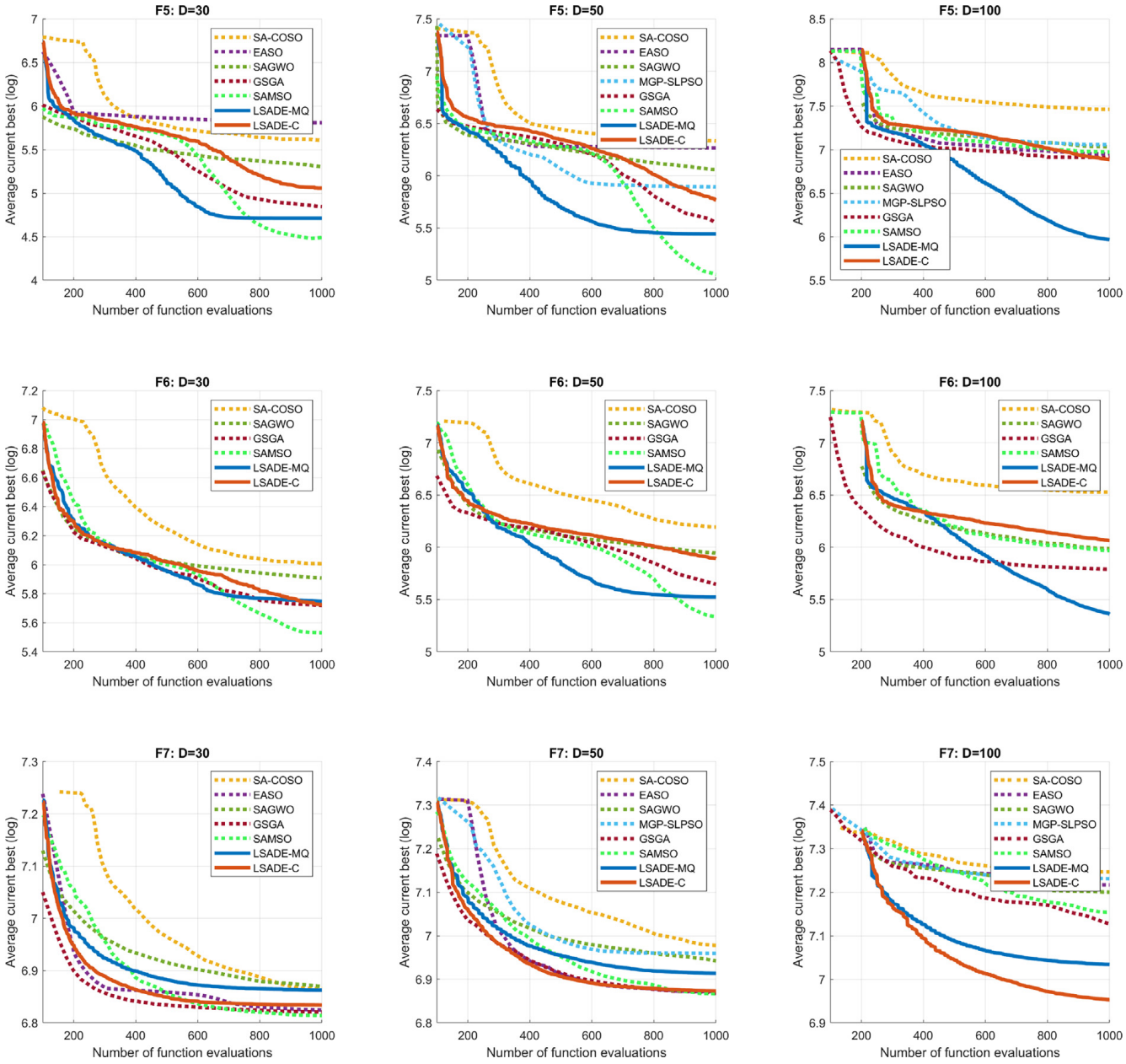


Fig. 4. Convergence history of the considered algorithms on the complicated benchmark functions F5–F7 in dimensions $D = [30, 50, 100]$.

and the local optimization procedure, as the time for real function evaluations depends on the problem the algorithm is applied to solve (these evaluations are expected to be costly, otherwise the algorithm should not be used). First, we compare the computational times for the individual components of the LSADE algorithm, using the R0 | Li0 | Lo1, R0 | Li1 | Lo0, and R1 | Li0 | Lo0 variants of the algorithm for the computation of the benchmark problems for $D = [30, 50]$. The results of these computations are reported in Table 7. We observe that the computation of Lipschitz surrogate model is significantly less computationally demanding than the computation of the (multiquadratic) RBF surrogate model. Unsurprisingly, the computational requirements for the local optimization are quite large, as these computations also contain the construction of the local RBF surrogate model.

The computational requirements for different variants of the LSADE algorithm will differ based on the number of RBF and Lipschitz surrogate evaluations, and on the number of times the local optimization procedure is used. The number of times these individual components were used for the variant of LSADE that was chosen for numerical comparisons (Li1–4 | Lo8–1), as well as for the other variants can be found in the Appendix. The average computational times of LSADE-MQ and LSADE-C for the benchmark problems for $D = [30, 50, 100, 200]$ can be found in Table 8. The computational times for different variants of LSADE as well as for different basis functions can also be found in the Appendix. Also in Table 8 are the computational times of SA-COSO, MGP-SLPSO, and SAGWO that were reported in the respective papers. As for the other compared algorithms, GSGA reported a computational time of 3 h for the function F3 in $D = 100$, and EASO and SAMSO did not include an empirical analysis of computational complexity. This comparison gives a clue as to why were the MGP-SLPSO, SAGWO, and GSGA algorithms not used for solving the large $D = 200$ problems – the computational times become a bit prohibitive for a large number of runs on numerous benchmark functions (but not necessarily prohibitive for a real application). On the other hand, the computational requirements for LSADE remain relatively low, with a dependence on the problem dimension that is roughly quadratic (at least for the considered benchmark problems). This is another indication that the LSADE algorithm is well suited for high-dimensional expensive problems.

Table 7
Computational time [s] of the individual components of LSADE, $D = [30, 50]$.

D	F	R0 | Li0 | Lo1	R0 | Li1 | Lo0	R1 | Li0 | Lo0
30	F1	76.35	3.44	38.96
	F2	43.12	3.36	38.97
	F3	46.59	3.19	38.36
	F4	30.02	3.03	37.46
	F5	70.94	3.40	39.10
	F6	55.24	7.01	43.27
	F7	74.55	7.03	44.49
50	F1	239.17	5.46	51.15
	F2	158.25	5.57	51.57
	F3	153.27	5.41	53.34
	F4	85.77	5.78	53.40
	F5	231.53	5.75	52.91
	F6	170.06	9.62	56.26
	F7	229.90	9.43	55.92

Table 8
Comparison with other algorithms, computational times [s].

D	SA-COSO	MGP-SLPSO	SAGWO	LSADE-MQ	LSADE-C
30	N/A	N/A	226	33.24	54.14
50	595	666	428	59.8	83.45
100	833	741	1099	164.1	167.7
200	N/A	N/A	N/A	591.3	685.6

Table 9
Dependence of computational time [s] of computing the Lipschitz constant estimate on dimension D and on the number of points for surrogate construction n.

		D					
		30	50	100	200	500	1000
n	30	2.13E-04	9.45E-05	1.34E-04	1.81E-04	2.76E-04	4.84E-04
	50	3.31E-04	3.03E-04	2.97E-04	4.52E-04	7.54E-04	1.32E-03
	100	9.77E-04	8.73E-04	1.11E-03	1.60E-03	2.96E-03	5.40E-03
	200	3.02E-03	3.41E-03	5.14E-03	6.27E-03	1.20E-02	2.13E-02
	500	1.87E-02	2.16E-02	2.73E-02	3.91E-02	7.56E-02	1.34E-01
	1000	6.98E-02	8.81E-02	1.13E-01	1.56E-01	3.11E-01	5.39E-01

Table 10
Dependence of computational time [s] for evaluating 10,000 points on the Lipschitz surrogate model on dimension D and on the number of points for surrogate construction n.

		D					
		30	50	100	200	500	1000
n	30	7.09E-02	6.81E-02	8.57E-02	1.19E-01	2.15E-01	3.92E-01
	50	8.97E-02	1.05E-01	1.32E-01	1.85E-01	3.43E-01	6.08E-01
	100	1.60E-01	1.88E-01	2.47E-01	3.53E-01	6.73E-01	1.19E+00
	200	3.07E-01	3.61E-01	4.81E-01	6.91E-01	1.33E+00	2.38E+00
	500	7.29E-01	8.88E-01	1.17E+00	1.69E+00	3.30E+00	5.93E+00
	1000	1.45E+00	1.76E+00	2.33E+00	3.37E+00	7.06E+00	1.22E+01

The complexity of the Lipschitz surrogate itself depends on two main operations: on the estimation of the Lipschitz constant and on the evaluation of the surrogate. Through empirical analysis (performed on F7) shown in Table 9 we can see that for the Lipschitz constant estimation there is a linear dependence of the computational time on the problem dimension D and a quadratic dependence on the number of evaluated points n. Similarly, in Table 10, we find that the computational time for evaluating the Lipschitz surrogate depends linearly on both D and n.

5. Conclusion

In this paper, we proposed a novel Lipschitz-based surrogate model for computationally expensive problems and used it to develop LSADE, a differential evolution-based surrogate-assisted evolutionary algorithm. The LSADE algorithm utilizes the combination of the Lipschitz-based and standard RBF surrogate models and a local optimization procedure to balance the exploration and the exploitation on a limited computational budget. The proposed LSADE algorithm was evaluated and its hyperparameters (such as the choice of the particular RBF surrogate and the frequency of its individual components) were tuned on a testbed of seven widely used 30, 50, 100, and 200 dimensional benchmark problems. The computational results show its effectiveness and competitiveness with other state-of-the-art algorithms, especially for complicated and high-dimensional problems.

There still remains much room for further improvements. The conditions for including new points based on the Lipschitz surrogate and local optimization could be made in an adaptive manner based on the progress of the algorithm. Similarly, the use of different RBFs or ensembles within the same algorithm, or the use of different evolutionary algorithms could also make the method more effective for certain classes of problems. The method could also be tested on a more diverse set of benchmark functions. Future work will also include the extension of the Lipschitz-based surrogate model to multifidelity and multicriteria optimization problems and its application to real-world problems.

CRediT authorship contribution statement

Jakub Kůudela: Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Radomil Matouše:** Funding acquisition, Methodology, Project administration, Resources, Supervision, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic project No. CZ.02.1.01/0.0/0.0/16_026/0008392 "Computer Simulations for Effective Low-Emission Energy" and by IGA BUT: FSI-S-20–6538.

Appendix A. Appendix A - Detailed Results for the Static Rules

In Table 11 are the detailed results for the static rules for $D = 50$. It shows, once again, that using both the Lipschitz surrogate model and the local optimization procedure provides substantial benefits. On its own, using the Lipschitz surrogate model was better than using the local optimization procedure for benchmark functions F3, F4, F6 and F7. However, the combinations of these two components are far superior for all considered benchmark functions.

Table 11
Results for the static rule, $D = 50$.

Li|Lo	F1	F2	F3	F4	F5	F6	F7
0|0	285.5	214.4	18.36	79.97	272.9	787.4	1229
0|1	3.728	65.41	17.99	191.9	20.03	752.7	1238
0|2	3.523	66.94	17.77	122.0	4.710	738.0	1231
0|4	22.32	69.03	17.71	73.31	14.93	703.4	1209
0|8	5.086	65.10	17.72	43.94	−6.35	697.0	1181
1|0	69.96	161.8	10.58	9.118	161.9	567.8	1047
2|0	41.16	112.8	13.81	5.002	204.2	597.8	1102
4|0	35.79	97.44	16.41	6.165	181.6	621.2	1133
8|0	34.67	90.68	16.95	8.976	216.4	654.0	1162
1|1	2.352	65.12	15.56	6.464	−138.0	410.4	1077
1|2	3.861	62.05	13.81	1.628	−132.2	363.2	1028
1|4	4.645	61.36	9.822	1.082	−120.9	364.7	1019
1|8	6.003	49.13	6.460	1.045	−106.2	389.8	1023
2|1	0.817	60.57	15.61	12.24	−76.34	454.4	1100
2|2	0.687	55.65	15.34	2.408	−92.19	423.2	1102
2|4	1.253	51.08	14.16	1.183	−90.54	423.9	1061
2|8	1.959	50.49	13.92	1.010	−60.46	440.9	1058
4|1	0.575	65.37	16.07	30.49	−66.03	558.6	1156
4|2	0.702	56.74	16.21	6.269	−64.67	544.1	1125
4|4	0.513	54.82	15.78	1.424	−45.27	491.7	1095
4|8	0.967	47.45	15.69	1.105	−46.09	468.2	1112
8|1	0.629	62.46	16.78	71.78	−18.57	615.8	1195
8|2	0.623	58.93	16.73	18.43	−40.99	570.5	1181
8|4	0.445	59.70	16.72	4.286	0.750	568.5	1156
8|8	0.708	48.66	16.50	1.587	−35.96	533.9	1150

Appendix B. Appendix B - Conditions for the Dynamic Rules

In Table 12 are the conditions used for the dynamic rules of the LSADE algorithm. The mod function gives the remainder after division (modulo operation) and $\lceil\cdot\rceil$ is the ceil operation that rounds the value inside to the nearest integer greater than or equal to that value.

Appendix C. Appendix C - Computational Complexity for Different Dynamic Rules and Basis Functions

The computational complexity of the different variants of the LSADE algorithm depends on the number of times the algorithm computed the RBF global and local models, the Lipschitz model and the local optimization procedure. Based on the rules described in Table 12, the number of evaluation of the individual components of the LSADE algorithm for the different variations of the dynamic rule are shown in Table 13.

In Table 14 are the computational times for the different variation of the dynamic rule for $D = [30, 50, 100]$. We can see that the computational effort is tied most directly to the number of times the local optimization procedure was used – the variants that use it more often needed more computational time, especially when the dimension of the problems increased. Another interesting observation can be made regarding the difference in computational complexity for the different benchmark functions – F1, F2, and F5 seem to require significantly more computational effort for the dynamic rules, especially in higher dimensions. We can compare this observation with the computational times for the individual components of LSADE that is reported in the paper. There, we can see that the computational times for local optimization procedure were quite

Table 12
Conditions for dynamic rules of the different variants of the LSADE algorithm.

Li|Lo	Lipschitz condition	Local Condition
1–4|8–1	$\mathrm{mod}(iter, \lceil\frac{8\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{8000-15\cdot iter}{1000}\rceil) = 0$
1–6|8–1	$\mathrm{mod}(iter, \lceil\frac{10\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{8000-15\cdot iter}{1000}\rceil) = 0$
1–8|8–1	$\mathrm{mod}(iter, \lceil\frac{14\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{8000-15\cdot iter}{1000}\rceil) = 0$
1–4|6–1	$\mathrm{mod}(iter, \lceil\frac{8\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{6000-12\cdot iter}{1000}\rceil) = 0$
1–6|6–1	$\mathrm{mod}(iter, \lceil\frac{10\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{6000-10\cdot iter}{1000}\rceil) = 0$
1–8|6–1	$\mathrm{mod}(iter, \lceil\frac{14\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{6000-10\cdot iter}{1000}\rceil) = 0$
1–4|4–1	$\mathrm{mod}(iter, \lceil\frac{8\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{4000-8\cdot iter}{1000}\rceil) = 0$
1–6|4–1	$\mathrm{mod}(iter, \lceil\frac{12\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{4000-8\cdot iter}{1000}\rceil) = 0$
1–8|4–1	$\mathrm{mod}(iter, \lceil\frac{15\cdot iter}{1000}\rceil) = 0$	$\mathrm{mod}(iter, \lceil\frac{4000-8\cdot iter}{1000}\rceil) = 0$

Table 13
Number of evaluations of the individual components of LSADE for different dynamic rules for $D = [30, 50]$

Li|Lo	global RBF surrogate	Lipschitz surrogate	local optimization (+local RBF)
1–4|8–1	495	260	145
1–6|8–1	510	231	159
1–8|8–1	531	189	180
1–4|6–1	471	254	175
1–6|6–1	512	231	157
1–8|6–1	533	189	178
1–4|4–1	445	248	207
1–6|4–1	469	200	231
1–8|4–1	483	172	245

Table 14
Computational time [s] for the different dynamic rules for $D = [30, 50, 100]$

D	F	1–4|8–1	1–6|8–1	1–8|8–1	1–4|6–1	1–6|6–1	1–8|6–1	1–4|4–1	1–6|4–1	1–8|4–1
30	F1	34.81	35.30	38.17	43.48	36.11	35.77	36.47	39.48	39.76
	F2	36.44	38.48	42.56	53.36	37.84	37.81	40.02	42.55	43.94
	F3	28.24	33.15	35.92	39.94	31.07	30.67	30.01	33.20	33.78
	F4	29.62	33.24	36.32	40.36	32.01	32.38	31.52	32.34	34.22
	F5	36.27	40.97	45.66	41.19	38.88	43.65	39.14	40.21	41.59
	F6	36.95	39.75	48.67	43.02	40.63	47.03	39.86	40.17	42.91
	F7	30.39	34.61	42.91	35.24	32.88	40.35	33.63	34.65	36.38
50	F1	66.73	69.96	75.20	84.41	73.73	90.86	82.39	89.50	88.31
	F2	70.27	74.56	81.22	97.81	81.04	94.28	93.03	97.69	97.13
	F3	44.13	46.28	49.45	59.08	51.52	61.87	58.04	58.47	58.86
	F4	49.63	52.70	58.52	64.85	59.76	75.67	63.64	70.84	74.52
	F5	70.52	74.81	80.54	97.25	77.47	94.39	91.03	91.95	94.57
	F6	64.46	69.44	73.05	80.62	69.74	77.45	82.91	85.41	86.28
	F7	52.90	55.64	61.47	58.95	58.13	66.13	66.27	69.47	71.75
100	F1	194.43	209.00	228.94	237.74	229.36	234.38	270.51	301.35	312.44
	F2	200.79	225.49	244.92	256.21	239.94	248.38	287.85	327.42	331.41
	F3	124.03	146.51	137.47	150.47	144.99	147.01	174.52	203.86	182.72
	F4	136.74	161.40	185.65	163.92	168.58	185.41	199.25	238.07	238.36
	F5	202.60	211.31	246.00	239.35	231.15	242.88	280.10	309.55	360.38
	F6	163.63	170.32	199.43	187.03	176.06	199.93	219.83	246.08	276.70
	F7	126.78	138.73	170.19	151.88	150.29	168.68	173.01	207.39	234.45

Table 15
Computational time [s] for the different basis functions, $D = [30, 50, 100, 200]$.

D	F	MQ	Cubic	TPS	Linear	Gaussian
30	F1	34.81	54.53	43.40	33.71	45.88
	F2	36.44	58.33	49.46	32.42	50.65
	F3	28.24	51.36	41.60	48.91	51.99
	F4	29.62	62.10	54.11	54.30	51.92
	F5	36.27	54.06	44.84	33.63	38.53
	F6	36.95	53.23	51.09	37.53	43.71
	F7	30.39	45.41	49.36	38.11	40.61
50	F1	66.73	102.76	92.01	44.63	86.73
	F2	70.27	108.29	85.52	48.05	85.45
	F3	44.13	64.07	62.10	77.25	76.60
	F4	49.63	93.61	84.86	76.13	77.39
	F5	70.52	76.13	70.61	41.16	47.43
	F6	64.46	72.62	73.43	44.33	53.72
	F7	52.90	66.70	59.00	43.56	52.37
100	F1	194.43	243.85	201.95	93.25	214.97
	F2	200.79	257.15	209.09	104.64	189.06
	F3	124.03	171.90	191.66	168.00	175.51
	F4	136.74	179.88	179.44	167.52	131.28
	F5	202.60	107.27	90.91	70.36	84.71
	F6	163.63	104.77	88.37	80.79	88.68
	F7	126.78	109.52	93.51	90.94	96.84
200	F1	883.57	1078.10	–	–	–
	F2	847.72	1114.60	–	–	–
	F3	446.99	801.77	–	–	–
	F4	420.28	501.90	–	–	–
	F5	640.02	546.68	–	–	–
	F6	467.05	327.74	–	–	–
	F7	433.72	428.97	–	–	–

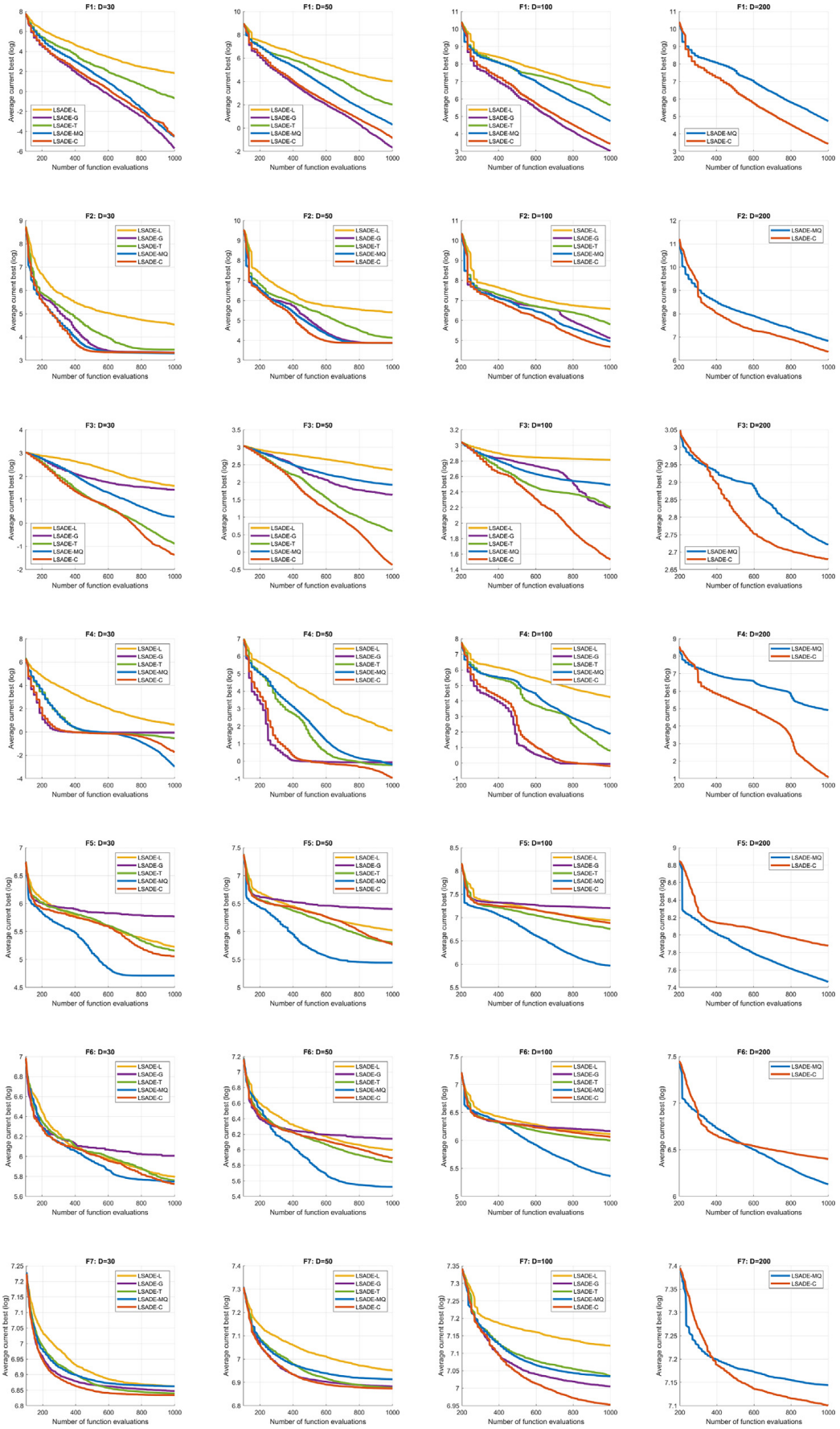


Fig. 5. Convergence history of LSADE with different basis functions on the benchmark functions F1–F7 in different dimensions.

Table 16
Detailed statistics of the results for SAMSO, MGP-SLPSO, GSGA, and SAGWO algorithms on all considered benchmark functions.

		SAMSO		MGP-SLPSO				GSGA				SAGWO			
D	F	mean	std	best	mean	worst	std	best	mean	worst	std	best	mean	worst	std
30	F1	0.0053	0.0057	N/A	0	N/A	0	0.0051	0.072	0.326	0.093	0.00001	0.000065	0.0003	0.000075
	F2	28.3	0.854	N/A	100	N/A	22.3	25.69	27.59	29.04	1.295	26.79	28.29	28.83	0.517
	F3	0.628	0.542	N/A	6.58	N/A	2.6	0.0065	0.023	0.057	0.023	0	0	0	0
	F4	0.538	0.144	N/A	0.013	N/A	0.005	0.095	0.228	0.383	0.222	0.000001	0.015	0.134	0.032
	F5	−239	24.3	N/A	−220	N/A	19.6	−245.2	−203	−159.0	24.87	−176	−128.8	−58.71	30.82
	F6	372	14.7	N/A	N/A	N/A	N/A	275.5	424.7	563.1	106.2	348.4	489.8	675.8	128.8
	F7	922	3.66	N/A	952	N/A	19	918.8	927.2	938.8	6.043	942.5	973.2	1016	18.47
50	F1	0.513	0.285	0	0	0	0	0.203	0.621	1.868	0.484	0.0007	0.004	0.015	0.0036
	F2	50.1	0.768	88.4	120	165	18.7	46.84	48.21	49.10	0.766	48.35	49.06	49.94	0.449
	F3	1.53	0.436	7.77	9.31	12.1	1.13	0.0060	0.021	0.076	0.023	0	0	0	0
	F4	0.666	0.107	0.037	0.154	0.614	0.13	0.272	0.346	0.442	0.071	0.000035	0.025	0.230	0.058
	F5	−169	31.7	−43.4	33	88.4	36.1	−139.6	−75.82	12.09	49.99	−16.63	98.39	161.5	46.90
	F6	326	98.6	N/A	N/A	N/A	N/A	271.8	403.3	524.8	87.59	430.2	502	564.2	45.25
	F7	970	29.2	1030	1060	1110	21.4	943.7	970.7	1002	18.18	910	1044	1132	40.83
100	F1	72.1	17.8	0	0.00005	0.001	0.0002	2.603	12.32	27.15	9.394	0.017	0.139	0.371	0.097
	F2	286	52.5	455	612	733	67.9	99.743	109.0	139.3	11.76	104.9	123.4	144.8	11.02
	F3	6.12	0.409	13.4	14.3	15.7	0.621	0.156	1.31	2.807	0.968	0	0	0	0
	F4	1.06	0.026	0.478	0.715	0.847	0.724	0.580	0.706	0.804	0.070	0.00021	0.023	0.229	0.052
	F5	737	42	877	885	1160	117	620.4	672.5	705.2	29.79	676.7	800.1	919.0	79.27
	F6	513	18.5	N/A	N/A	N/A	N/A	422.4	447.2	472.6	14.25	482.0	518.6	555.3	20.54
	F7	1290	33.4	1330	1390	1490	47.7	1220	1256	1287	24.56	910.2	1350	1437	107.5
200	F1	1520	21.2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F2	1150	11.6	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F3	12	0.4	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F4	9.03	1.33	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F5	4960	138	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F6	684	37.2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	F7	1340	24.3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

high for problems F1, F5, and F7, while the other two components had only small dependence of computational time on the benchmark function.

However, when we look at the computational complexity for different basis functions, that is reported in Table 15, we see that this dependence on the benchmark function is not shared among them. What we see instead is that for each choice of a basis function there are benchmark functions for which the computations seem to be more "difficult", regardless of dimension. For instance, F3 and F4 need more computational time for the linear basis function, while being among the "easiest" for the multiquadratic basis function. This could be explained by the different nature (and, thus, different "difficulty") of the local RBF models for the sequential quadratic programming optimizer that is used as the local optimization procedure.

Appendix D. Appendix D - Convergence Histories for Different Basis Functions

The convergence histories for different basis functions are depicted in Fig. 5. We can see that, most of the time, the best variant (i.e., the best choice of the basis function) of LSADE for a particular problem instance did not plateau around the 1000 real function evaluation limit. Also, the best performing variant for the particular problem instance (i.e, the one that had be best result after 1000 real function evaluations) is not necessarily the one that was the best when the number of real function evaluations was smaller. This phenomenon can be clearly observed for the $D = 200$ benchmark problems, where the convergence histories for cubic and multiquadratic basis functions cross one another for the majority of the considered benchmark functions. This suggests that it may be beneficial to consider several basis functions in an ensemble at the same time and find a rule for using one of them based on the properties of the particular problem.

Appendix E. Appendix E - Detailed Results for the Algorithms Considered for the Comparison

In Tables 16 and 17 are detailed results of the computations of the six algorithms considered for comparison and two two LSADE variants (LSADE-MQ and LSADE-C). These detailed results were obtained from the respective publications, with the expection of the results for EASO and SA-COSO that were obtained from the SAMSO paper, and contain the best value, mean, the worst value, and standard deviation from the corresponding computational experiments (for some algorithms, some of these statistics were not available, and not all of the algorithms were tried on all of the benchmark functions).

From these results, we can see that although the LSADE variants are mediocre for the benchmark problems in smaller dimensions, they are performing very well in the dimensions $D = [100, 200]$, especially for the benchmark functions F5-F7

Table 17
Detailed statistics of the results for EASO, SA-COSO, LSADE-MQ, and LSADE-C algorithms on all considered benchmark functions.

		EASO		SA-COSO		LSADE-MQ				LSADE-C			
D	F	mean	std	mean	std	best	mean	worst	std	best	mean	worst	std
	F1	0.027	0.070	3.85	1.19	0.0039	0.011	0.021	0.005	0.0008	0.011	0.047	0.012
	F2	25.04	1.57	59.9	24.3	24.31	27.06	29.35	1.243	27.20	27.77	29.36	0.546
	F3	2.521	0.84	5.01	1.22	0.026	1.308	3.028	1.011	0.0025	0.256	1.186	0.441
30	F4	0.953	0.05	1.44	0.18	0.0098	0.051	0.107	0.027	0.046	0.176	0.673	0.172
	F5	6.325	26.5	−57.4	17.5	−278.2	−218.7	−136.0	35.68	−256.2	−172.6	−81.31	39.83
	F6	N/A	N/A	528	94.8	229.2	433.7	664.1	149.3	233.5	426.2	674.3	148.1
	F7	931.6	8.94	969	24.3	922.2	965.7	1097	51.86	916.1	938.8	1004.3	26.37
	F1	0.740	0.555	46.6	17.4	0.265	1.358	3.500	0.860	0.047	0.433	1.304	0.299
	F2	47.39	1.71	253	56.7	43.92	47.65	49.17	1.332	45.53	47.98	49.19	0.864
	F3	1.431	0.249	8.86	1.1	2.615	6.876	15.39	3.456	0.029	0.695	2.264	0.600
50	F4	0.94	0.042	5.63	0.892	0.560	0.819	1.051	0.132	0.198	0.38	0.662	0.129
	F5	198.6	45.8	235	40.9	−194.6	−98.78	−5.288	52.92	−183.0	−10.03	151.2	93.88
	F6	N/A	N/A	613	37.4	259.0	370.3	579.8	109.5	339.2	481.6	657.1	80.89
	F7	975.3	37.1	1080	36.6	954.3	1016	1134	53.369	936.1	976.3	1103	38.52
	F1	1283	134	985	214	58.02	112.8	171.2	33.61	12.93	30.94	61.73	12.46
	F2	578.8	44.8	2500	97.4	108.3	140.6	194.3	24.10	97.62	106.4	120.8	6.631
	F3	10.36	0.211	15.9	0.514	9.431	12.05	16.54	2.203	3.540	4.622	6.157	0.619
100	F4	57.34	5.84	63.5	14.9	3.344	6.517	10.04	1.974	0.694	0.816	0.923	0.059
	F5	713.4	26.5	1420	123	−71.63	60.28	426.2	121.0	503.0	646.8	768.0	64.37
	F6	N/A	N/A	807	65.7	267.3	332.7	419.4	37.77	486.8	550.4	688.2	43.30
	F7	1372	27.5	1410	22.8	1076	1144	1232	44.45	1002	1056	1146	34.27
	F1	17616	1170	16382	2980	2473	3959	5192	705.2	587.6	793.5	1137	154.3
	F2	4318	284	16411	4100	683.0	927.2	1087	112.4	507.3	576.3	662.5	46.35
	F3	14.69	0.219	17.86	0.022	13.97	15.2	16.08	0.509	11.59	14.58	17.30	1.400
200	F4	572.9	36	577.7	101	93.89	135.6	188.2	22.05	2.149	2.892	3.456	0.394
	F5	5389	157	3927	27.3	1114	1416	2034	287.1	2042	2305	2625	156.8
	F6	N/A	N/A	N/A	N/A	474.5	578.7	883.3	88.76	541.2	722.7	818.7	60.65
	F7	1456	20.4	1347	24.7	1226	1276	1352	28.15	1140	1222	1274	33.38

with a more complicated multimodal landscape. For instance, the worst solution obtained by LSADE-MQ in $D = 100$ for benchmark functions F5-F7 was better than the mean of the solutions of all other compared algorithms (except for LSADE-C).

References

[1] Y. Jin, B. Sendhoff, A systems approach to evolutionary multiobjective structural optimization and beyond, IEEE Comput. Intell. Mag. 4 (3) (2009) 62–76.
[2] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, K. Giannakoglou, Metamodel-Assisted evolution strategies, in Parallel Problem Solving From Nature–PPSN, Springer, Heidelberg, Germany, 2002, pp. 361–370.
[3] Y. Jin, H. Wang, T. Chugh, D. Guo, K. Miettinen, Data-Driven Evolutionary Optimization: An Overview and Case Studies, IEEE Trans. Evol. Comp. 23 (3) (2019) 442–458.
[4] Y. Jin, Surrogate-assisted evolutionary computation: Recent advances and future challenges, Swarm Evol. Comput. 1 (2) (2011) 61–70.
[5] R.H. Myers, D.C. Montgomery, Response Surface Methodology: Process and Product in Optimization Using Designed Experiments, Wiley, New York, NY, USA, 1995.
[6] J.D. Martin, T.W. Simpson, Use of Kriging models to approximate deterministic computer models, AIAA J. 43 (4) (2005) 853–863.
[7] Y. Jin and B. Sendhoff, Reducing fitness evaluations using clustering techniques and neural network ensembles, in Proc. Genet. Evol. Comput. Conf., Seattle, WA, USA, 2004, pp. 688–699.
[8] N. Dyn, D. Levin, S. Rippa, Numerical procedures for surface fitting of scattered data by radial functions, SIAM J Sci. Stat. Comput. 7 (2) (1986) 639–659.
[9] S.R. Gunn, Support vector machines for classification and regression, Dept. Electron. Comput. Sci., Univ. Southampton, Southampton, U.K., Rep., 1998.
[10] R. Jin, W. Chen, T.W. Simpson, Comparative studies of metamodelling techniques under multiple modelling criteria, Struct. Multidiscipl. Optim. 23 (1) (2001) 1–13.
[11] S.R. Young, D.C. Rose, T.P. Karnowski, S.H. Lim, and R.M. Patton, Optimizing deep learning hyper-parameters through an evolutionary algorithm, in Proceedings of the workshop on machine learning in high-performance computing environments, pp. 1–5, 2015.
[12] R. Matousek, P. Popela, J. Kudela, Heuristic approaches to stochastic quadratic assignment problem: Var and cvar cases, MENDEL 23 (2017) 73–78.
[13] P. Zufan, M. Bidlo, Advances in evolutionary optimization of quantum operators, MENDEL 27 (2) (2021) 12–22.
[14] Z. Abo-Hammour, O. Alsmadi, S. Momani, and O.A. Arqub, A Genetic Algorithm Approach for Prediction of Linear Dynamical Systems, Mathematical Problems in Engineering, vol. 2013, Article ID 831657, 2013.
[15] O.A. Arqub, Z. Abo-Hammour, Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm, Information Sciences 279 (2014) 396–415.
[16] H. Rakhshani, L. Idoumghar, J. Lepagnot, M. Brévilliers, Speed up differential evolution for computationally expensive protein structure prediction problems, Swarm and Evolutionary Computation 50 (2019), paper ID 100493.
[17] H. Dong, X. Li, Z. Yang, L. Gao, Y. Lu, A two-layer surrogate-assisted differential evolution with better and nearest option for optimizing the spring of hydraulic series elastic actuator, Applied Soft Comp. 100 (2021), paper ID 107001.
[18] D. Wang, Z. Wu, Y. Fei, W. Zhang, Structural design employing a sequential approximation optimization approach, Comput. Struct. 134 (2014) 75–87.
[19] S. Wang, J. Liu, Y. Jin, Surrogate-Assisted Robust Optimization of Large-Scale Networks Based on Graph Embedding, IEEE Trans. Evol. Comp. 24 (4) (2020) 735–749.
[20] J. Kudela, R. Matousek, Recent Advances and Applications of Surrogate Models for Finite Element Method Computations: A Review, Soft Computing (2022), https://doi.org/10.1007/s00500-022-07362-8.
[21] R.G. Regis, Particle swarm with radial basis function surrogates for expensive black-box optimization, J. Comp. Sci. 5 (2014) 12–23.
[22] C. Sun, Y. Jin, J. Zeng, Y. Yu, A two-layer surrogate-assisted particle swarm optimization algorithm, Soft Comput. 19 (6) (2014) 1461–1475.
[23] B. Liu, Q. Zhang, G.G.E. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems, IEEE Trans. Evol. Comp. 18 (2) (2014) 180–192.
[24] F. Li, X. Cai, L. Gao, Ensemble of surrogates assisted particle swarm optimization of medium scale expensive problems, Appl. Soft Comput. 74 (2019) 291–305.
[25] Y.S. Ong, P.B. Nair, A.J. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling, AIAA J. 41 (4) (2003) 687–696.
[26] C. Sun, Y. Jin, R. Cheng, J. Ding, J. Zeng, Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems, IEEE Trans. Evol. Comp. 21 (4) (2017) 644–660.
[27] H. Wang, Uncertainty in surrogate models, in Proc. ACM Genet. Evol. Comput. Conf., 2016, p. 1279.
[28] D. Lim, Y. Jin, Y.-S. Ong, B. Sendhoff, Generalizing surrogateassisted evolutionary computation, IEEE Trans. Evol. Comp. 14 (3) (2010) 329–355.
[29] R. Mallipeddi, M. Lee, An evolving surrogate model-based differential evolution algorithm, Appl. Soft Comp. 34 (2015) 770–787.
[30] X.G. Zhou, G.J. Zhang, Abstract convex underestimation assisted multistage differential evolution, IEEE transactions on cybernetics 47 (9) (2017) 2730–2741.
[31] F. Li, X. Cai, L. Gao, W. Shen, A surrogate-assisted multi-swarm optimization algorithm for high-dimensional computationally expensive problems, IEEE Trans. Cyber. 51 (2021) 1390–1402.
[32] H. Wang, Y. Jin, J. Doherty, Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems, IEEE Trans. Cybern. 47 (2017) 2664–2677.
[33] X. Wang, G.G. Wang, B. Song, P. Wang, Y. Wang, A Novel Evolutionary Sampling Assisted Optimization Method for High-Dimensional Expensive Problems, IEEE Trans. Evol. Comput. 23 (5) (2019) 815–827.
[34] X. Cai, L. Gao, X. Li, Efficient Generalized Surrogate-Assisted Evolutionary Algorithm for High-Dimensional Expensive Problems, IEEE Trans. Evol. Comp. 24 (2020) 365–379.
[35] J. Tian, Y. Tan, J. Zeng, C. Sun, Y. Jin, Multiobjective Infill Criterion Driven Gaussian Process-Assisted Particle Swarm Optimization of High-Dimensional Expensive Problems, IEEE Trans. Evol. Comp. 23 (2019) 459–472.
[36] H. Dong, Z. Dong, Surrogate-assisted grey wolf optimization for high-dimensional, computationally expensive black-box problems, Swarm, Evol. Comput. 57 (2020), article no. 100713.
[37] A.I.J. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Progr. Aerosp. Sci. 45 (1–3) (2009) 50–79.
[38] A. Díaz-Manríquez, G. Toscano, C.A.C. Coello, Comparison of metamodeling techniques in evolutionary algorithms, Soft. Comput. 21 (2017) 5647–5663.
[39] S.A. Piyavskii, An algorithm for finding the absolute extremum of a function, USSR Computational Mathematics and Mathematical Physics 12 (4) (1972) 57–67.
[40] B.O. Shubert, A sequential method seeking the global maximum of a function, SIAM Journal on Numerical Analysis 9 (3) (1972) 379–388.
[41] C. Malherbe and N. Vayatis, Global optimization of Lipschitz functions, arXiv, arXiv:1703.02628, 2017.
[42] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comp. 15 (1) (2011) 4–31.
[43] R. Matousek, L. Dobrovsky, J. Kudela, How to start a heuristic? utilizing lower bounds for solving the quadratic assignment problem, International Journal of Industrial Engineering Computations 13 (2) (2022) 151–164.
[44] K.M. Sallam, S.M. Elsayed, R.K. Chakrabortty, and M.J. Ryan, Improved Multi-operator Differential Evolution Algorithm for Solving Unconstrained Problems, 2020 IEEE Congress on Evolutionary Computation (CEC), paper no. 19931315, 2020.

[45] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, Nat. Comput. (2005) 341–357.
[46] J. Kudela, R. Matousek, New Benchmark Functions for Single-Objective Optimization Based on a Zigzag Pattern, IEEE Access 10 (2022) 8262–8278.
[47] F.A.C. Viana. (2011). SURROGATES Toolbox User's Guide, Version 3.0. [Online]. Available: http://sites.google.com/site/felipeacviana/surrogatestoolbox.
[48] A. Kazikova, M. Pluhacek, R. Senkerik, Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison?, MENDEL 26 (2) (2020) 9–16

A9

KYBERNETIKA — VOLUME *53* (2017), NUMBER 6, PAGES 1012–1025

WARM-START CUTS FOR GENERALIZED BENDERS DECOMPOSITION

Jakub Kůdela and Pavel Popela

In this paper, we describe a decomposition algorithm suitable for two-stage convex stochastic programs known as Generalized Benders Decomposition. For this algorithm we propose a new reformulation that incorporates a lower bound cut that serves as a warm-start, decreasing the overall computation time. Additionally, we test the performance of the proposed reformulation on two modifications of the algorithm (bunching and multicut) using numerical examples. The numerical part is programmed in MATLAB and uses state-of-the-art conic solvers.

Keywords: stochastic programming, Generalized Benders Decomposition, *L*-shaped method, warm–start

Classification: 90C15, 90C25, 49M27

1. INTRODUCTION

In stochastic programming, we usually have to deal with problems that are large-scale but have a special structure [3]. Proper utilization of this special structure is the key part in the construction of any practically usable algorithm. One of the most widely used algorithms for two-stage stochastic linear programs is the *L*-shaped method developed by Van Slyke and Wets [12]. This method is based on (or, as the authors of the method wrote in the original paper: "is essentially the same as") the algorithm developed by Benders in [2] known as the Benders Decomposition. Over the years, numerous extensions for the *L*-shaped method have been proposed. A summary of the ones that are currently used can be found in [13] and [14].

A further generalization of the Benders decomposition for nonlinear convex problems ([1, 4]) was proposed by Geoffrion in [7] and was named the Generalized Benders Decomposition (GBD). The method found its main use as a solution technique for mixed-integer nonlinear problems, described in [5] and [6].

In this paper, we describe a formulation of the GBD that suits the particular structure of two-stage stochastic programming problems. After that, we introduce a reformulation that enables us to add a lower bound cut, which acts as a "warm-start" for the algorithm. As the lower bound cut, we decided to use the one that we can compute with the least effort. As there have been several lower bounds proposed for stochastic programs (for

DOI: 10.14736/kyb-2017-6-1012

example in [3] and [11]) the question of the appropriate one for our problem will be left open for future research.

2. GBD – MAIN IDEAS

In this section, we give a brief insight into the GBD, as it is not our intention to devote several pages to its thorough description. An interested reader can find an in-depth analysis of the method in the original paper [7] and in the works of Floudas in [5] and [6].

The problems GBD aims to solve are of the form:

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & f(x,y) \\ \text{subject to} & G(x,y) \leq 0, \quad x \in X, y \in Y, \end{array} \tag{1}$$

where $x \in X \subseteq \Re^{n_1}$, $y \in Y \subseteq \Re^{n_2}$, $f : \Re^{n_1} \times \Re^{n_2} \longrightarrow \Re$ is a real-valued objective function and $G : \Re^{n_1} \times \Re^{n_2} \longrightarrow \Re^m$ is an m-vector of constraint functions. The variable x is called a complicating variable in the sense that (1) is a much easier optimization problem in y when x is temporarily held fixed. The following conditions are required:

C1: Y is a nonempty, convex set and the functions f and G are convex for each fixed $x \in X$.

C2: The set

$$Z_x = \{z \in \Re^m : G(x,y) \leq z \text{ for some } y \in Y\} \tag{2}$$

is closed for each fixed $x \in X$.

C3: For each fixed $x \in X \cap V$, where

$$V = \{x : G(x,y) \leq 0, \text{ for some } y \in Y\}, \tag{3}$$

one of the following conditions holds:

(i) the problem (1) has a finite solution and has an optimal multiplier vector for the inequalities.

(ii) the problem (1) is unbounded, that is, its objective function value goes to $-\infty$.

This covers quite a wide range of problems [5]. The particular situation we are interested in is when f and G are linearly separable in x and y, i. e.

$$\begin{array}{rcl} f(x,y) & = & f_1(x) + f_2(y), \\ G(x,y) & = & G_1(x) + G_2(y). \end{array} \tag{4}$$

The basic idea in GBD is the generation, at each iteration, of an upper bound and a lower bound on the optimal solution of (1). The upper bound results from a subproblem, while the lower bound results from a master problem. The subproblem corresponds to the problem (1) with fixed x-variable (i. e., it is in the y-space only), and its solution

provides information about the upper bound and the Lagrange multipliers ([1, 4]) associated with the inequality constraints. The master problem is derived via nonlinear duality theory, makes use of the Lagrange multipliers obtained in the subproblem, and its solution provides information about the lower bound, as well as the next set of fixed x-variable to be used subsequently in the subproblem [5].

3. GBD FOR TWO-STAGE STOCHASTIC PROGRAMMING PROBLEMS

In stochastic programming linear separability of the objective function and constraints is a very common property. Especially the two-stage stochastic programming problems can be often linearly separated into the functions concerning only the first-stage and the second-stage decision variables – this is the raison d'etre of the following passages, and it is why we believe that the GBD (in its slightly modified form) is a well-suited algorithm for these kinds of problems.

3.1. Problem formulation

Let us consider the following problem:

$$\begin{aligned} \underset{x,y_1,\dots,y_K}{\text{minimize}} \quad & f_1(x) + \sum_{k=1}^{K} p(\xi_k) f_2(y_k, \xi_k) \\ \text{subject to} \quad & G_{11}(x) \le 0, \\ & G_{21}(\xi_k)x + G_{22}(y_k, \xi_k) \le 0,\ \xi_k \in \Xi, \end{aligned} \tag{5}$$

where $f_1 : \Re^{n_1} \longrightarrow \Re$ is a convex function, all m_1 constraint functions $G_{11} : \Re^{n_1} \longrightarrow \Re^{m_1}$ are convex, and for all $\xi_k \in \Xi$ with $|\Xi| = K$ finite, $G_{21}(\xi_k)$ is a $m_2 \times n_1$ matrix, $f_2(\cdot, \xi_k) : \Re^{n_2} \longrightarrow \Re$ is convex, all m_2 constraint functions $G_{22}(\cdot, \xi_k) : \Re^{n_2} \longrightarrow \Re^{m_2}$ are convex, $P(\xi = \xi_k) \equiv p(\xi_k) > 0, \sum_{k=1}^{K} p(\xi_k) = 1$.

The master problem corresponding to (5) has the following form:

$$\begin{aligned} \underset{x,\theta}{\text{minimize}} \quad & f_1(x) + \theta \\ \text{subject to} \quad & G_{11}(x) \le 0, \\ & D_i x \le d_i, \qquad i = 1, \dots, p, \\ & E_j x - \theta \le e_j, \quad j = 1, \dots, r, \end{aligned} \tag{6}$$

where $\theta \in \Re$ serves as the lower bound on the second stage objective value. The meaning of matrices D, E and vectors d, e will be fully discussed in the actual solution procedure. These matrices and vectors correspond to the feasibility and optimality cuts derived from the solutions of the subproblem.

Because of the structure of the two-stage stochastic programming problems, the subproblem separates into K independent subproblems (one for each scenario) in the form:

$$\begin{aligned} \underset{y_k}{\text{minimize}} \quad & f_2(y_k, \xi_k) \\ \text{subject to} \quad & G_{21}(\xi_k)x + G_{22}(y_k, \xi_k) \le 0. \end{aligned} \tag{7}$$

Remark 3.1. Regarding our notation – one could use k instead of ξ_k in the formulations above (and in the ones that will follow). The use of ξ is standard in the stochastic programming literature.

3.2. Solution procedure

The following algorithm is an implementation of the GBD inspired by [7] and [5]. The single difference (apart from the notation) is that the separability of the subproblem into K independent subproblems is taken into account. At the start of the procedure, the matrices D, E and vectors d, e are empty (they store the successive cuts as the iterations progress).

To our best knowledge, this is the first implementation of the GBD for two-stage stochastic convex programming problems of the form (5).

Step 0. Set $p = 0, r = 0$, and $\epsilon > 0$.

Step 1. Solve (6) and obtain $(\bar{x}, \bar{\theta})$. The optimal objective value of (6) gives us a lower bound on optimal objective value of (5).

Step 2. For fixed $x = \bar{x}$ solve all K subproblems (7). One of two possibilities can happen.

Step 2A. For some k the subproblem (7) is infeasible. Solve the following problem:

$$\begin{array}{ll} \underset{y_k, v \geq 0}{\text{minimize}} & ||v||_1 \\ \text{subject to} & G_{21}(\xi_k)\bar{x} + G_{22}(y_k, \xi_k) \leq v, \end{array} \tag{8}$$

where $v \in \Re^{m_2}$ is a decision vector representing "slacks" in the constraints. Get $(\bar{y}_k, \bar{v})$ and from its dual obtain the optimal Lagrange multipliers λ. Set $p = p+1$. Add a new row to the matrix D and vector d in (6):

$$D_p = \lambda^T G_{21}(\xi_k), d_p = \lambda^T(-G_{22}(\bar{y}_k, \xi_k)). \tag{9}$$

Return to Step 1.

Step 2B. All the subproblems have finite optimal values, we obtained $(\bar{y}_k, u_k)$, where u_k are optimal Lagrange multipliers. The evaluation of the objective of (5) at $(\bar{x}, \bar{y}_1, \ldots, \bar{y}_K)$ gives us an upper bound on its optimal value. Check for optimality: if

$$\bar{\theta} + \epsilon \geq \sum_{k=1}^{K} p(\xi_k) f_2(\bar{y}_k, \xi_k), \tag{10}$$

terminate, $(\bar{x}, \bar{y}_1, \ldots, \bar{y}_K)$ are ϵ-optimal [7]. Otherwise, set $r = r+1$ and add a new row to the matrix E and vector e in (6):

$$\begin{aligned} E_r &= \sum_{k=1}^{K} p(\xi_k)(u_k^T G_{21}(\xi_k)), \\ e_r &= -\sum_{k=1}^{K} p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T(G_{22}(\bar{y}_k, \xi_k)). \end{aligned} \tag{11}$$

Return to Step 1.

Remark 3.2. In Step 1, before any optimality cut is added, $\bar{\theta}$ as well as the optimal objective value of (6) will be $-\infty$. For computational reasons it is advisable to include a lower bound on θ in the actual implementation of the algorithm.

Remark 3.3. If $X \subseteq V$ (i. e., in the case of complete or relatively complete recourse [3]), the Step 2A is never needed and for a given $\epsilon > 0$ the GBD terminates in a finite number of iterations. If however, $X \nsubseteq V$, then we may need to solve Step 2A infinitely many successive times. In such a case, to preserve finite ϵ-convergence, we can modify the procedure so as to finitely truncate any excessively long sequence of successive executions of Step 2A and go to Step 2B with $\hat{x}$ equal to the extrapolated limit point which is assumed to belong to $X \cap V$, see [5] or [6].

4. REFORMULATION WITH A BOUNDING CUT

In this section, we introduce a novel reformulation of the master problem (6) that includes bounds obtained from problems, that can be thought of as predecessors of the two-stage stochastic programming problem (5). The definitions of these problems, as well as their subsequent relations, are based on [9].

4.1. Bounds

Let us define

$$\begin{array}{ll} \underset{x_k, y_k}{\text{minimize}} & f_1(x_k) + f_2(y_k, \xi_k) \\ \text{subject to} & G_{11}(x_k) \leq 0, \\ & G_{21} x_k + G_{22}(y_k, \xi_k) \leq 0, \end{array} \tag{12}$$

as the optimization problem for one particular realization $\xi_k \in \Xi$ and denote its optimal objective function value as $z(\xi_k)$ The wait-and-see solution is the solution without nonanticipativity constraints (i. e. all scenarios are treated and optimized separately). We will denote the average of the optimal objective values of (12) (when treated separately) as:

$$\mathrm{WS} = \sum_{k=1}^{K} p(\xi_k) z(\xi_k). \tag{13}$$

Now we may compare this wait-and-see solution to the solution of (5). We will denote the optimal objective value of (5) as RP (the recourse problem [3]). The following inequality holds for any stochastic program:

$$\mathrm{WS} \leq \mathrm{RP}. \tag{14}$$

From this, we can see that WS creates a valid lower bound on the harder problem we are aiming to solve. The idea behind the reformulation is to include such a valid lower bound to the algorithmic procedure to "jumpstart" it and by doing so save on iterations, and, as a result, save on the overall computational effort and time.

For practical purposes, many people would believe that finding the wait-and-see solution is still too much work. A natural temptation is to solve a much simpler problem:

the one obtained by replacing all random variables by their expected values. This is called the expected value problem, which is simply

$$\mathrm{EV} = z(\bar{\xi}), \tag{15}$$

where $\bar{\xi} = \sum_{k=1}^{K} p(\xi_k)\xi_k$.

4.2. Reformulation

Although WS is a valid bound, the computational effort for its enumeration is much higher compared to the effort to compute EV (if $|\Xi| = K$, then computing EV is K times faster). However, EV does not necessarily have to play the role of a lower bound on RP; there are instances, where RP $\leq$ EV. For the purpose of deriving the reformulation, we will, for now, suppose that EV is, in fact, a valid lower bound on RP. The discussion on what is going to occur when it is not will follow shortly after. Suppose

$$\mathrm{EV} \leq \mathrm{RP}, \tag{16}$$

holds, then

$$f_1(x) + \sum_{k=1}^{K} p(\xi_k) f_2(y_k, \xi_k) \geq \mathrm{EV}, \tag{17}$$

holds for the optimum of (5). This inequality cannot be added directly to (5) since it would cease to be a convex program. The reformulation we propose does not directly alter (5) but is instead aimed at the master problem (6). A new variable z is introduced to bound the first-stage objective from above (by minimizing this variable we effectively minimize the first-stage objective itself)

$$f_1(x) \leq z, \tag{18}$$

which is a convexity preserving inequality. Furthermore, this new variable z added to the variable representing the second stage θ form a lower bound on the overall objective function. Finally, the bound

$$z + \theta \geq \mathrm{EV}, \tag{19}$$

since it is affine, can be added to (6), and the reformulation of the problem is

$$\begin{aligned} \underset{z,x,\theta}{\text{minimize}} \quad & z + \theta \\ \text{subject to} \quad & f_1(x) \leq z, \\ & G_{11}(x) \leq 0, \\ & z + \theta \geq \mathrm{EV}, \\ & D_i x \leq d_i, \qquad i = 1, \ldots, p \\ & E_j x - \theta \leq e_j, \quad j = 1, \ldots, r. \end{aligned} \tag{20}$$

After this reformulation, the algorithm continues as usual, arriving at an ϵ-optimal solution in, preferably, a shorter time than its original counterpart (we will see the results of some numerical examples in later sections).

Now, let us address what happens if (16) does not hold. One of two possibilities can occur, namely, that optimal objective function value (as determined by the algorithm) will be equal to EV, or that the problem will be infeasible. The price we pay for mistakenly using the cuts (16) is, in both cases, one iteration of the algorithm – i. e. after one iteration we can assess, if our algorithm will arrive at the desired solution, and, either restart it without (16) (possibly including WS instead), or continue.

However, certain situations can happen when we restart the algorithm without (16) and get the same result again. This occurs if the original problem is infeasible (in which case we have some serious model or data issues) or if EV = RP, in that case we would have to run the entire algorithm only to arrive at the same objective function value (which is a bit unfortunate, but unavoidable).

Another important question is if the cut (16) is worth having an additional variable. The numerical examples we provide in the later sections should supply us with some, although not definitive, insight into this issue.

Lastly, the question whether or not it is better to use the guaranteed lower bound in WS is also present. As we mentioned earlier, WS is computationally much more expensive than EV. In the examples that will follow we did not carry any examination of the WS bound, nor of any other possible bound. This is one of the areas that require further future investigation.

The solution procedure can be summarized in the following steps:

Step 0. Solve the expected value problem to get EV (15). Set $p = 0, r = 0$, and $\epsilon > 0$. Solve (20) and obtain $(\bar{z}, \bar{x}, \bar{\theta})$. If $\bar{z} + \bar{\theta} = \text{EV}$, terminate (and use the original method without the EV cut, or use WS instead). Otherwise, go to Step 2.

Step 1. Solve (20) and obtain $(\bar{z}, \bar{x}, \bar{\theta})$.

Step 2., Step 2A., Step 2B. The same as in section 3.2.

5. BUNCHING AND MULTICUTS

Just as in the linear case with the L-shaped method, different implementations of the algorithm can be researched for improving its performance ([3, 13]). Two possible adjustments suitable for GBD – bunching and the multicut formulation will be discussed and brought into the numerical examination.

Bunching, as the name suggests, is a technique that instead of the full scenario decomposition uses "bunches" of scenarios and decomposes the original problem alongside these bunches. Having L bunches of scenarios and sets of indices $B_l \neq \emptyset, l = 1, \ldots, L$, such that $B_i \cap B_j = \emptyset$ for $i \neq j$ and $\bigcup_{l=1}^{L} B_l = \{1, \ldots, K\}$. The subproblems (7) for each bunch l have the form

$$\begin{array}{ll} \underset{y_k, k \in B_l}{\text{minimize}} & \displaystyle\sum_{k \in B_l} p(\xi_k) f_2(y_k, \xi_k) \\ \text{subject to} & G_{21}(\xi_k)x + G_{22}(y_k, \xi_k) \leq 0, k \in B_l. \end{array} \tag{21}$$

The feasibility and optimality cuts in Step 2A. and Step 2B. of the algorithm are changed accordingly. The feasibility cut in Step 2A. becomes

$$D_p = \sum_{k \in B_l} \lambda_k^T G_{21}(\xi_k), \quad d_p = -\sum_{k \in B_l} \lambda_k^T G_{22}(\bar{y}_k, \xi_k), \tag{22}$$

and the optimality cut in Step 2B. becomes

$$\begin{aligned} E_r &= \sum_{l=1}^{L} \sum_{k \in B_l} u_k^T G_{21}(\xi_k), \\ e_r &= -\sum_{l=1}^{L} \sum_{k \in B_l} p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T G_{22}(\bar{y}_k, \xi_k), \end{aligned} \tag{23}$$

where λ_k and u_k are the Lagrange multipliers corresponding to the inequalities from scenario $k \in B_l$.

In the linear case, bunching comes from the idea that several second-stage problems might have the same optimal basis [3]. In the convex case, the justification is a bit different. Our argumentation is purely in the realm of the actual computation – it is sometimes faster (due to a non-zero initialization time, etc.) to compute a larger instance containing several separable problems than to solve these problems separately. The examples will show, up to a certain point, exactly this kind of behavior.

The multicut formulation comprises of developing one cut for every second-stage problem (i. e. for every scenario) instead of the aggregated cut introduced in (6). It results in adding a separate θ_k for each scenario and as a consequence in a much greater number of cuts which more accurately describe the recourse function [3]. The master problem for multicut formulation has the following form (without the additional cut developed in the previous section)

$$\begin{aligned} \underset{x,\theta_1,\ldots,\theta_K}{\text{minimize}} \quad & f_1(x) + \sum_{k=1}^{K} \theta_k \\ \text{subject to} \quad & G_{11}(x) \leq 0, \\ & D_i x \leq d_i, \qquad\qquad i = 1, \ldots, p, \\ & E_{j(k)} x - \theta_k \leq e_{j(k)}, \quad j(k) = 1, \ldots, r(k), \\ & k = 1, \ldots, K, \end{aligned} \tag{24}$$

where $r(k)$ and $j(k)$ indices are related to the kth subproblem, see the steps below. In this case, the feasibility cuts remain the same, but the remaining steps require the following changes:

Step 0. – Multicut Set $p = 0, r(k) = 0$, for $k = 1, \ldots, K$ and $\epsilon > 0$.

Step 1. – Multicut Solve (24) and obtain $(\bar{x}, \bar{\theta}_1, \ldots, , \bar{\theta}_K)$.

Step 2. – Multicut For fixed $x = \bar{x}$ solve all K subproblems (7). One of two possibilities can happen.

Step 2A. – Multicut As before.

Step 2B. – Multicut All the subproblems have finite optimal values, we obtained $(\bar{y}_k, u_k)$, where u_k are optimal Lagrange multipliers. For $k = 1, \dots, K$ if

$$\bar{\theta}_k + \epsilon \leq p(\xi_k) f_2(\bar{y}_k, \xi_k), \tag{25}$$

set $r(k) = r(k) + 1$ and add a new row to the matrix E and vector e in (24):

$$\begin{aligned} E_{r(k)} &= p(\xi_k)(u_k^T G_{21}(\xi_k)), \\ e_{r(k)} &= -p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T G_{22}(\bar{y}_k, \xi_k)). \end{aligned} \tag{26}$$

If (25) does not hold for any k, terminate. Otherwise, return to Step 1.

Even though this formulation provides a more accurate description of the recourse function, its usefulness in the convex case is highly ambiguous. The number of variables in the master problem is much larger than in the original algorithm and the number of constraints (cuts) added in each iteration is also much higher.

6. NUMERICAL EXAMPLES

To test the above mentioned theoretical concepts, we designed two convex two-stage problems. On these problems, we compare the performance of different variations of the GBD as well as a formulation without any decomposition (denoted as full recourse problems).

The implementation was done in MATLAB using its embedded `fmincon` solver and the state-of-the-art conic solvers `SeDuMi` and `SDPT3` [10] (which are a part of the CVX modeling system [8]). Although the examples are not derived from any applied problems, they provide a valid insight into the advantages and disadvantages of the presented methods.

6.1. Example 1

The first example investigates the following problem

$$\begin{aligned} \underset{\boldsymbol{x}, \boldsymbol{y_1}, \dots, \boldsymbol{y_k}}{\text{minimize}} \quad & (x_1 - 4)^4 + (x_2 - 3)^4 + \sum_{k=1}^{K} p_k (q_{k,1} e^{y_{k,1}} + q_{k,2} y_{k,2}^4) \\ \text{subject to} \quad & x_2 - \ln(x_1 + 1) - 1 \leq 0, \\ & x_2 + x_1^3 - 8 \leq 0, \\ & x_1, x_2 \geq 0, \\ & x_1 + h_{k,1} - y_{k,1} \leq 0, k = 1, \dots, K, \\ & x_2 + h_{k,2} - y_{k,2} \leq 0, k = 1, \dots, K, \end{aligned}$$

where the random parameters q and h are $q \sim |N(0, 3)|$, $h \sim 0.7 \cdot |N(0, 1)| + 0.5$. The scenarios are then constructed using the usual Monte Carlo sampling, the number of scenarios will vary to demonstrate the performance of the different approaches. The methods and solvers used for solving the problem were:

- vanilla (original) version of GBD (master and subproblems solved by `fmincon`);

- reformulation with the EV cut (master and subproblems solved by `fmincon`);
- bunching of several scenarios (master and subproblems solved by `fmincon`);
- bunching of several scenarios with the EV cut (master and subproblems solved by `fmincon`);
- full recourse problem (FRP) solved by `fmincon`;
- FRP solved by `SDPT3` (as a part of the CVX modeling system);
- FRP solved by `SeDuMi` (as a part of the CVX modeling system).

The required precision for all the methods was set to $\epsilon = 10^{-5}$. The results are summarized in the tables that follow. The Time[s] value represents the computational time it took the procedure to terminate, given the same level of accuracy for all methods. The number of scenarios in the first instance is $K = 60$. The first two tables show, how the computational time of the GBD is affected by introducing the EV cut:

Method	Vanilla	EV cut
Time[s]	12.26	9.05

Tab. 1. Computational time [s] for Vanilla version and EV cut, $K = 60$.

and by bunching with different sizes of the bunch:

Bunch size	2	3	5	10	12	15	20	30
Time[s] – Without EV cut	7.04	5.08	3.61	2.76	2.64	2.65	2.75	3.22
Time[s] – With EV cut	5.19	3.79	2.75	2.07	2.02	1.99	2.13	2.44

Tab. 2. Computational time [s] for bunching with different sizes of the bunch, $K = 60$.

An identical structure is utilized in the case of $K = 240$ scenarios:

Method	Vanilla	EV cut
Time[s]	43.84	35.42

Bunch size	2	3	5	6	8	10	12	15
Time[s] – Without EV cut	24.8	17.7	12.4	11.4	9.9	9.2	8.8	8.7
Time[s] – With EV cut	19.9	14.3	10.2	9.2	8.0	7.4	7.2	7.0
Bunch size	16	20	24	30	40	60	80	120
Time[s] – Without EV cut	8.7	8.9	9.4	10.5	12.2	16.8	21.5	25.3
Time[s] – With EV cut	7.0	7.2	7.7	9.5	9.8	13.6	17.3	20.2

Tab. 3. Computational time [s] for Vanilla, EV cut and bunching, $K = 240$.

From these result, we see that the EV cut, as well as efficient bunching, can have a strong effect on the overall computation time. The experiments suggest that there exists an "optimal" bunch size that is independent of the number of scenarios. For this particular problem, it seemed that a bunch size between 12 and 16 was the one. For the subsequent computations, the bunch size 15 was chosen.

In the following table, we compare the computation times for growing number of scenarios using the methods and solvers mentioned above:

Number of scenarios	60	240	1,500	2,400	4,800	6,000
Vanilla	12.3	43.8	258.6	410.1	–	–
EV cut	9.1	35.4	208.9	332.7	–	–
Bunch 15	2.7	8.7	52.5	78.0	154.75	194.9
Bunch 15 with EV	1.9	7.0	40.1	62.6	124.6	156.4
FRP – `SDPT3`	6.1	22.3	139.9	241.7	–	–
FRP – `SeDuMi`	1.4	6.2	32.8	65.2	170.3	242.5
FRP – `fmincon`	0.5	12.2	3,000*	3,000*	–	–

Tab. 4. Computational time [s] for different methods, increasing number of scenarios.

The asterisk(*) denotes that the algorithm did not arrive at the desired precision (i. e. even after 3,000s the `fmincon` did not arrive sufficiently near the optimum). The dash(–) means that we did not pursue the analysis in this direction since we anticipated results incomparable with the more efficient methods.

These results show that for big enough problems, the efficient implementation GBD, even with simpler solvers, can outperform the state-of-the-art solvers. For smaller instances, however, these solvers are more efficient (as will be presented in the results of the second example).

6.2. Example 2

The second example included in our investigation, compared to the first one, adds some more first and second-stage variables and non-differentiable functions. These are the reason why, in the implementation, the more efficient solvers had to be utilized for the solution of the master problem (`fmincon` performed very poorly in this case). The problem in question is the following

$$\begin{aligned}\underset{\boldsymbol{x},\boldsymbol{y_1},\ldots,\boldsymbol{y_K}}{\text{minimize}} \quad & (x_1-4)^4+(x_2-3)^4+(x_3-2.5)^2+3|x_1+x_4+4x_5-15| \\ & +\sum_{k=1}^{K} p_k(q_{k,1}e^{y_{k,1}}+q_{k,2}y_{k,2}^4+q_{k,3}(y_{k,3}-2)^2 \\ & +q_{k,4}|y_{k,4}+q_{k,5}y_{k,5}|)\end{aligned}$$

$$\begin{aligned} \text{subject to} \quad & x_2 - \ln(x_1 + 1) - \sqrt{x_3} + x_4 + x_5^2 - 10 \leq 0, \\ & x_2 + x_1^3 + x_3^3 - 10 \leq 0, \\ & -x_4 - \sqrt{x_5} + 5 <= 0, \\ & x_i \geq 0,\ i = 1, \ldots, 5 \\ & T_k \boldsymbol{x} + W_k \boldsymbol{y_k} \leq h_k,\ k = 1, \ldots, K. \end{aligned}$$

The random parameters q, h, W and T are (using some MATLAB syntax), $q \sim |N(0,1)|$, $h \sim -0.7 \cdot |N(0,1)| - 1$, $W = -\mathrm{I}_5$, $M = 5\mathrm{x}5$ matrix with 1 to 3 zeros in each column, the rest are 1, $T = \mathrm{abs}(0.2. * \mathrm{randn}(5)). * M$. The scenarios are, again, constructed using the Monte Carlo sampling. As before, we used several methods and solvers for solving the problem:

- vanilla version of GBD (master solved by `SeDuMi`, subproblems by `fmincon`);
- EV cut version (master solved by `SeDuMi`, subproblems by `fmincon`);
- bunching + EV cut (master solved by `SeDuMi`, subproblems by `by fmincon`);
- bunching + EV cut + multicut (master solved by `SeDuMi`, subproblems by `fmincon`);
- FRP solved by `SDPT3`;
- FRP solved by `SeDuMi`.

The required precision for all the methods was set to $\epsilon = 10^{-5}$. By computations similar to that of the first example, we found the appropriate bunching size to be 5. The comparison of the different methods for varying number of scenarios is summarized in the following table:

Number of scenarios	125	250	500	1,000	2,000	3,000	5,000	7,500
Vanilla	19	45	77	164	313	–	–	–
EV cut	15	34	61	123	320	–	–	–
Multicut	19	36	134	150	309	–	–	–
Bunch 5	12	32	44	84	170	255	452	650
Bunch 5 + EV	9	17	34	71	175	210	364	578
Bunch 5 + Multicut	11	20	32	74	250	452	–	–
Bunch 5 + Multicut + EV	9	15	34	99	256	463	–	–
FRP – `SDPT3`	13	30	64	130	334	–	–	–
FRP – `SeDuMi`	1	2	6	15	40	102	355	622

Tab. 5. Computational time [s] for different methods, increasing number of scenarios.

The results demonstrate the pros and cons of using the GBD algorithm. For smaller instances, it is much more efficient to use the appropriate state-of-the-art and free solver (`SeDuMi`) to attack the full recourse formulation. However, for larger problems, the bunching variation of the GBD was able to outperform all the rest. The multicut variation suffered from a growing size of the master problem and, in this setting, cannot be considered as an improvement (a similar behavior for linear problems was shown in [13]).

7. CONCLUSION

In this paper, we introduced a novel utilization and reformulation of the traditional Generalized Benders Decomposition. To support the utility of our reformulation (as well as the utility of the GBD itself), we presented our computational experience.

From the result of the numerical examples, it is apparent that the GBD and our modifications definitely have a place as solid techniques for solving medium-sized convex two-stage stochastic problems and that especially the bunching ideas and modifications produce fruitful results.

It must be acknowledged that further investigation (i. e. a wider variety of numerical test, preferably from applications) is needed to make the arguments more conclusive. Also, further research in terms of usable lower bound as the "warm-start" cuts is anticipated.

ACKNOWLEDGEMENT

This work was supported by NETME CENTRE PLUS (LO1202) created with financial support from the Ministry of Education, Youth and Sports under National Sustainability Programme I, by the research project "Moderní matematické metody pro modelování problémů technických a přírodních věd" (FSI-S-17-4464), and by the research project "Vývoj a aplikace moderních dekompozičních metod v oblasti stochastického programování" (FSI-FV-17-04).

(Received April 12, 2017)

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty: Nonlinear Programming: Theory and Algorithms. Third edition. Wiley–Interscience, Hoboken, N. J. 2006. DOI:10.1002/0471787779

[2] J. F. Benders: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik *4* (1962), 1, 238–252. DOI:10.1007/bf01386316

[3] J. R. Birge and F. Louveaux: Introduction to Stochastic Programming. Springer, New York 2000.

[4] S. P. Boyd and L. Vandenberghe: Convex Optimization. Cambridge University Press, New York 2004. DOI:10.1017/cbo9780511804441

[5] C. A. Floudas: Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford 1995.

[6] C. A. Floudas and P. M. Pardalos: Encyclopedia of Optimization. Springer, New York 2009. DOI:10.1007/978-0-387-74759-0

[7] A. M. Geoffrion: Generalized Benders decomposition. Journal of Optimization Theory and Applications *10* (1972), 4, 237–260. DOI:10.1007/bf00934810

[8] M. Grant and S. Boyd: Graph implementations for nonsmooth convex programs. In: Recent Advances in Learning and Control (V. Blondel, S. Boyd and H. Kimura, eds.), Springer–Verlag Limited, Berlin 2008, pp. 95–110. DOI:10.1007/978-1-84800-155-8_7

[9] A. Madansky: Inequalities for stochastic linear programming problems. Management Science *6* (1960), 197–204. DOI:10.1287/mnsc.6.2.197

[10] H. D. Mittelmann: The state-of-the-art in conic optimization software. In: Handbook on Semidefinite, Conic and Polynomial Optimization (M. F. Anjos and J. B. Lasserre, eds.), Springer US, New York 2012, pp. 671–686. DOI:10.1007/978-1-4614-0769-0_23

[11] G. Pflug and F. Maggioni: Bounds and approximations for multistage stochastic programs. SIAM J. Optim. *26* (2016), 1, 831–855. DOI:10.1137/140971889

[12] R. M. Van Slyke and R. Wets: L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. SIAM J. Appl. Math. *17* (1969), 4, 638–663. DOI:10.1137/0117061

[13] C. Wolf, C. I. Fabián, A. Koberstein, and L. Suhl: Applying oracles of on-demand accuracy in two-stage stochastic programming – A computational study. Europ. J. Oper. Res. *239* (2014), 2, 437–448. DOI:10.1016/j.ejor.2014.05.010

[14] V. Zverovich, C. I. Fabián, E. F. D. Ellison, and G. Mitra: A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. Math. Program. Computation *4* (2012), 3, 211–238. DOI:10.1007/s12532-012-0038-z

Jakub Kůdela, FSI, Technická 2, Brno. Czech Republic.
e-mail: jakub.kudela89@gmail.com

Pavel Popela, FSI, Technická 2, Brno. Czech Republic.
e-mail: popela@fme.vutbr.cz

A10

KYBERNETIKA — VOLUME *54* (2018), NUMBER 6, PAGES 1201–1217

CHANCE CONSTRAINED OPTIMAL BEAM DESIGN: CONVEX REFORMULATION AND PROBABILISTIC ROBUST DESIGN

Jakub Kůdela and Pavel Popela

In this paper, we are concerned with a civil engineering application of optimization, namely the optimal design of a loaded beam. The developed optimization model includes ODE-type constraints and chance constraints. We use the finite element method (FEM) for the approximation of the ODE constraints. We derive a convex reformulation that transforms the problem into a linear one and find its analytic solution. Afterwards, we impose chance constraints on the stress and the deflection of the beam. These chance constraints are handled by a sampling method (Probabilistic Robust Design).

Keywords: optimal design, stochastic programming, chance constrained optimization, probabilistic robust design, geometric programming

Classification: 90C15, 90C30, 65C05, 49M25

1. INTRODUCTION

Optimal design problems in engineering frequently lead to optimization problems involving differential equations. One of the classes of these problems is shape optimization [11]. The particular shape optimization problem considered in this paper is the optimal design of a beam (be it a fixed beam, a cantilever beam, etc.) subjected to some kind of loading. Since shape optimization problems are inherently non-convex, most approaches use metaheuristics such as genetic algorithms [13] or cuckoo search [9]. A closely related field of topology optimization (where the size and shape of the structure can be manipulated) has developed a multitude of successful methods (level set, homogenization, topological derivative, etc.), see [19].

This problem was previously also examined in [21] and [24], where the authors used the finite element method (FEM) and the finite difference method to approximate the ordinary differential equations (ODE) and solve the problem by nonlinear programming techniques. Our paper shows that this beam design problem can be formulated as a geometric programming problem, which can be further transformed into a convex one, and thus can be efficiently solved (in comparison with the previous approaches). Geometric programming problems with random coefficients (although without chance constraints) were investigated in [8].

DOI: 10.14736/kyb-2018-6-1201

An important issue regarding the design is its reliability (see [12]). In the context of this paper the reliability of the design will mean that the constraints in the resulting optimization program should hold with high probability. Depending on how reliable design is required, we can destinguish between the so-called chance constrained (or probabilistic constrained) optimization problems (see, e. g. [20]) and the robust optimization problems (see, e. g. [3]). Current approaches dealing with reliability constrained beam design, such as [2] and [25] use simple (point) loads and Gaussian distribution of the unknown parameters. In this paper we investigate the chance constrained beam design problem under more complicated random loads. We utilize the sampling approach (called Probabilistic Robust Design) developed in [5, 6] and [7] to obtain a manageable approximation of the chance constrained problem and use a scenario-deletion method to compute a trade-off between the reliability of the design and the objective value.

2. PROBLEM FORMULATION

The problem is best described by Figure 1. We consider a fixed beam of length l with rectangular cross-section that is subjected to a load $h(x)$ (with the opposite direction than the axis y), which is depicted in Figure 1a. The task is to find the optimal design, in terms of the cross-section dimensions a and b (Figure 1b), that minimizes the weight of the beam.

Naturally, given a load $h(x)$ the beam will deflect and will be subjected to a bending stress. The requirement for the design is that the maximum stress in the beam is less then a material-specific constant, that ensures that the design is safe (we use the value at which the material begins to deform plastically). The problem can be formulated as the following ODE-constrained optimization program:

$$\underset{a,b,v(x)}{\text{minimize}} \quad \rho a b l \tag{1}$$

$$\text{subject to} \quad E\frac{ab^3}{12}\frac{\mathrm{d}^4 v}{\mathrm{d}x^4}(x) = h(x), \qquad x \in [0,l], \tag{2}$$

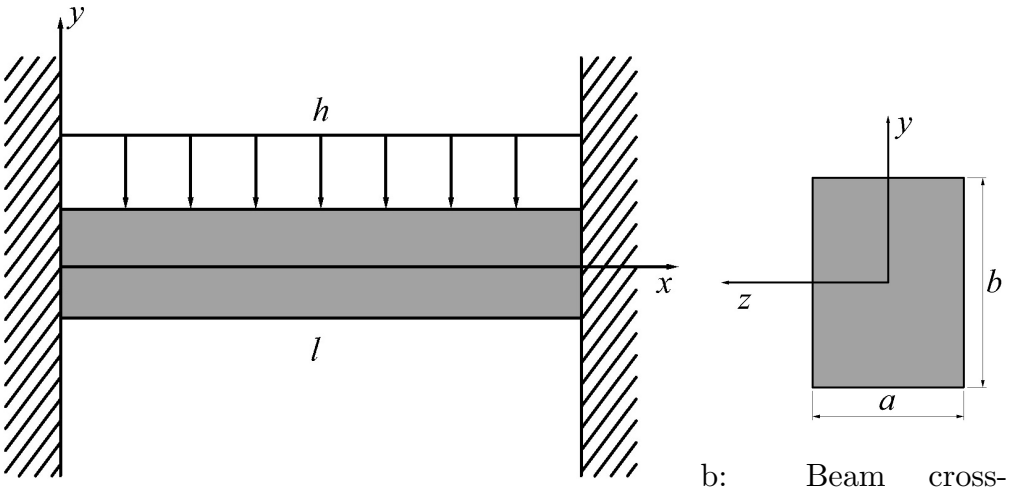
$$\left| E\frac{b}{2}\frac{\mathrm{d}^2 v}{\mathrm{d}x^2}(x) \right| \leqslant \sigma_M, \qquad x \in [0,l], \tag{3}$$

$$v(0) = 0, \;\; \frac{\mathrm{d}v}{\mathrm{d}x}(0) = 0, \;\; v(l) = 0, \;\; \frac{\mathrm{d}v}{\mathrm{d}x}(l) = 0, \tag{4}$$

$$a_L \leqslant a \leqslant a_U, \quad b_L \leqslant b \leqslant b_U, \tag{5}$$

$$\tag{6}$$

where ρ is the density of the material, $v(x)$ is the deflection of the beam (with the opposite direction than the axis y) in a point $x \in [0,l]$, E is the Young modulus, σ_M is the maximum stress allowed, and a_L, a_U, b_L, b_U are the bounds on the cross-section dimensions. The constraint (2) is the ODE that governs the deflection of the beam $v(x)$ given a specific load $h(x)$. The constraint (3) is the maximum allowed stress in the beam. The constraint (4) defines the boundary conditions for the ODE (i. e. that we have a fixed beam).



a: The scheme of loaded beam.

b: Beam cross-section.

Fig. 1. The problem geometry.

2.1. FEM problem approximation and solution

To tackle the problem (1) – (5) we use the FEM to approximate the ODE in (2) and (3). Following [22] (p. 25 – 27), we divide the one-dimensional beam with the space dimension x into N finite elements. We will denote the nodal value of the deflection $v(x)$ in the node x_e as $V_e = v(x_e)$ and the nodal value of its derivative in the same node as $\theta_e = \frac{\mathrm{d}v}{\mathrm{d}x}(x_e)$. The continuous variable v is approximated by $\tilde{v}$ in terms of nodal values as follows:

$$\tilde{v}_e = [N_1\, N_2\, N_3\, N_4][V_{e-1}\, \theta_{e-1}\, V_e\, \theta_e]^T$$

where $N_1, \ldots, N_4$ are the following cubic shape functions:

$$N_1 = \frac{1}{d^3}(d^3 - 3\mathrm{d}x^2 + 2x^3), \qquad N_2 = \frac{1}{d^2}(\mathrm{d}^2 x - 2\mathrm{d}x^2 + x^3),$$
$$N_3 = \frac{1}{\mathrm{d}^3}(3\mathrm{d}x^2 - 2x^3), \qquad N_4 = \frac{1}{\mathrm{d}^2}(x^3 - \mathrm{d}x^2),$$

and $d = \frac{l}{N}$ is the length of one element. Substitution in (2) and application of Galerkin's method leads to four element equations:

$$\int_0^d \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} E\frac{ab^3}{12}\frac{\mathrm{d}^4}{\mathrm{d}x^4}[N_1\, N_2\, N_3\, N_4]\,\mathrm{d}x \begin{bmatrix} V_{e-1} \\ \theta_{e-1} \\ V_e \\ \theta_e \end{bmatrix} = \int_0^d \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} h(x)\,\mathrm{d}x.$$

To avoid differentiating four times, the following approximation is used:

$$\int N_i \frac{\mathrm{d}^4 N_j}{\mathrm{d}x^4}\mathrm{d}x \approx -\int \frac{\mathrm{d}N_i}{\mathrm{d}x}\frac{\mathrm{d}^3 N_j}{\mathrm{d}x^3}\mathrm{d}x \approx \int \frac{\mathrm{d}^2 N_i}{\mathrm{d}x^2}\frac{\mathrm{d}^2 N_j}{\mathrm{d}x^2}\mathrm{d}x.$$

The resulting system of linear equations has the form: $E\frac{ab^3}{12}\mathbb{K}\mathcal{V} = h$, where $\mathcal{V} = (V_0, \theta_0, \ldots, V_N, \theta_N)^T$. The dimensions of the stiffness matrix $\mathbb{K}$ are $(2N+2)\times(2N+2)$ and its precise description can be found in [21] or [22]. The order of accuracy of the finite element approximation is $\mathcal{O}(d^2)$.

Using this approximation of the deflection, the stress limit (3) on each element is

given by

$$|E\frac{b}{2}[N_1'' \; N_2'' \; N_3'' \; N_4''][V_{e-1} \; \theta_{e-1} \; V_e \; \theta_e]^T| \leqslant \sigma_M.$$

This equation describing the stress in one specific node holds only for the end nodes belonging to one element (the first one at x_0 and the last one at x_N). Since the rest of the nodes belongs to two adjacent elements, the stresses are not equal. Therefore, we consider the average stress from this discontinuity:

$$E\frac{b}{2}\frac{1}{2}\left|[N_1''(0) \; N_2''(0) \; N_3''(0) \; N_4''(0)]\begin{bmatrix} V_{e-1} \\ \theta_{e-1} \\ V_e \\ \theta_e \end{bmatrix} + [N_1''(d) \; N_2''(d) \; N_3''(d) \; N_4''(d)]\begin{bmatrix} V_e \\ \theta_e \\ V_{e+1} \\ \theta_{e+1} \end{bmatrix}\right|.$$

The system of inequalities that approximates (4) can be written as $|E\frac{b}{2}\mathbb{C}\mathcal{V}| \leqslant \sigma_M$, where the matrix $\mathbb{C}$ has dimensions $(N+1)\times(2N+2)$ and its complete description can be found in [21].

The FEM approximation of the problem (1) – (5) is then the following (using the notation described above):

$$\underset{a,b,\mathcal{V}}{\text{minimize}} \quad \rho abl \tag{7}$$

$$\text{subject to} \quad E\frac{ab^3}{12}\mathbb{K}\mathcal{V} = h, \tag{8}$$

$$|E\frac{b}{2}\mathbb{C}\mathcal{V}| \leqslant \sigma_M, \tag{9}$$

$$a_L \leqslant a \leqslant a_U, \quad b_L \leqslant b \leqslant b_U. \tag{10}$$

This problem has $2N$ variables ($2N+2$ in $\mathcal{V}$ of which 4 are fixed by boundary conditions, and 2 design variables a and b), $2N+2$ constraints and a box constraints on a and b, and is non-convex, meaning that the certification of global optimality is computationally very demanding.

The crucial realization (the one that is absent in [21] and [24]) is that the stiffness matrix $\mathbb{K}$ is, by design, always invertible. In other words – given a, b and h, the equation describing the deflection of the beam has a unique solution. Using this fact, we can rewrite (8) as:

$$\mathcal{V} = \frac{12}{Eab^3}\mathbb{K}^{-1}h, \tag{11}$$

and (9) becomes:

$$|\frac{6}{ab^2}\mathbb{C}\mathbb{K}^{-1}h| = \frac{1}{ab^2}|6\mathbb{C}\mathbb{K}^{-1}h| \leqslant \sigma_M. \tag{12}$$

Let us denote as v_M the maximum of $|6\mathbb{C}\mathbb{K}^{-1}h|$ over all the nodes of the FEM discretization. Since σ_M is the same for all $N+1$ nodes, the $N+1$ inequalities (9) are equivalent to a single inequality:

$$\frac{v_M}{ab^2} \leqslant \sigma_M. \tag{13}$$

Utilizing these results and neglecting the constants ρ and l in the objective (7), we can

reformulated the problem (7) – (10) as the following equivalent problem:

$$\underset{a,b}{\text{minimize}} \quad ab \tag{14}$$

$$\text{subject to} \quad \frac{v_M}{ab^2} \leqslant \sigma_M, \quad a_L \leqslant a \leqslant a_U, \quad b_L \leqslant b \leqslant b_U, \tag{15}$$

which is a geometric program, that can be transformed into a convex program (this transformation is utilized in the following sections), with 2 variables, 1 constraint and box constraints on variables. This problem has the following analytic solution (that is derived in the Appendix A):

- if $\frac{v_M}{a_U b_U^2} > \sigma_M$, the problem is infeasible,
- if $\frac{v_M}{a_L b_L^2} \leqslant \sigma_M$, the solution is $a^* = a_L, b^* = b_L$,
- if $b = \sqrt{\frac{v_M}{a_L \sigma_M}}$ is within the bounds, $b^* = b, a^* = a_L$,
- else $a = \frac{v_M}{b_U^2 \sigma_M}$ and $a^* = a, b^* = b_U$.

This can be readily seen from the problem structure – a percentage increase in both a and b has the same result on the objective function value. However, percentage increase in b causes the left hand side of the inequality (13) to decrease faster than an equal percentage increase in a, making it preferable to increase b as much as needed (i. e. satisfying the inequality or the box constraint) before increasing a.

This result covers some of the numerical examinations done in [21] and [24] (which were more focused on the illustration of the combination of FEM and stochastic programming), without the need for using any optimization software (the only value one has to compute numerically is v_M). Another advantage is that for the same geometry (i. e. the same boundary conditions and number of elements) we can precompute the FEM matrices $\mathbb{C}$ and $\mathbb{K}$ (or its appropriate factorization, see [22], Chapter 3) and use them to quickly get optimal solution for different values of the load h.

2.2. Additional variable, constraints and convex reformulation

The structure of the problem allows us to consider the material constant E as a variable, without destroying the convexity of the upcoming reformulation. This means we can choose the quality of the material – higher E corresponding to better and more expensive one. To be able to perform the convex reformulation, the dependence of the cost on the material (per volume units) must be in the form cE^p, with $c > 0, p \in \mathbf{R}$. The objective function then becomes cE^pabl, where the constants c and l can be dropped during the optimization.

An additional restriction on the solution involves the maximum absolute deflection of the beam, which we denote as δ_M. In our FEM formulation, the vector $\mathcal{V}$ includes both the deflection of the beam and its first derivative in each node of the division. The condition on maximum deflection involves only the odd components in $\mathcal{V}$:

$$|\mathcal{V}_i| \leqslant \delta_M, i = 1, 3, 5, \ldots, 2N + 1, \tag{16}$$

which is equivalent to a single inequality

$$\max_{i=1,3,5,\dots,2N+1} |\mathcal{V}_i| \leqslant \delta_M, \tag{17}$$

using (11) and denoting the maximum of the odd components of $|12\mathbb{K}^{-1}h|$ as w_M we get

$$\frac{w_M}{Eab^3} \leqslant \delta_M. \tag{18}$$

The final constraint restricts the ratio between b and a to be less then the maximum allowed r_M.

Adding these constraints to (14) – (15), treating E as a design variable (within the bounds $0 < E_L \leqslant E_U$) and changing the objective yields the following geometric program (presented here in its standard form):

$$\begin{aligned}
\underset{a,b,E}{\text{minimize}} \quad & E^p ab && (19)\\
\text{subject to} \quad & \frac{v_M}{\sigma_M} a^{-1} b^{-2} \leqslant 1, && (20)\\
& \frac{w_M}{\delta_M} E^{-1} a^{-1} b^{-3} \leqslant 1, && (21)\\
& \frac{1}{r_M} b a^{-1} \leqslant 1, && (22)\\
& a_L a^{-1} \leqslant 1, \frac{1}{a_U} a \leqslant 1,\ \ b_L b^{-1} \leqslant 1, \frac{1}{b_U} b \leqslant 1,\ \ E_L E^{-1} \leqslant 1, \frac{1}{E_U} E \leqslant 1, && (23)
\end{aligned}$$

where all the coefficients of the monomials in (19) – (23) are clearly positive, meaning we can use the following transformation to derive an equivalent convex program. First, we transform the variables: $y_a = \log a, y_b = \log b, y_E = \log E$. Then, we can write every monomial $f(a, b, E) = ca^{\alpha_1} b^{\alpha_2} E^{\alpha_3}$, where $c > 0, \alpha_1, \alpha_2, \alpha_3 \in \mathbf{R}$ in the form

$$f(a, b, E) = f(e^{y_a}, e^{y_b}, e^{y_E}) = ce^{\alpha_1 y_a} e^{\alpha_2 y_b} e^{\alpha_3 y_E} = e^{\alpha_1 y_a + \alpha_2 y_b + \alpha_3 y_E + \log c},$$

turning a monomial function into the exponential of an affine function. Next we transform the objective and the constraints, by taking the logarithm. Since every function both in the objective and the constraints is a monomial, the transformation results in a linear program:

$$\begin{aligned}
\underset{y_a, y_b, y_E}{\text{minimize}} \quad & y_a + y_b + p\, y_E && (24)\\
\text{subject to} \quad & -y_a - 2y_b + \log v_M - \log \sigma_M \leqslant 0, && (25)\\
& -y_a - 3y_b - y_E + \log w_M - \log \delta_M \leqslant 0, && (26)\\
& -y_a + y_b - \log r_M \leqslant 0, && (27)\\
& \log a_L \leqslant y_a \leqslant \log a_U,\ \ \log b_L \leqslant y_b \leqslant \log b_U,\ \ \log E_L \leqslant y_E \leqslant \log E_U. && (28)
\end{aligned}$$

Remark 2.1. If the dependence of the material cost was $\sum_{i \in I} c_i E^{p_i}$, $c_i > 0$, instead of the simple cE^p, there would still be a convex reformulation, but it would no longer result in a linear program – there would be a term involving a logarithm of a sum of exponentials in the objective (see [4], p. 160 – 162).

3. RANDOM LOADS AND ROBUST SOLUTION

Next we investigate how the problem changes, when we introduce uncertainty. The previous papers [21] and [24] dealt with the situation, when the Young modulus E was random. In this paper, we assume that the randomness is in the load h. Instead of specifying the distribution of h by its cumulative distribution function or moment generating function (that would allow us to use the Bernstein approximation, see [16]), we devised a mechanism that produces random samples/scenarios. The use of scenarios is typical for engineering applications because of the difficulty of identifying the probability distribution. In this way, we imitate the situation when one does not know the distribution of a certain random variable, but only has access to its realizations – in our experience a much more common case. The sampling procedure is the following ($\mathcal{U}(a, b)$ denotes a

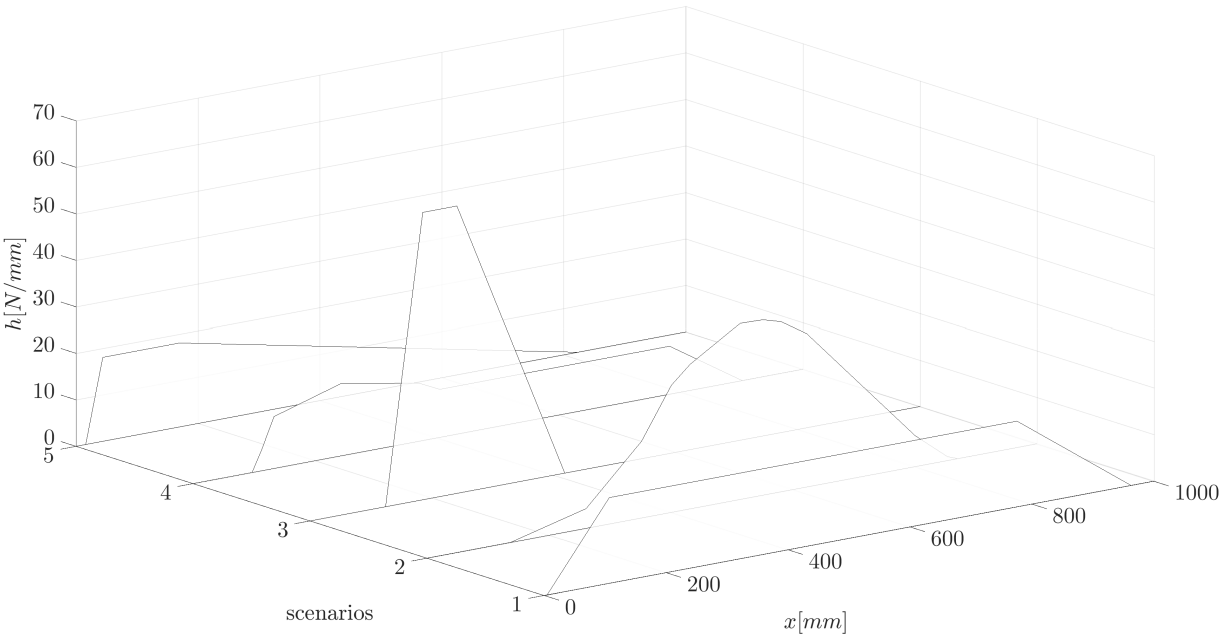


Fig. 2. A sample of 5 scenarios of the load $h(x)$.

uniform distribution):

0. Pick a random integer i between 1 and 4. Set $h(x) = 0$.
1. Repeat i times: Generate a Bernoulli trial.
   a) If 0, randomly pick 4 points $0 \leqslant x_a \leqslant x_b \leqslant x_c \leqslant x_d \leqslant l$ and add to $h(x)$ a trapezoidal load $h_a(x)$ between x_a and x_d. Height of the trapezoid is $h_M \sim \mathcal{U}(0, 1)$ (Figure 2, scenarios 1 and 3).
   b) If 1, sample $h_\mu \sim \mathcal{U}(0, l), h_\sigma \sim \mathcal{U}(0, l)$ and add to $h(x)$ the bell curve load:
$$h_b(x) = \frac{1}{h_\sigma \sqrt{2\pi}} e^{\frac{-(x-h_\mu)^2}{2h_\sigma^2}}.$$
2. Normalize the load $h(x)$: Pick $H \sim \mathcal{U}(8000\,\mathrm{N}, 15000\,\mathrm{N})$. Compute $h_i = \int_0^l h(x)\,\mathrm{d}x$, and set $h(x) = \frac{H}{h_i} h(x)$.

This sampling procedure generates very real-life like loads as can be seen in Figure 2 (see [15]). Because we transformed the original problem (1) – (5) into the problem (24) – (28), we are much more interested in the values v_M and w_M resulting from the the different load scenarios, and the actual loads $h(x)$ are of little importance. In Figure 3 we see the scatter plots and histograms of $\log v_M$ and $\log w_M$ using 2,000 scenarios of the load.

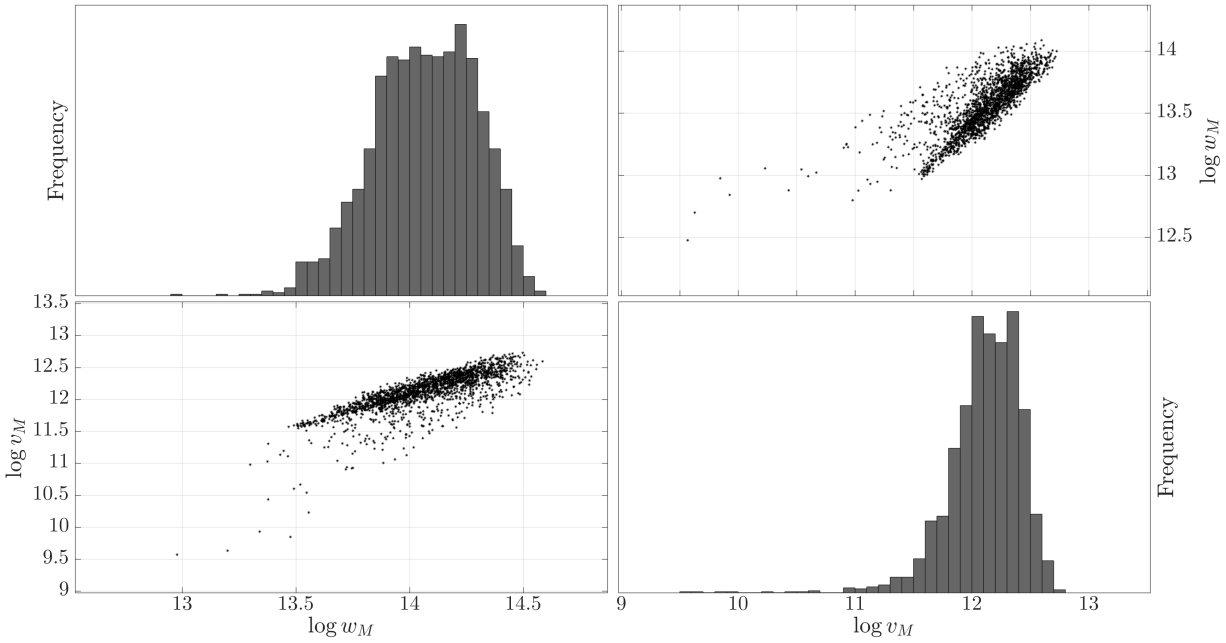


Fig. 3. Scatter plots and histograms of $\log v_M$ and $\log w_M$, 2,000 scenarios.

The important question is how to approach the optimization model (24) – (28) when some of its parameters, namely v_M and w_M, are random. One possibility is to use a so called robust formulation (see [3]), i. e. to enforce that the constraints will hold for any possible value of the random parameter. This results in the following formulation:

$$\underset{y_a, y_b, y_E}{\text{minimize}} \quad y_a + y_b + p\, y_E \tag{29}$$

$$\text{subject to} \quad -y_a - 2y_b + \log v_M(\xi) - \log \sigma_M \leqslant 0, \qquad \forall \xi \in \Xi, \tag{30}$$

$$-y_a - 3y_b - y_E + \log w_M(\xi) - \log \delta_M \leqslant 0, \qquad \forall \xi \in \Xi, \tag{31}$$

$$-y_a + y_b - \log r_M \leqslant 0, \tag{32}$$

$$\log a_L \leqslant y_a \leqslant \log a_U, \;\; \log b_L \leqslant y_b \leqslant \log b_U, \;\; \log E_L \leqslant y_E \leqslant \log E_U, \tag{33}$$

where ξ is a random outcome from a sample space Ξ. This formulation is best suited for situation, when the violation of the constraints would have disastrous consequences.

Given our scenario generation procedure, the robust formulation requires us to find the scenarios that result in the highest values of v_M and w_M, and then optimize the design with respect to these extreme values. The generation procedure allows for point loads (setting all 4 point of the trapezoid into a single point) and the magnitude of the point load is restricted to $15,000\,\mathrm{N}$ by the normalization step. This allows us to find the worst-case scenarios simply by using the formulas for the deflection and stress of a fixed

beam under point load (these can be found in [17] and [23]). The analysis of the worst case situations is carried out in the Appendix B.

4. CHANCE CONSTRAINTS AND PROBABILISTIC ROBUST DESIGN

The issue with the robust formulation is that it produces solutions that may be overly conservative. A different approach is to allow the possibility, that some of the constraints are violated, provided that the probability of violation is small. This corresponds to the following chance constrained (or probabilistic constrained, see [20]) formulation of the problem:

$$\underset{y_a,y_b,y_E}{\text{minimize}} \quad y_a + y_b + p\,y_E \tag{34}$$

$$\text{subject to} \quad P\begin{pmatrix} -y_a - 2y_b + \log v_M(\xi) - \log \sigma_M \leqslant 0, \\ -y_a - 3y_b - y_E + \log w_M(\xi) - \log \delta_M \leqslant 0 \end{pmatrix} \geqslant 1 - \epsilon, \tag{35}$$

$$-y_a + y_b - \log r_M \leqslant 0, \tag{36}$$

$$\log a_L \leqslant y_a \leqslant \log a_U, \;\; \log b_L \leqslant y_b \leqslant \log b_U, \;\; \log E_L \leqslant y_E \leqslant \log E_U, \tag{37}$$

where $1-\epsilon$ is the reliability level (or, alternatively, ϵ is the allowed violation probability). Except for some special cases, the formulation (34) – (37) is hard to solve exactly (see [20]).

One of the standard approaches (see [14]) to get an approximate solution is to fix the reliability level ϵ, draw a large number S of scenarios and construct a mixed-integer program, where for each scenario we have a binary decision variable, that corresponds to that scenario being neglected or not. One of the constraints then requires that we neglect less then ϵS scenarios. This method is clearly constrained by our ability to solve large mixed-integer programs. One of the most recent of the multiple approaches for solving the mixed-integer formulation was developed in [1].

In this paper we use a different approach based upon a method called Probabilistic Robust Design (see [5, 6] and [7]). This approach requires only that the objective is a convex functions and that the constraint functions are convex for any realization of ξ – there are no other restrictions on the position of the random variable (such as only right-hand side, linearly perturbed, etc.). The first part of the method is, again, to draw a large number S of scenarios (denoted by s) and solve the following problem:

$$\underset{y_a,y_b,y_E}{\text{minimize}} \quad y_a + y_b + p\,y_E \tag{38}$$

$$\text{subject to} \quad -y_a - 2y_b + \log v_M(s) - \log \sigma_M \leqslant 0, \qquad s = 1,\ldots,S, \tag{39}$$

$$-y_a - 3y_b - y_E + \log w_M(s) - \log \delta_M \leqslant 0, \qquad s = 1,\ldots,S, \tag{40}$$

$$-y_a + y_b - \log r_M \leqslant 0, \tag{41}$$

$$\log a_L \leqslant y_a \leqslant \log a_U, \;\; \log b_L \leqslant y_b \leqslant \log b_U, \;\; \log E_L \leqslant y_E \leqslant \log E_U, \tag{42}$$

where the $2N$ constraints (39) and (40) can be reduced to the following 2 constraints:

$$-y_a - 2y_b + \max_s(\log v_M(s)) - \log \sigma_M \leqslant 0, \tag{43}$$

$$-y_a - 3y_b - y_E + \max_s(\log w_M(s)) - \log \delta_M \leqslant 0. \tag{44}$$

For a high enough choice of S, the optimal solution to (38) – (44) yields a feasible solution for the chance constrained problem (29) – (33) with high probability (see [5]). As

investigated in [18], the approach tends to be overly conservative (i. e., the feasibility result holds, but we get a solution that is far from optimal for the chance constrained problem).

The result regarding optimality for this approach was added in [6], where the idea of discarding scenarios was developed. The main idea is, in addition to drawing S scenarios, to determine a number $k < S$, such that if we remove any k scenarios, the optimal solution of this modified problem is, again, feasible for the chance constrained problem with high probability. Furthermore, if the k scenarios are removed in an optimal fashion (i. e. we select those whose removal decreases the optimal objective value the most), there is a direct link between the optimal solution of the modified problem and the optimal solution of the chance constrained problem (in the sense that we get closer the more scenarios S we draw). Although this basically recovers the standard mixed-integer approach discussed above, there is a crucial difference in how the scenario-removal is achieved.

As discussed in [6], we can remove the k scenarios at once (the mixed-integer variant) or we can use a greedy approach that removes just one scenario at a time. In our case, the greedy approach makes perfect sense – there are only two scenarios (called support scenarios in [7]) whose removal can decrease the optimal objective value of (38) – (44):

$$s_1 = \operatorname*{argmax}_s(\log v_M(s)) \quad \text{and} \quad s_2 = \operatorname*{argmax}_s(\log w_M(s)).$$

To determine, which one of the two scenarios should be removed, we must solve two additional linear problems (with s_1 or s_2 temporarily removed) and compare their optimal objective values – this is repeated k times. The individual optimization problems have three variables and differ only in the value of one coefficient in (40) or (41) and as such can be efficiently solved by warm-starting the optimization algorithm with the last solution.

There is one different approach we will discuss, and that is the approximation of the joint chance constraint (35) by individual chance constraints:

$$P(-y_a - 2y_b + \log v_M(\xi) - \log \sigma_M \leqslant 0) \geqslant 1 - \epsilon_1, \tag{45}$$

$$P(-y_a - 3y_b - y_E + \log w_M(\xi) - \log \delta_M \leqslant 0) \geqslant 1 - \epsilon_2, \tag{46}$$

which become

$$-y_a - 2y_b + \Phi_v^{-1}(1-\epsilon_1) - \log \sigma_M \leqslant 0, \tag{47}$$

$$-y_a - 3y_b - y_E + \Phi_w^{-1}(1-\epsilon_2) - \log \delta_M \leqslant 0, \tag{48}$$

where Φ_v^{-1} and Φ_w^{-1} are the (empirical) quantile functions of $\log v_M(\xi)$ and $\log w_M(\xi)$, and ϵ_1, $\epsilon_2 > 0$ are appropriately chosen. The problem then becomes:

$$\underset{y_a, y_b, y_E}{\text{minimize}} \quad y_a + y_b + p\, y_E \tag{49}$$

$$\text{subject to} \quad -y_a - 2y_b + \Phi_v^{-1}(1-\epsilon_1) - \log \sigma_M \leqslant 0, \tag{50}$$

$$-y_a - 3y_b - y_E + \Phi_w^{-1}(1-\epsilon_2) - \log \delta_M \leqslant 0, \tag{51}$$

$$-y_a + y_b - \log r_M \leqslant 0, \tag{52}$$

$$\log a_L \leqslant y_a \leqslant \log a_U, \ \log b_L \leqslant y_b \leqslant \log b_U, \ \log E_L \leqslant y_E \leqslant \log E_U. \tag{53}$$

The choice of ϵ_1 and ϵ_2 is crucial – simply setting $\epsilon_1 = \epsilon_2 = \epsilon$ does not guarantee that the reliability of the optimal solution of (49) – (53) is better than $1 - \epsilon$ (see Figure 4). To

obtain a safe approximation of the joint chance constraint (35), ϵ_1 and ϵ_2 must satisfy (see [16]): $\epsilon_1 + \epsilon_2 \leqslant \epsilon$, the simplest values being $\epsilon_1 = \epsilon_2 = \frac{\epsilon}{2}$.

5. NUMERICAL RESULTS

Our goal is to obtain a trade-off curve between the optimal objective value (the weight of the beam) and the reliability of the design. To achieve this we used our scenarios generation technique to draw two large sets of scenarios, where the first one contained S_1 and the second S_2 scenarios. The first one was used for the optimization part (i. e. solving (38) – (44)), the second one was used for the estimate of the reliability level ϵ. The method proceeded as follows:

0. Generate the two sets of scenarios.
   Repeat k times:

1. Solve (38) – (44) using the first set of scenarios. Obtain an optimal design.

2. Estimate the reliability of the design using the second set of scenarios: given a design in the form of a, b and E, the constraints (39) – (40) either both hold, or at least one of them does not hold. This outcome describes a binomial random variable – compute its point estimate (a fraction of scenarios for which at least on of the constraints did not hold) and its 99.9% confidence interval (using the Clopper-Pearson interval).

3. Determine, which one of the two support scenarios to remove, and delete it from the first set of scenarios. Return to 1.

The problem setting under the numerical investigation was as follows: the length of the beam $l = 1\,\mathrm{m}$, number of elements for the FEM formulation $N = 1{,}000$, objective coefficient $p = \frac{1}{2}$, limits on the variables $a_L = b_L = 10^{-2}\,\mathrm{m}$, $a_U = b_U = 10^{-1}\,\mathrm{m}$, $E_L = 1.9{\cdot}10^5\,\mathrm{MPa}$ and $E_U = 2.2{\cdot}10^5\,\mathrm{MPa}$, maximum stress $\sigma_M = 120\,\mathrm{MPa}$, maximum deflection $\delta_M = 5{\cdot}10^{-4}\,\mathrm{m}$, maximum ratio between the variables $r_M = 5$, number of scenarios in the first set $S_1 = 50{,}000$, number of scenarios in the second set $S_2 = 100{,}000$, number of scenarios to discard $k = 2{,}500$.

The number of elements was chosen such that the length of one element $d = \frac{l}{N} = 10^{-3}\,\mathrm{m}$ results in accuracy $\mathcal{O}(10^{-6})$ of the FEM approximation, which is roughly of the same order as the accuracy of the optimization algorithm (termination criteria for optimality), that was set to 10^{-7}. The accuracy of the FEM was checked using the analytic results in the Appendix B and using ANSYS (commercial engineering simulation software). The FEM formulation was programmed and solved in MATLAB, the optimization parts were computed using the CVX modeling system (see [10]).

In Figure 4 is depicted the trade-off between the reliability level ϵ and the optimal objective value using the two approaches (38) – (44) and (49) – (53). In the first approach we gradually remove the scenarios (upto $k = 2{,}500$) – the computational time for each iteration (two optimization problems, scenario removal) was around 0.4 s. In the second approach (49) – (53) we vary the values of $\epsilon_1 = \epsilon_2$ between 0 and 0.05 – the computational time for each value was around 0.2 s. Furthermore, used a grid of 1,001 steps for ϵ_1 and ϵ_2 between 0 and 0.05 and computed the results for all of these grid values (they fill the

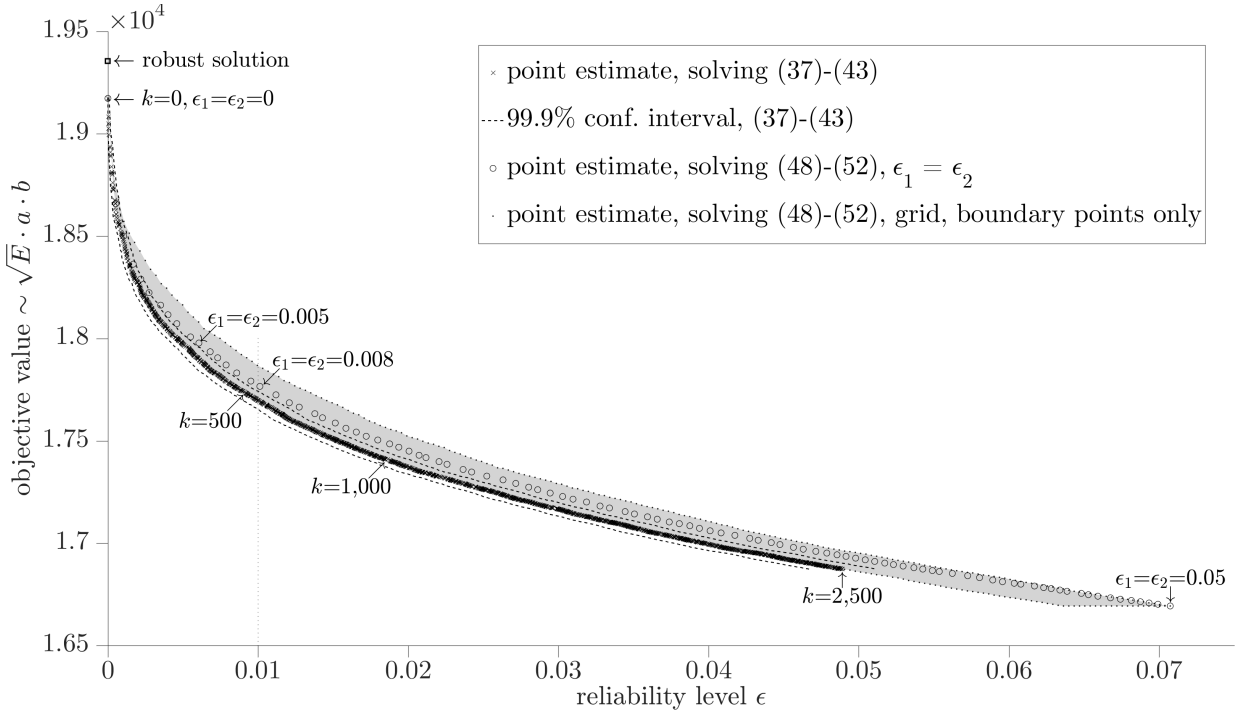


Fig. 4. The trade-off between reliability and optimal objective value.

grey area in Figure 4), this took 45 hours. The robust solution was computed using the results in the Appendix B (maximum point loads in $\frac{1}{2}l$ and $\frac{1}{3}l$).

The comparison between the two methods favours the scenario-removal one (38) – (44) over solving (49) – (53) with $\epsilon_1 = \epsilon_2$, as it produces designs with better objective value. For example, given the target (point estimate of) $\epsilon = 0.01$, the closest design produced by (49) – (53) is for $\epsilon_1 = \epsilon_2 = 0.008$, with the objective value $1.776{\cdot}10^4$, whereas the method using (38) – (44) with $k = 568$ deleted scenarios achieved the objective value $1.769{\cdot}10^4$. Moreover, the scenario-removal method (38) – (44) produced as good solutions as the best ones using the grid values for ϵ_1 and ϵ_2 and solving (49) – (53).

The shape of the trade-off heavily depends on the distribution of $h(x)$ (and, consequently, on the distribution of v_M and w_M). For the computation we used the scenario generation described earlier, which was constructed ad hoc to demonstrate the method. In a real situation (e. g., the one in [12]), the scenario generation will be swapped for the particular problem-specific outcomes.

6. CONCLUSION

In this paper, we have presented new reformulation for the optimal beam design problem, that serves as a test example for a larger set of problems solvable by similar techniques as presented. This reformulation leads to a geometric program and as such can be solved to global optimality. We then used this reformulation and extended the problem by considering randomness in the load and presented the robust and chance constrained problems. The chance constrained variant was handled by the Probabilistic Robust

Design approach. For the given scenario generation procedure we computed the trade-off between reliability and optimal objective value. Further research will be focused on situations, when the cross-section of the beam is not rectangular and the reformulation results in a possibly non-convex problem.

APPENDIX

A. The Analytic Solution

Here we derive the analytic solution for (14) – (15). We use the same convex reformulation as in (34) – (37) to derive an equivalent linear program:

$$\underset{y_a, y_b}{\text{minimize}} \quad y_a + y_b \tag{54}$$

$$\text{subject to} \quad -y_a - 2y_b + \log \frac{v_M}{\sigma_M} \leqslant 0, \tag{55}$$

$$\log a_L \leqslant y_a \leqslant \log a_U, \quad \log b_L \leqslant y_b \leqslant \log b_U. \tag{56}$$

0. a) If $\log \frac{v_M}{\sigma_M} \leqslant \log a_L + 2 \log b_L$, we are done, $a^* = a_L, b^* = b_L$.
   b) If $\log \frac{v_M}{\sigma_M} > \log a_U + 2 \log b_U$, the problem is infeasible.

1. Otherwise, we need $y_a, y_b : y_a + 2y_b = \log \frac{v_M}{\sigma_M}, \log a_L \leqslant y_a \leqslant \log a_U, \log b_L \leqslant y_b \leqslant \log b_U$.

2. The KKT conditions:
$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \nu \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \lambda_1 \begin{pmatrix} -1 \\ 0 \end{pmatrix} + \lambda_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix} + \lambda_3 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \lambda_4 \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{57}$$
$$\lambda_1(\log a_L - y_a) = 0, \lambda_2(\log b_L - y_b) = 0, \lambda_3(y_a - \log a_U) = 0, \lambda_4(y_b - \log b_U) = 0, \tag{58}$$
$$\log a_L \leqslant y_a \leqslant \log a_U, \log b_L \leqslant y_b \leqslant \log b_U, y_a + 2y_b = \log \frac{v_M}{\sigma_M}, \lambda_i \geqslant 0, i = 1, \ldots, 4. \tag{59}$$

3. From complementary slackness condition (58) we get 16 different possible situations - corresponding to a or b being at the specific bounds. From the outset it is clear that the variables cannot be at the lower and upper bound at the same time: λ_1 and λ_3 cannot be both nonzero, the same holds for λ_2 and λ_4. This rules out 7 possibilities.

4. If $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 = 0$, from (57) we have
$$\nu = -1, \quad \text{from the first row,} \qquad \nu = -\frac{1}{2}, \quad \text{from the second row,}$$
which is not possible. This means that there cannot be an optimal solution such that $a_L < a^* < a_U$ and $b_L < b^* < b_U$ at the same.

5. If $\lambda_1 > 0, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 = 0$, i. e. $a^* = a_L, y_a^* = \log a_L$. From (57) we have
$$\nu = -\frac{1}{2}, \quad \lambda_1 = \frac{1}{2} > 0,$$

meaning that $y_b^* = \frac{1}{2}\log\frac{v_M}{a_L\sigma_M}$ and $b^* = e^{y_b^*} = \sqrt{\frac{v_M}{a_L\sigma_M}}$ is a possible solution, provided $b_L < b^* < b_U$.

6. If $\lambda_1 > 0, \lambda_2 > 0, \lambda_3 = 0, \lambda_4 = 0$, i. e. $a^* = a_L, b^* = b_L$. This is the situation in 0. a).

7. If $\lambda_1 > 0, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 > 0$, i. e. $a^* = a_L, b^* = b_U$. From (57) we have
$$\lambda_1 = 1 + \nu > 0 \Rightarrow \nu > -1, \quad \lambda_4 = -1 - 2\nu > 0 \Rightarrow \nu < -\frac{1}{2}, \quad \text{which is possible.}$$
This is the (arguably rare) situation when $\log a_L + 2\log b_U = \log\frac{v_M}{\sigma_M}$.

8. If $\lambda_1 = 0, \lambda_2 > 0, \lambda_3 = 0, \lambda_4 = 0$, i. e. $a^* = a_U$. From (57) we have
$$\nu = -1, \quad \lambda_2 = -1 > 0, \quad \text{which is not possible.}$$

9. If $\lambda_1 = 0, \lambda_2 > 0, \lambda_3 > 0, \lambda_4 = 0$, i. e. $a^* = a_U, b^* = b_L$ from (57) we have
$$\lambda_3 = -1 - \nu > 0 \Rightarrow \nu < -1, \quad \lambda_2 = 1 + 2\nu > 0 \Rightarrow \nu > -\frac{1}{2}, \quad \text{which is not possible.}$$

10. If $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 > 0, \lambda_4 = 0$, i. e. $b^* = b_L$. From (57) we have
$$\nu = -\frac{1}{2}, \quad \lambda_3 = -\frac{1}{2} > 0, \quad \text{which is not possible.}$$

11. If $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 > 0, \lambda_4 > 0$, i. e. $a^* = a_U, b^* = b_U$. From (57) we have
$$\lambda_3 = -1 - \nu > 0 \Rightarrow \nu < -1, \quad \lambda_4 = -1 - 2\nu > 0 \Rightarrow \nu < -\frac{1}{2}, \quad \text{which is possible.}$$
This is the situation when $\log a_U + 2\log b_U = \log\frac{v_M}{\sigma_M}$.

12. If $\lambda_1 = 0, \lambda_2 = 0, \lambda_3 = 0, \lambda_4 > 0$, i. e. $b^* = b_U$. From (57) we have
$$\nu = -1, \quad \lambda_4 = 1 > 0, \quad \text{which is possible, } y_a^* = \log\frac{v_M}{b_U^2\sigma_M}, a^* = \frac{v_M}{b_U^2\sigma_M}.$$

B. Worst case deflection and stress for point load

The following results are using the known formulas for deflection and bending moment for fixed beam under a point load that can be found in [17] and [23]. The Figure 5 depicts the situation and provides a graphical description of the used notation.

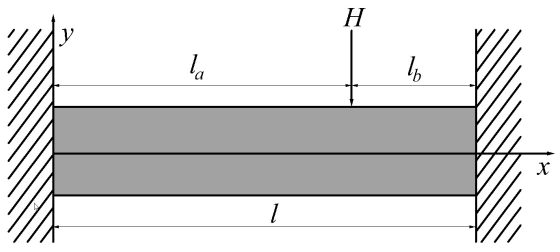


Fig. 5. Point load.

The maximum deflection of a fixed beam under point load is computed by the following formula (can be found in [23], p. 190):

$$\delta_M = \frac{2Hl_a^3 l_b^2}{3EI(3l_a + l_b)^2}, \tag{60}$$

where l_a and l_b correspond to the location of the point load ($l_a + l_b = l$), I is the moment of inertia of the cross-section and E is the Young modulus. In our case $I = \frac{ab^3}{12}$. If we look at (60) as a function of the location l_a of the point load, its maximum occurs when $l_a = l_b = \frac{l}{2}l$.

The maximum stress for each point $x \in [0, l]$ in the beam can be expressed in the following terms: $\sigma_M(x) = \frac{M(x)}{I} y_M$, where $M(x)$ is the bending moment and $y_M = \pm\frac{b}{2}$. This allows us to use the formulas for maximum bending moment of fixed beam under point load to find the critical points (the signs in the formulas are neglected, since the constraint (3) restricts the absolute value of the stress). The bending moment of a beam under point load changes linearly between the points $0, l_a$, and l, so it suffices to compute the bending moment in these three points. Given a point load at $x = l_a$ bending moment at the ends of the beam ($x = 0$ and $x = l$) is

$$\text{left end: } M(x = 0) = \frac{Hl_a l_b^2}{l^2}, \quad \text{right end: } M(x = l) = \frac{Hl_a^2 l_b}{l^2},$$

the maximum occurs when $l_a = \frac{1}{3}l$ (or $l_a = \frac{2}{3}l$) resulting in $M(x = 0 \text{ or } x = l) = \frac{4}{27}lH$.

The moment at the location of the point is $M(x = l_a) = \frac{2Hl_a^2 l_b^2}{l^3}$, for which the maximum occurs when $l_a = l_b = \frac{1}{2}l$, resulting in $M(x = \frac{1}{2}l) = \frac{1}{8}lH$. This means that worst case occurs, when the point load is located in $l_a = \frac{1}{3}l$ or $l_a = \frac{2}{3}l$.

ACKNOWLEDGEMENT

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project LTC18053 and by the project "Computer Simulations for Effective Low-Emission Energy" funded as project No. CZ.02.1.01/0.0/0.0/16_026/0008392 by Operational Programme Research, Development and Education, Priority axis 1: Strengthening capacity for high-quality research.

Comments by the anonymous reviewers have significantly improved the paper and are also greatly acknowledged.

(Received January 3, 2018)

REFERENCES

[1] L. Adam and M. Branda: Nonlinear chance constrained problems: Optimality conditions, regularization and solvers. J. Optim. Theory Appl. *170* (2016), 2, 419–436. DOI:10.1007/s10957-016-0943-9

[2] A. T. Beck, W. J. S. Gomes , R. H. Lopez, and L. F. F. Miguel: A comparison between robust and risk-based optimization under uncertainty. Struct. Multidisciplin. Optim. *52* (2015), 3, 479–492. DOI:10.1007/s00158-015-1253-9

[3] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski: Robust Optimization. Princeton University Press, 2009. DOI:10.1515/9781400831050

[4] S. P. Boyd and L. Vandenberghe: Convex Optimization. Cambridge University Press, New York 2004. DOI:10.1017/cbo9780511804441

[5] G. C. Calafiore and M. C. Campi: The Scenario approach to robust control design. IEEE Trans. Automat. Control *51* (2006), 5, 742–753. DOI:10.1109/tac.2006.875041

[6] M. C. Campi and S. Garatti: A Sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. J. Optim. Theory Appl. *148* (2011), 257–280. DOI:10.1007/s10957-010-9754-6

[7] A. Carè, S. Garatti, and M. C. Campi: Scenario min-max optimization and the risk of empirical costs. SIAM J. Optim. *25* (2015), 4, 2061–2080. DOI:10.1137/130928546

[8] J. Dupačová: Stochastic geometric programming with an application. Kybernetika *46* (2010), 3, 374–386.

[9] A. H. Gandomi, X.-S. Yang, and A. H. Alavi: Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. Engrg. Comput. *29* (2013), 1, 17–35. DOI:10.1007/s00366-011-0241-y

[10] M. Grant and S. Boyd: Graph implementations for nonsmooth convex programs. In: Recent Advances in Learning and Control (V. Blondel, S. Boyd and H. Kimura, eds.), Springer–Verlag Limited, Berlin 2008, pp. 95–110. DOI:10.1007/978-1-84800-155-8_7

[11] J. Haslinger and R. A. E. Mäkinen: Introduction to Shape Optimization: Theory, Approximation, and Computation (Advances in Design and Control). SIAM, 2003. DOI:10.1137/1.9780898718690

[12] I. Laníková, P. Štěpánek, and P. Šimůnek: Optimized Design of concrete structures considering environmental aspects. Advances Structural Engrg. *17* (2014), 4, 495–511. DOI:10.1260/1369-4332.17.4.495

[13] M. Lepš and M. Šejnoha: New approach to optimization of reinforced concrete beams. Computers Structures *81* (2003), 1, 1957–1966. DOI:10.1016/s0045-7949(03)00215-3

[14] J. Luedtke, S. Ahmed, and G. L. Nemhauser: An integer programming approach for linear programs with probabilistic constraints. Math. Programm. Ser. A *122* (2010), 247–272. DOI:10.1007/s10107-008-0247-4

[15] P. Marek, J. Brozzetti, and M. Gustar: Probabilistic Assessment of Structures using Monte Carlo Simulation. TeReCo, Praha 2001. DOI:10.1115/1.1451167

[16] A. Nemirovski: On safe tractable approximations of chance constraints. Europ. J. Oper. Res. *219* (2012), 707–718. DOI:10.1016/j.ejor.2011.11.006

[17] E. Oberg, F. D. Jones, and H. H. Ryffel: Machinery's Handbook Guide. 29th edition. Industrial Press, 2012.

[18] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro: Sample average approximation method for chance constrained programming: Theory and applications. J. Optim. Theory Appl. *142* (2009), 399–416. DOI:10.1007/s10957-009-9523-6

[19] G. I. N. Rozvany and T. Lewiński (eds.): CISM Courses and Lectures: Topology Optimization in Structural and Continuum Mechanics. Springer-Verlag, Wien 2014.

[20] A. Ruszczynski and A. Shapiro (eds.): Handbooks in Operations Research and Management Science: Stochastic Programming. Elsevier, Amsterdam 2003.

[21] Z. Šabartová and P. Popela: Beam design optimization model with FEM based constraints. Mendel J. Ser. *1* (2012), 422–427.

[22] I. M. Smith and D. V. Griffiths: Programming the Finite Element Method. Fourth edition. John Wiley and Sons, New York 2004.

[23] W. C. Young, R. G. Budynas and A. M. Sadegh: Roark's Formulas for Stress and Strain. Seventh edition. McGraw-Hill Education, 2002.

[24] E. Žampachová, P. Popela, and M. Mrázek: Optimum beam design via stochastic programming. Kybernetika *46* (2010), 3, 571–582.

[25] X. Zhuang and R. Pan: A sequential sampling strategy to improve reliability-based design optimization with implicit constraint functions. J. Mechan. Design *134* (2012), 2, Article number 021002. DOI:10.1115/1.4005597

Jakub Kůdela, FSI, Technická 2, Brno. Czech Republic.
e-mail: Jakub.Kudela@vutbr.cz

Pavel Popela, FSI, Technická 2, Brno. Czech Republic.
e-mail: popela@fme.vutbr.cz

A11

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Pool & Discard Algorithm for Chance Constrained Optimization Problems

JAKUB KŮDELA[1], PAVEL POPELA[2]

[1]Institute of Computer Science and Automation, Faculty of Mechanical Engineering, Brno University of Technology, Technicka 2, Brno, Czech Republic (e-mail: Jakub.Kudela@vutbr.cz)

[2]Institute of Mathematics, Faculty of Mechanical Engineering, Brno University of Technology, Technicka 2, Brno, Czech Republic (e-mail: popela@fme.vutbr.cz)

Corresponding author: Jakub Kůdela (e-mail: Jakub.Kudela@vutbr.cz).

The financial support for this work was provided by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project LTC18053, and by the project "Computer Simulations for Effective Low-Emission Energy" funded as project No. CZ.02.1.01/0.0/0.0/16 026/0008392 by Operational Programme Research, Development and Education, Priority axis 1: Strengthening capacity for high-quality research.

ABSTRACT In this paper, we describe an effective algorithm for handling chance constrained optimization problems, called the Pool & Discard algorithm. The algorithm utilizes the scenario approximation framework for chance constrained optimization problems, and the warm-start and problem modification features of modern solvers. The exploitation of the problem structure and efficient implementation allows us to considerably speed up the computations, especially for large instances, when compared with conventional methods.

INDEX TERMS chance constrained programming, scenario approximation, P&D algorithm, stochastic programming, constraint removal

I. INTRODUCTION

This article describes a novel method for handling chance constrained optimization problems that was developed in the author's dissertation [1]. The introduction into the topic of chance constrained optimization is derived (more or less directly) from [2] – with most of the used notation adapted from [2] as well. Let $\mathcal{X} \subseteq \Re^{n_x}$ be a convex and closed domain of optimization and consider a family of constraints $x \in \mathcal{X}_\xi$ parameterized in $\xi \in \Xi$. The uncertain parameter ξ describes different instances of an uncertain optimization scenario. We adopt a probabilistic description of uncertainty and suppose that the support Ξ for ξ is endowed with a σ-algebra $\mathcal{D}$ and that a probability measure $\mathcal{P}$ is defined over $\mathcal{D}$. The probability measure $\mathcal{P}$ describes the probability with which the uncertain parameter ξ takes value in Ξ. Then, a chance constrained optimization program is written as:

$$\text{CCP}_\epsilon: \quad \begin{aligned} &\underset{x\in\mathcal{X}}{\text{minimize}} && c^T x \\ &\text{subject to} && \mathcal{P}\{\xi : x \in \mathcal{X}_\xi\} \geq 1-\epsilon. \end{aligned} \tag{1}$$

Here, we assume that the σ-algebra $\mathcal{D}$ is large enough, so that $\{\xi : x \in \mathcal{X}_\xi\} \in \mathcal{D}$, i.e. $\{\xi : x \in \mathcal{X}_\xi\}$ is a measurable set. Also, linearity of the objective function can be assumed without loss of generality, since any objective of the form

$$\underset{x\in\mathcal{X}}{\text{minimize}} \quad c(x),$$

where $c(x) : \mathcal{X} \to \Re$ is a convex function, can be re-written as

$$\underset{x\in\mathcal{X}, y\geq c(x)}{\text{minimize}} \quad y,$$

where y is a scalar variable.

In the CCP_ϵ (1), constraint violation is tolerated, but the violated constraint set must be no larger than ϵ. The parameter ϵ allows us to trade robustness (in terms of the probability of constraint violation) for performance (in terms of the optimal objective value): the optimal objective value J^*_ϵ of CCP_ϵ is a decreasing function of ϵ and provides a quantification of such a trade-off. Depending on the particular application (the range of applications is quite wide), ϵ can take different values and has not necessarily to be thought of as a "small" parameter.

Chance constrained programming has been around for more than half a century, at least since the work of Charnes, Cooper and Symonds in the fifties, see [3]. In [3], however, only individual chance constraints

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

were considered. Joint probabilistic constraints, as in (1), were first considered by Miller and Wagner, [4], in an independent context, while a general theory is due to Prékopa, see [5], [6]. Prékopa was also the one to introduce the convexity theory based on logconcavity, which was a fundamental step toward solvability of a large class of chance constrained problems. The books [7] and [8] provide an excellent and broad overview on logconcavity theory in stochastic programming, and related results. Yet another study about the convexity of chance constrained problems is [9], while convex approximations of chance constrained problems are considered in [10], [11], and [12]. Stability of the solution under perturbation of the chance constrained problem is studied in [13] and [14]. Although chance constrained problems can be efficiently solved in some special cases, it remains true that the feasible set of CCP_ϵ is in general non-convex in spite of the convexity of the sets $\mathcal{X}_\xi$. Therefore, an exact numerical solution of CCP_ϵ is, at least in general, extremely hard to find.

II. SAMPLE COUNTERPART APPROACH

We can view the variable $x \in \mathcal{X} \subseteq \Re^{n_x}$ as the "design variable". The family of possible instances is parameterized by an "uncertainty vector" $\xi \in \Xi \subseteq \Re^{n_\xi}$. Then, the prototype optimization problem consists in minimizing a linear objective $c^T x$, subject to that x satisfies the constraints $g(x,\xi) \leq 0, \forall \xi \in \Xi$, where $g(x,\xi) : \mathcal{X} \times \Xi \to [-\infty, \infty]$ is a scalar-valued function that specifies the constraints. Note that considering scalar-valued constraint functions can be assumed without loss of generality, since multiple constraints $g_1(x,\xi) \leq 0, \ldots, g_m(x,\xi) \leq 0$ can be expressed by a single scalar-valued constraint by the position $g(x,\xi) = \max_{i=1,\ldots,m} g_i(x,\xi)$. Although convexity is preserved by this operation, other valuable properties, such as linearity or differentiability, are lost. In typical situations, Ξ has infinite cardinality, i.e., it contains an infinite number of possible instances for ξ.

Assumption 2.1 (Convexity):
For each $\xi \in \Xi$ the sets $\mathcal{X}_\xi$ are convex and closed.

Assumption 2.1 requires convexity only with respect to the design variable x, while generic nonlinear dependence with respect to ξ is allowed.

Depending on the situation at hand, the measure $\mathcal{P}$ can have different interpretations. On one hand, it can be the actual probability with which the uncertainty parameter ξ takes on value in Ξ. On the other hand, $\mathcal{P}$ can simply describe the relative importance we assign to different uncertainty instances. We have the following definition:

Definition 2.2 (Probability of Violation):
Let $x \in \mathcal{X}$ be given. The probability of violation of x is defined as

$$\mathcal{V}(x) = \mathcal{P}\{\xi \in \Xi : g(x,\xi) > 0\}.$$

For example, if we assume a uniform probability density, then $\mathcal{V}(x)$ measures the "volume of bad" parameters ξ such that the constraint $g(x,\xi) \leq 0$ is violated. A solution x with small associated $\mathcal{V}(x)$ is feasible for most of the problem instances, i.e., it is approximately feasible for the worst-case problem. This concept of approximate feasibility has been introduced in the context of robust control in [15]. Any such solution is here named an "ϵ-level" solution:

Definition 2.3 (ϵ-Level Solution):
Let $\epsilon \in (0,1)$. We say that $x \in \mathcal{X}$ is an ϵ-level robustly feasible (or, more simply, an ϵ-level) solution, if $\mathcal{V}(x) \leq \epsilon$.

Our ultimate goal is to devise an algorithm that returns a ϵ-level solution, where ϵ is any fixed small reliability level. The approach utilized in this paper uses a surrogate model called "Scenario Design Problem". By scenario it is here meant any possible realization or instance of the uncertainty parameter. In the "scenario design" we optimize the objective subject to a finite number of these randomly selected scenarios.

Definition 2.4 (Scenario Design Problem):
Assume that S independent identically distributed samples $\xi^1, \ldots, \xi^S$ are drawn according to probability $\mathcal{P}$. A scenario design problem is given by the convex program

$$\mathrm{SDP}_S: \quad \begin{aligned} &\underset{x \in \mathcal{X}}{\text{minimize}} && c^T x \\ &\text{subject to} && g(x,\xi^i) \leq 0, \quad i = 1, \ldots, S. \end{aligned} \tag{2}$$

The acronym SDP_S refers to the fact that (2) is a convex program with S constraints. Here we assume the following technical condition on the scenario problem:

Assumption 2.5 (Feasibility):
For all possible extractions $\xi^1, \ldots, \xi^S$, the optimization problem (2) is either infeasible, or, if feasible, it attains a unique optimal solution.

The scenario problem SDP_S is a standard convex optimization problem with a finite number of constraints S and, hence, its optimal solution $\hat{x}_S$ is (usually) efficiently computable by means of numerical algorithms [16].

The relationship between the number of sampled scenarios S and the probability of violation of the optimal solution to corresponding Scenario Design Problem $\mathcal{V}(\hat{x}_S)$ was investigated in [17] and [18] – for a chosen ϵ we can always find S large enough such that $\hat{x}_S$ is ϵ-level feasible for the original problem (1) with arbitrarily high confidence. There is, however, no guarantee, that the resulting optimal objective value of (2) will be anywhere close to the true optimal value J^*_ϵ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

The results derived in [18] show that the distribution function of $\mathcal{V}(\hat{x}_S)$ is bounded by a beta distribution with parameters n_x and $S - n_x + 1$, and imposing that $\mathcal{V}(\hat{x}_S) \leq \epsilon$ holds with high confidence implies that $\mathcal{V}(\hat{x}_S)$ will be much less than ϵ in many cases, resulting in a conservative solution.

Next we introduce a concept that is crucial for the success of the Pooling part of the upcoming algorithm. Of the S generated scenarios, only some of these S will be "bounding" in the sense that they prevent the solution from "falling" to a lower objective value.

Definition 2.6 (Support Scenario):
Scenario $\xi^i, i \in \{1, \ldots, S\}$, is a support scenario for the scenario problem SDP_S if its removal changes the optimal solution of SDP_S.

The following theorem, whose proof can be found in [18] or in a different form in [19], gives us the bound on the number of support scenarios:

Theorem 2.7 (Number of Support Scenarios):
The number of support scenarios for SDP_S is at most n_x, the size of x.

What is most important about this result is the fact that the number of support scenarios does not depend on the number of generated scenarios S. The first main contribution of this paper is an efficient way of solving (2), with the use of Theorem 2.7.

III. POOLING PART OF THE POOL & DISCARD ALGORITHM

The idea behind the Pooling part of the algorithm is the following: if one were to verbally describe the problem (2), the one word that came to our mind was "long", as there are much more constraints than decision variables. Moreover, the number of support constraints (or support scenarios), that the optimal solution of (2) depends upon is very small, when compared to the overall number of constraints (or scenarios).

The method consists of solving (2) by the following procedure. First, we start by completely neglecting the constraints in (2) that correspond to the different scenarios and solve this relaxed optimization problem. Then we find the most violated constraints (by computing the slacks), add them to the relaxed problem and find a new optimal solution.

The Pooling part can be summarized as follows:

Step 0. Set $\mathcal{I} = \emptyset$.

Step 1. Solve the following problem:

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & g(x, \xi^i) \leq 0, \quad i \in \mathcal{I}, \end{aligned} \tag{3}$$

and obtain a solution $\hat{x}$.

Step 2. Check feasibility of the solution by computing the slacks s^i:

$$s^i = g(\hat{x}, \xi^i), \quad i \in \{1, \ldots, S\}. \tag{4}$$

Step 3. If $\max\limits_{i \in \{1,\ldots,S\}} s^i > 0$, find the associated index of the maximum value $\hat{i} = \operatorname{argmax}\limits_{i \in \{1,\ldots,S\}} s^i$, add it to the set $\mathcal{I}$ and return to Step 1. Otherwise, set $x^* = \hat{x}, \mathcal{I}^* = \mathcal{I}$ and terminate.

It is important to remark that by the end of this procedure, we not only get the optimal solution of (2), but also an index set $\mathcal{I}$ that contains the support scenarios – this will be very significant for the success of the Discarding part of the P&D algorithm.

Another equally important remark concerns the efficient implementation of the algorithm. In Step 1, we are sequentially solving problems that are extremely similar, only differing in a single constraint. The use of warm-starts (when made possible by a proper choice of solution method) or even problem modification[1] (when supported by our choice of a solver) have immense effect on the efficiency of the Pooling part. For the implementation of the numerical examples that are investigated in this paper, we have chosen the JuMP package [20] for modeling optimization in the Julia language [21] and the CPLEX 12.7 solver [22]. This combination allowed us to use the algorithm to its full extent[2]. The machine, on which we conducted the numerical examples, was a PC with 3.6 GHz AMD Ryzen 5 2600X Six-Core CPU, 32 GB RAM, NVIDIA GeForce GTX 1050 Ti, running on 64-bit Windows 10.

A. NUMERICAL EXAMINATION – ASSET ALLOCATION PROBLEM

The first numerical example we chose to demonstrate the utility of the Pooling part of the P&D algorithm is the (by now, almost canonical) asset allocation problem [8]. Suppose we have n assets $x_1, \ldots, x_n$ that we want to invest in. The returns $r_1, \ldots, r_n$ of these assets are random variables. Our goal is to allocate our resources to these different assets, in order to maximize the ϵ quantile (often called the Value at Risk, or VaR) of the returns. This formulation neglects several of the important real-world issues – we do not allow short position, do not consider more than one trading period, etc. – the example is, above all else, intended to show the capabilities of the P&D algorithm. Our asset allocation problem can be summarized as follows:

$$\begin{aligned} \underset{x \geq 0, t \in \Re}{\text{maximize}} \quad & t \\ \text{subject to} \quad & \mathcal{P}\{t \leq \textstyle\sum_{j=1}^n r_j x_j\} \geq 1 - \epsilon, \\ & \textstyle\sum_{i=j}^n x_j \leq 1. \end{aligned} \tag{5}$$

Our ability to solve (with no quotation marks) this problem depends heavily on the distribution of the

[1] https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.0/ilog.odms.cplex.help/CPLEX/OverviewAPIs/topics/Modify.html

[2] The implementation of all of the presented numerical examples can be found on the authors GitHub: https://github.com/JakubKudela89

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

returns $r_1, \ldots, r_n$ and the chosen quantile ϵ. Thanks to [23], we know that the feasible set of a scalar chance constraint

$$\mathcal{P}\{a^T x \leq b\} \geq 1 - \epsilon,$$

is convex, provided that the vector $(a^T, b)^T$ of the coefficients has symmetric logarithmically concave density and $\epsilon < 1/2$. We will use this result and model the returns r as random variables that are independent and normally distributed (and, hence, have a symmetric logarithmically concave density). More precisely, the return r_j has the following distribution

$$r_j \sim \mathcal{N}(\mu_j, \sigma_j), \quad \mu_j = 1 + 0.1\frac{j-1}{n-1}, \quad \sigma_j = 0.1\frac{j-1}{n-1},$$

i.e., the first return is "deterministic", with return $r_1 = 1$, and the nth return has mean $\mu_n = 1.1$ and standard deviation $\sigma_n = 0.1$. Because of the chosen distribution of returns, the problem (5) can be transformed [8] into the following second order cone problem (SOCP, see [16]):

$$\begin{array}{ll} \underset{x\geq 0, t\in\Re}{\text{maximize}} & t \\ \text{subject to} & \sum_{j=1}^n \mu_j \cdot x_j \geq t + \\ & \quad \Phi^{-1}(1-\epsilon)\cdot||(\sigma_1 \cdot x_1, \ldots, \sigma_n \cdot x_n)||_2, \\ & \sum_{j=1}^n x_i \leq 1, \end{array} \tag{6}$$

where $\Phi^{-1}(1-\epsilon)$ is the $1-\epsilon$ quantile of the standard normal distribution. As an SOCP, this problem falls into the category of "easy" to solve (we can compute the optimal solution with little effort for large values of n – well into thousands) and as such provides the perfect ground for illustrating the capacities of the P&D algorithm.

The scenario approach, works with a sample of S scenarios of the returns $r_j^i, j = 1, \ldots, n, i = 1, \ldots, S$. Using these scenarios, the sample counterpart to (5) has the following form:

$$\begin{array}{ll} \underset{x\geq 0, t\in\Re}{\text{maximize}} & t \\ \text{subject to} & t \leq \sum_{j=1}^n r_j^i x_j, \quad i \in \{1, \ldots, S\} \\ & \sum_{j=1}^n x_j \leq 1. \end{array} \tag{7}$$

First of all, we will investigate on (7) the dependence of computation time of the Pooling part (CTPP) of the P&D algorithm for varying number of assets n and scenarios S. Additionally, we provide the computation time for the Pooling part without the use of warm-starts and problem modification (CnoWS) and the computation time for solving the problem (7) conventionally (CTC), i.e., passing it to the solver (CPLEX) with all the scenarios.

The results of the computations are summarized in Table 1 and clearly demonstrate the effectiveness of the Pooling part of the P&D algorithm. As the number of scenarios grows, CTPP grows very slowly when compared to CTC, becoming over 20 times faster for the largest number of considered scenarios. The main factor in the effectiveness of the Pooling part is the low growth in the number of iterations needed to solve the problems with more scenarios – this should not be too surprising, since the number of support scenarios stays the same (for the same n). The variant without warm-start or problem modification CnoWS eventually (for high values of n) suffers from too big of an overhead when constructing the corresponding optimization problem, but can still outperform CTC in large number of instances.

IV. CONSTRAINT REMOVAL ALGORITHM

If all the S constraints are enforced, however, one cannot expect that good approximations of chance constrained solutions are obtained. To get a less conservative solution we use the framework introduced in [2] for relaxing problem (7). Their approach allows us to remove k constraints out of the S scenario constraints. A general removal procedure is formalized in the following definition:

Definition 4.1 (Constraint Removal Algorithm):
Let $k < S$. An algorithm $\mathcal{A}$ for constraints removal is any rule by which k constraints out of a set of S constraints are selected and removed. The output of $\mathcal{A}$ is the set $\mathcal{A}\{\xi^1, \ldots, \xi^S\} = \{i_1, \ldots, i_k\}$ of the indexes of the k removed constraints.

The sample-based optimization program where k constraints are removed as indicated by $\mathcal{A}$ is expressed as

$$\begin{array}{lll} \text{SDP}_{S,k}^{\mathcal{A}}: & \underset{x\in\mathcal{X}}{\text{minimize}} & c^T x \\ & \text{subject to} & g(x, \xi^i) \leq 0, \\ & & i \in \{1, \ldots, S\} \setminus \mathcal{A}\{\xi^1, \ldots, \xi^S\}, \end{array} \tag{8}$$

and its solution will be hereafter indicated as $x_{S,k}^*$. We introduce the following assumptions:

Assumption 4.2 (Constraint Violation):
Almost surely with respect to the multi-sample $(\xi^1, \ldots, \xi^S)$, the solution $x_{S,k}^*$ of the sample-based optimization program $\text{SDP}_{S,k}^{\mathcal{A}}$ violates all the k constraints that $\mathcal{A}$ has removed.

This assumption requires that the algorithm $\mathcal{A}$ chooses constraints whose removal improves the solution by violating the removed constraints, and it rules out for example algorithms that remove inactive constraints only, or algorithms that remove constraints at random. Thus, this assumption is very natural and reflects the fact that we want to remove the constraints that improve the optimal objective value.

The next Theorem (proved in [2]) provides theoretical guarantees that $\mathcal{V}(x_{S,k}^*) \leq \epsilon$, i.e. that the optimal solution $x_{S,k}^*$ of the optimization program $\text{SDP}_{S,k}^{\mathcal{A}}$ is feasible for the CCP_ϵ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

TABLE 1: Results of the computation. Average over 10 runs.

n	S	CTC [s]	CnoWS [s]	CTPP [s]	iterations
10	100	0.002	0.008	0.002	12.7
	1,000	0.012	0.013	0.004	18.9
	5,000	0.062	0.011	0.005	17.4
	10,000	0.125	0.014	0.005	15.3
	20,000	0.301	0.013	0.005	13.5
	50,000	0.920	0.024	0.020	11.9
	100,000	1.897	0.033	0.044	10.7
20	100	0.004	0.013	0.005	17.3
	1,000	0.021	0.030	0.008	29.3
	5,000	0.105	0.044	0.011	37.3
	10,000	0.239	0.057	0.014	40.4
	20,000	0.557	0.079	0.026	46
	50,000	1.928	0.106	0.071	49.9
	100,000	4.169	0.137	0.192	50.5
30	100	0.003	0.021	0.004	21.8
	1,000	0.030	0.046	0.011	38.4
	5,000	0.158	0.077	0.016	50.9
	10,000	0.299	0.109	0.028	54.2
	20,000	0.866	0.131	0.054	59.1
	50,000	2.816	0.164	0.102	67.8
	100,000	6.165	0.230	0.192	70.5
50	100	0.004	0.031	0.006	26.3
	1,000	0.036	0.093	0.018	51.7
	5,000	0.239	0.198	0.031	72.6
	10,000	0.708	0.234	0.048	73.2
	20,000	1.496	0.251	0.142	83.3
	50,000	4.542	0.359	0.163	92.6
	100,000	10.088	0.492	0.472	98.9
100	100	0.008	0.078	0.012	36.1
	1,000	0.078	0.322	0.039	78.6
	5,000	0.648	0.596	0.072	104.9
	10,000	1.374	0.796	0.136	118.6
	20,000	3.328	0.936	0.181	129.8
	50,000	9.255	1.234	0.427	142.6
	100,000	21.359	1.698	0.686	155.9
200	100	0.012	0.216	0.027	48.1
	1,000	0.172	1.197	0.105	114.3
	5,000	1.421	2.350	0.223	161
	10,000	3.125	2.872	0.418	174.7
	20,000	7.236	3.911	0.534	195.9
	40,000	16.130	5.066	0.864	215.7
	80,000	36.252	6.525	1.547	233.8
300	100	0.021	0.426	0.046	56.7
	1,000	0.291	2.363	0.194	136.6
	5,000	2.538	5.368	0.515	197.8
	10,000	5.222	7.187	0.779	223.2
	20,000	11.847	9.573	1.149	248.1
	40,000	26.222	12.496	1.670	272.5
	80,000	58.863	15.490	2.924	296.6
500	100	0.039	0.890	0.073	62.1
	1,000	0.603	6.538	0.473	173.2
	5,000	5.094	16.820	1.292	261.3
	10,000	11.616	23.065	1.805	295.3
	20,000	22.105	28.940	2.699	324.6
	50,000	66.267	42.398	4.929	371
1,000	100	0.084	2.806	0.201	75.2
	1,000	1.311	29.882	1.894	239
	5,000	14.877	97.380	6.016	385.4
	10,000	32.063	134.708	8.987	440.4
	20,000	65.047	174.532	12.640	492.6
	50,000	155.843	244.268	20.227	561.8

Theorem 4.3 (Feasibility):
Let $\beta \in (0,1)$ be any small confidence parameter value. If S and k are such that

$$\binom{k+n_x-1}{k} \sum_{i=0}^{k+n_x-1} \binom{S}{i} \epsilon^i (1-\epsilon)^{S-i} \leq \beta, \quad (9)$$

then $\mathcal{P}^S\{\mathcal{V}(x^*_{S,k}) \leq \epsilon\} \geq 1-\beta$.

The final result establishes that the objective value of CCP_ϵ (whose optimal objective value will be denoted as J^*_ϵ) can be approached at will, provided that sampled constraints are optimally removed. Let $\mathcal{A}_{opt}$ be the optimal constraints removal algorithm which leads – among all possible eliminations of k constraints out of S – to the best possible improvement in the cost objective; further, let $x^*_{S,k,opt}$ and $J^*_{S,k,opt}$ be the corresponding optimal solution and objective value. We have the following theorem (again, proved in [2]).

Theorem 4.4 (Optimality):
Let $\beta \in (0,1)$ be any small confidence parameter value, and let $\nu \in (0,\epsilon)$ be a performance degradation parameter value. If S and k are such that

$$\binom{k+n_x-1}{k} \sum_{i=0}^{k+n_x-1} \binom{S}{i} \epsilon^i (1-\epsilon)^{S-i} + \sum_{i=k+1}^{S} \binom{S}{i} (\epsilon-\nu)^i (1-\epsilon+\nu)^{S-i} \leq \beta, \quad (10)$$

then

(i) $\mathcal{V}(x^*_{S,k}) \leq \epsilon$
(ii) $J^*_{S,k,opt} \leq J^*_{\epsilon-\nu}$

simultaneously hold with probability at least $1-\beta$.

One optimal way of removing constraints consists in discarding those constraints that lead to the largest possible improvement of the cost function. This approach is implemented by the following integer program, which has been described and investigated in [24], [25] and [26]:

$$\begin{aligned} \underset{x\in\mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & g(x,\xi^i) - M z_i \leq 0, \quad i = 1,\dots,S, \\ & \textstyle\sum_{i=1}^S z_i \leq k, \qquad\quad z \in \{0,1\}^S. \end{aligned} \quad (11)$$

where M is a constant large enough so that, if $z_i = 1$, then the constraint is satisfied for any candidate solution x. For $k = 0$, the formulations (2) and (11) are equivalent. By construction, problem (11) provides a framework for optimally selecting the constraints to be removed based on the inequality (10). However, solving (11) may be computationally challenging due to the increase in complexity from (2) to (11) that arises from the introduction of one binary variable per each of the S scenarios. In recent years, there have been developed strengthening procedures (see [27] and [28]) for some special structured problems, that significantly improve upon the formulation (11).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

V. POOL & DISCARD ALGORITHM

The Discarding part of the algorithm consists of utilizing the index set $\mathcal{I}$, finding the support scenarios among this set and finding the one scenario, whose removal decreases the optimal objective value the most – this is repeated k times, where k is either set a priori (by Theorem 4.3), or is terminated once an estimate of the probability of violation of obtained solution $\mathcal{V}(x)$ reaches certain threshold. This approach is almost identical to the one discussed in [29] (called greedy constraint removal), with the distinction that our algorithm utilizes the Pooling step and uses warm-starts (primarily utilizing $\mathcal{I}$) throughout the iterations and as such can be rather effective (this will be demonstrated in the following sections). The P&D algorithm can be summarized as follows:

Step 0. Solve the pooling part described above to obtain $\mathcal{I}^*$ and x^*. Set $\gamma > 0, k > 0, \mathcal{I}_p = \emptyset$.

Repeat k times, or terminate once an estimate of $\mathcal{V}(x^*)$ reaches a threshold:

Step 1. Find the set of support scenarios $\mathcal{I}_r \subset \mathcal{I}^*$ – either by examining the slacks ($s^i > -\gamma$) or the associated dual variables ($\mu^i > \gamma$).

Step 2. For each of the support scenarios $i_r \in \mathcal{I}_r$, solve the following problem:

$$\begin{aligned} \underset{x \in \mathcal{X}}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & g(x, \xi^i) \leq 0, \\ & i \in \{1, \ldots, S\} \setminus \{i_r \cup \mathcal{I}_p\}, \end{aligned} \tag{12}$$

using the Pooling part, warm-started by using $\mathcal{I} = \mathcal{I}^* \setminus \{i_r\}$ and $x = x^*$. Denote the solution to (12) as $x^*_{i_r}$, its optimal objective function value $v^*_{i_r}$ and its final set of scenarios $\mathcal{I}^*_{i_r}$.

Step 3. Find the index with the best optimal objective value: $i^* = \underset{i_r}{\operatorname{argmin}}\, v^*_{i_r}$. Set $x^* = x^*_{i^*}$, $\mathcal{I}^* = \mathcal{I}^*_{i^*}$ and add the corresponding scenario to the set of permanently discarded ones $\mathcal{I}_p$.

The parameter γ can be, in theory, set to 0 – what discourages us from doing so are the implementation issues of numerical computing. When reporting the optimal dual variables μ the solvers rarely return exactly 0, more often, we get values ranging from 10^{-8} to 10^{-16} (the same goes for the slacks in the active constraints). If we did set γ to 0 we would (likely) have to consider all the scenarios as possible support scenarios and the execution of the algorithm would be significantly prolonged. Unless stated otherwise, the parameter γ was set to 10^{-6}. It should be added, that Step 2. of the Discarding part can be fully parallelized to work more efficiently on multi-core machines or distributed computing environments.

A. NUMERICAL EXAMINATION – ASSET ALLOCATION PROBLEM CONTINUED

We will return to the same problem structure (7) again and examine the computational time for the whole P&D algorithm for varying number of variables and scenarios. In the Discarding part of the algorithm, we decided to discard $k = \lfloor \epsilon S \rfloor$ scenarios – note that this choice does not guarantee, that the resulting solution obtained by the P&D algorithm will be a ϵ-level feasible, not to mention having the objective value close to the optimal value objective J^*_ϵ.

To examine the effect of the warm-start in the discarding part (using the best solution x^* and the index set $\mathcal{I}^*$ from the previous iteration), we will first compare the computational times of the P&D algorithm, an algorithm that uses just the Pooling part without warm-starts (denoted as "PnoD"), and an algorithm that uses neither Pooling nor Discarding (denoted as "noPnoD", which is essentially the one used in [29]), on a small-scale example ($n = 20, \epsilon = 0.01$).

TABLE 2: Comparing the algorithms. Average over 10 runs.

n	ϵ	S	k	noPnoD [s]	PnoD [s]	P&D [s]
20	0.01	100	1	0.02	0.03	0.01
		1,000	10	2.35	1.10	0.23
		2,500	25	15.60	3.59	0.68
		5,000	50	78.19	8.90	1.66
		10,000	100	371.48	22.79	3.71
		20,000	200	1,931.58	59.14	8.64

The comparison is summarized in Table 2 – the utilization of Pooling and the warm-starts in Discarding combined provide immense computational savings compared to the other two methods (while arriving at the exact same solution). To further compare the effectivity of the P&D algorithm, we set the parameters n, S, and k to the same values that can be found in [29] and compare the computation times directly (although they used different distributions for the asset returns, the problem structure is exactly the same). The results of the computation are reported in Table 3 – for $n = 20$, the results reported in [29] are comparable with the noPnoD variant of the algorithm, with slight improvement that is most likely caused by a more powerful machine and a newer version of the optimization solver. In the largest instance, the P&D algorithm was more that 200 times faster. For the $n = 200$, the authors in [29] used a random scenario removal strategy, instead of the greedy one (removing one of the support scenarios at random, instead of the one whose removal decreased the optimal objective value the most) – this algorithm is $O(n)$ times faster than the greedy one, but results in an inferior solution. In this setting, the P&D variant with randomized removal (denoted as P&D*) was almost 500 times faster than the one in [29].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

TABLE 3: Comparing the different algorithms. The results with a * are for random scenario removal. Average over 5 runs.

n	S	k	[29] results [s]	noPnoD [s]	PnoD [s]	P&D [s]	P&D* [s]
20	2,500	18	15.1	11.15	2.56	0.49	0.04*
	5,000	76	138.0	117.16	13.63	2.36	0.18*
	10,000	220	875.3	772.93	47.03	6.99	0.54*
	20,000	582	5,412.4	5,315.1	176.28	23.44	1.73*
200	20,000	582	5,521*	-	-	3,486.5	44.3*
	40,000	1,164	26,307*	-	-	8,312.4	126.3*
	80,000	2,328	120,535*	-	-	19,521.7	241.7*

The real crux of the matter, however, is the following: "How good a solution (in terms of ϵ-level feasibility and objective value) do we get by using the P&D algorithm?" The remarkable thing about our optimal asset allocation problem is that for a chosen value of ϵ, we can get the optimal solution by solving the SOCP (6). Moreover, for every asset allocation x, we can find the corresponding ϵ quantile of the returns exactly. Or, alternatively, we can for a given value of the returns t and a given asset allocation x compute (again, exactly) the probability $\mathcal{P}\{t \leq \sum_{j=1}^{n} r_j x_j\}$ (i.e., the smallest value of ϵ, for which our choice of x and t is feasible).

For the examination, we chose a problem with $n = 30$ assets and $\epsilon = 0.01$. The optimal objective value (obtained by solving (6)) was 1.0309. The results are summarized in Table 4, Figure 1 and Figure 2. Using the formula for the needed number of scenarios from [18], with $\beta = 10^{-10}$, we get that to obtain a feasible solution to this problem with high probability $(1 - \beta)$, we need to solve (7) with at least $S = 8{,}547$ scenarios (without any discarding). The solution to this problem had the objective value 1.0179 (third column of Table 4), with the probability of violation 0.0029 (i.e. $\mathcal{P}\{t \leq \sum_{j=1}^{n} r_j x_j\} = 0.0029$) – i.e. we obtained a feasible solution, but with a rather poor objective value.

Afterwards, we ran the Discarding part of the algorithm, discarding $\lfloor \epsilon S \rfloor$ scenarios. The objective value improved to 1.0318 (fifth column of the table), but the corresponding probability of violation increased to 0.0138 – meaning that the combination of x and t (obtained after discarding) was no longer feasible. However, during the Discarding part of the algorithm we stored the particular solutions in each iteration. This allows us to find the last admissible (feasible) solution and find its corresponding objective and a number of scenarios that we discarded to get it – in this case the objective value was 1.0291 (seventh column of the table) with 31 discarded scenarios. An interesting thing to note is that even for 5,000 scenarios we still get a feasible solution (with probability of violation 0.0041) and can remove some scenarios, but for the lower numbers of scenarios even the "robust" solution is not feasible.

When we vary the number of scenarios several interesting phenomena appears. Firstly, when increasing the number of scenarios S we get a smaller value of the "robust" solution objective (the solution after the Pooling part) and smaller corresponding probability of violation (both of these are rather intuitive). Secondly, when we increase the number of scenarios, the probability of violation of the solution after discarding $\lfloor \epsilon S \rfloor$ scenarios approaches ϵ and the number of removed scenarios for an admissible solution gets closer to $\lfloor \epsilon S \rfloor$. Thirdly, and most impressively, the admissible solution objective gets surprisingly close to the optimal value of (6).

Another feature of the P&D algorithm is that since we remove one scenario at a time, we can use the successive results to construct an approximation of the trade-off between reliability and optimal objective function value. This is best shown on Figure 1, where we can see the progression of the P&D algorithm for different number of scenarios – each point corresponds to a solution with different number of removed scenarios (typically, more removed scenarios correspond to points more up and to the right). We included the optimal trade-off curve obtained by solving the SOCP (6) for different values of ϵ (called "exact solution" in the legend of Figure 2).

It must be emphasized that the P&D algorithm does not in any way incorporate any knowledge about the underlying distribution of the random variables. All it "sees" are the realizations in the form of individual scenarios.

The relationship between computational times, number of scenarios S, number of variables n and chosen probability of violation ϵ is further investigated in Tables 5, 6 and 7. From these results we can see that when we increase ϵ, the computation times increase linearly. The same cannot be said for when increasing S – the number of scenario removals and computational times of the pooling steps grow simultaneously, resulting in a superlinear increase computational time. Similar (and more impactful) behaviour can be observed in Table 7, where the number of variables n changes – this results in a larger number of possible support scenarios and larger solution times for the successive optimization problems.

When we increase the number of scenarios S, the resulting solutions (after discarding $\lfloor \epsilon S \rfloor$ scenarios) get very close to the true optimum J_ϵ^*, although it was not guaranteed by any theory. In similar fashion, the probability of violation of the solutions get very close to ϵ.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

TABLE 4: The "quality" of the solutions produced by the P&D algorithm, $n = 30$, target $\epsilon = 0.01$, optimal objective $J^*_\epsilon = 1.0309$. Varying number of scenarios, single run of the algorithm.

S	k	RSO	RSPV	ODϵS	PVDϵS	ASO	NRS
500	5	1.0324	0.0291	1.0400	0.0466	–	–
1,000	10	1.0292	0.0183	1.0377	0.0368	–	–
2,000	20	1.0289	0.0120	1.0336	0.0220	–	–
5,000	50	1.0231	0.0041	1.0326	0.0160	1.0303	31
8,547	85	1.0179	0.0029	1.0318	0.0138	1.0291	53
20,000	200	1.0167	0.0014	1.0319	0.0128	1.0304	147
50,000	500	1.0140	0.0004	1.0310	0.0111	1.0305	463
100,000	1,000	1.0129	0.0003	1.0309	0.0102	1.0308	980
250,000	2,500	1.0101	0.0001	1.0308	0.0101	1.0308	2,487

RSO – "robust" solution optimal objective value (no scenarios removed),
RSPV – "robust" solution probability of violation,
ODϵS – optimal objective value after discarding $\lfloor \epsilon S \rfloor$ scenarios,
PVDϵS – probability of violation after discarding $\lfloor \epsilon S \rfloor$ scenarios,
ASO – admissible solution optimal objective value,
NRS – number of removed scenarios for admissible solution.

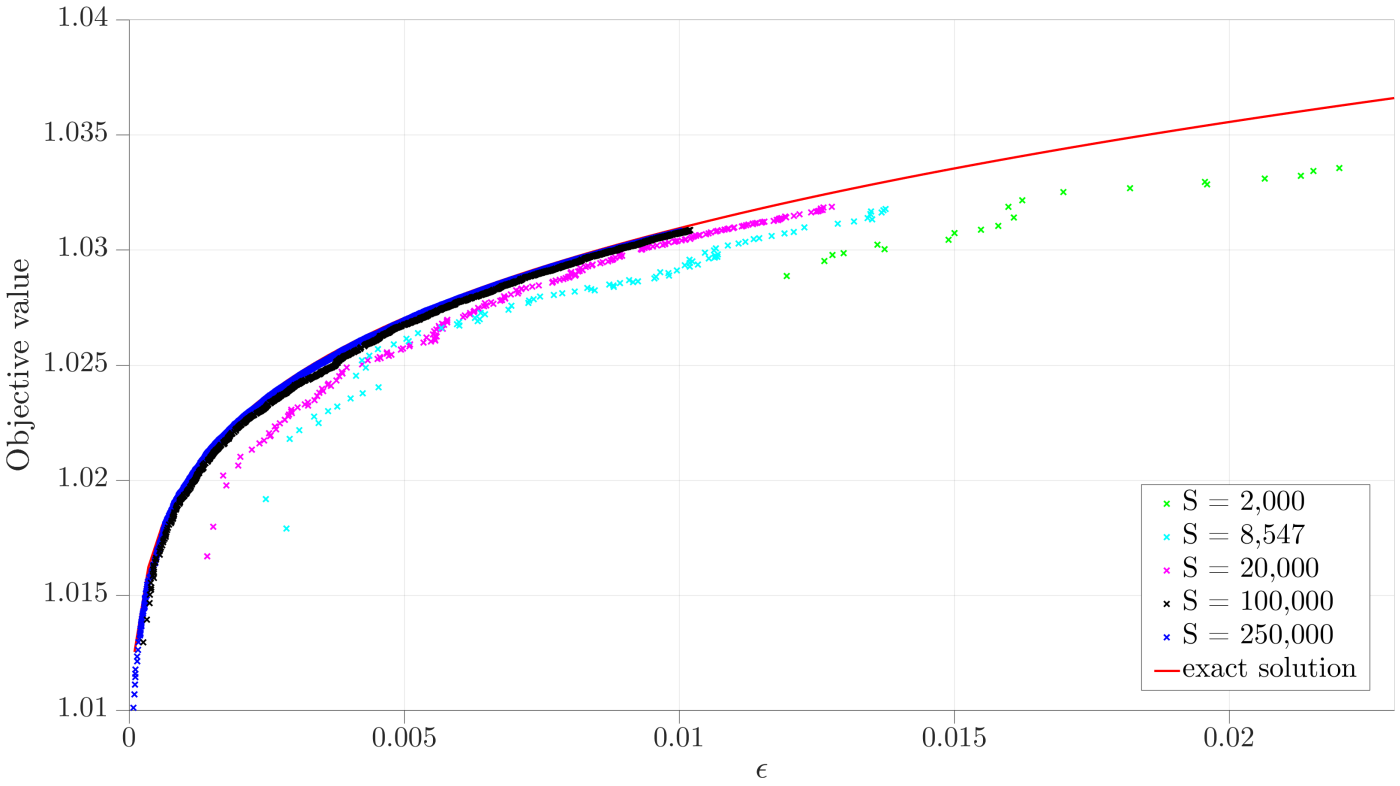


FIGURE 1: The "quality" of the solutions produced by the P&D algorithm, $n = 30$. Varying number of scenarios, single run of the algorithm.

TABLE 5: Results of the computations, $n = 20$. Average over 10 runs.

	$\epsilon = 0.02$ $J^*_\epsilon = 1.0280$			$\epsilon = 0.05$ $J^*_\epsilon = 1.0364$			$\epsilon = 0.15$ $J^*_\epsilon = 1.0517$		
S	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]
100	1.0456	0.1465	<0.1	1.0497	0.1859	0.1	1.0601	0.2584	0.2
1,000	1.0322	0.0411	0.4	1.0387	0.0700	1.0	1.0528	0.1709	2.9
2,500	1.0299	0.0295	1.4	1.0374	0.0602	3.1	1.0512	0.1548	8.0
5,000	1.0286	0.0237	3.2	1.0363	0.0539	6.7	1.0509	0.1508	17.5
10,000	1.0284	0.0225	7.5	1.0361	0.0513	15.3	1.0510	0.1514	39.0
20,000	1.0280	0.0211	17.2	1.0363	0.0514	37.7	1.0510	0.1516	93.7
50,000	1.0279	0.0205	53.3	1.0360	0.0504	121.7	1.0508	0.1505	317.2
100,000	1.0278	0.0202	146.0	1.0359	0.0500	335.6	1.0507	0.1497	914.4
250,000	1.0278	0.0200	854.5	1.0360	0.0500	2,058.9	1.0508	0.1502	5,987.6

ODϵS – optimal objective value after discarding $\lfloor \epsilon S \rfloor$ scenarios,
PVDϵS – probability of violation after discarding $\lfloor \epsilon S \rfloor$ scenarios.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

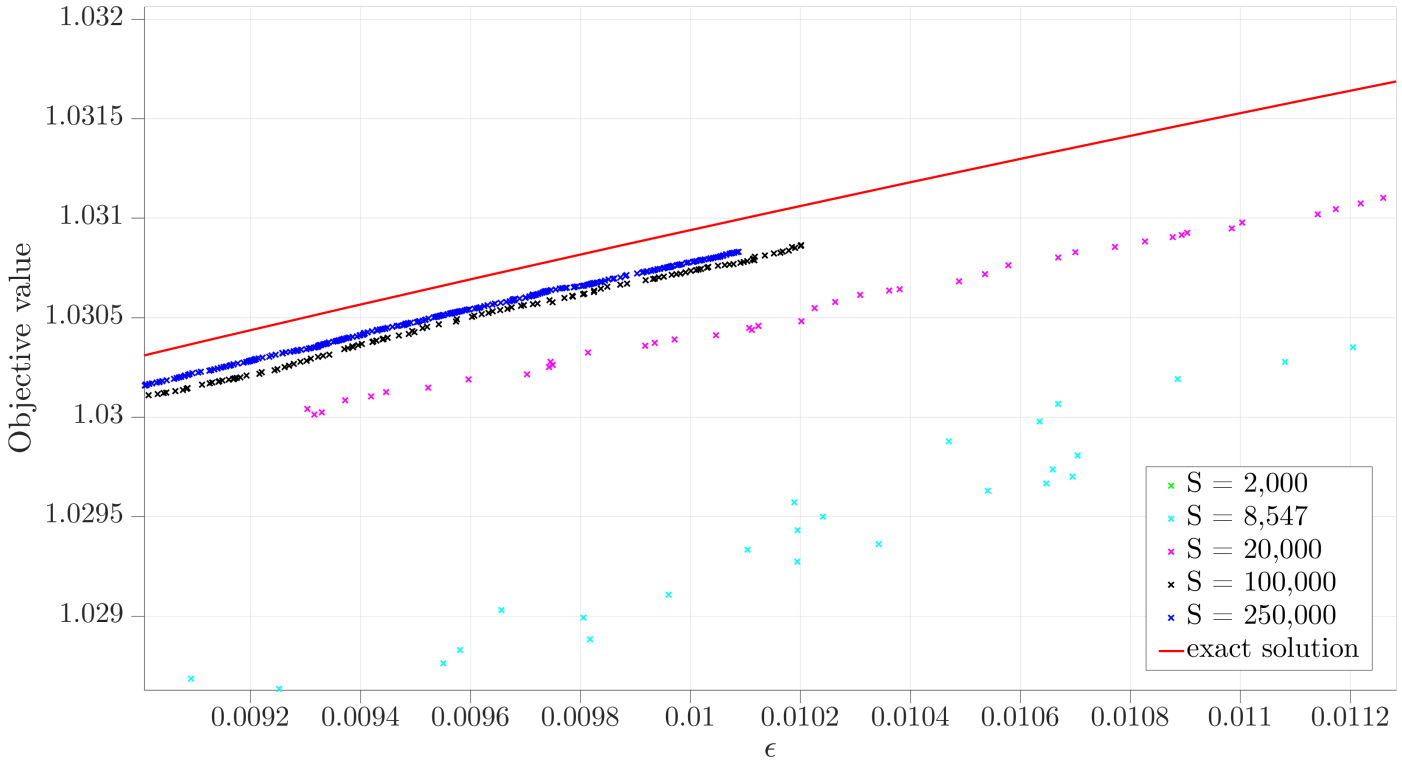


FIGURE 2: The "quality" of the solutions produced by the P&D algorithm, $n = 30$. Varying number of scenarios, single run of the algorithm. Close up on $\epsilon = 0.01$.

TABLE 6: Results of the computations, $n = 30$. Average over 10 runs.

	$\epsilon = 0.02$ $J^*_\epsilon = 1.0355$			$\epsilon = 0.05$ $J^*_\epsilon = 1.0433$			$\epsilon = 0.10$ $J^*_\epsilon = 1.0511$		
S	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]
100	1.0526	0.1785	<0.1	1.0585	0.2159	0.1	1.0636	0.2692	0.2
1,000	1.0406	0.0487	0.8	1.0457	0.0737	1.9	1.0527	0.1285	3.8
2,500	1.0375	0.0310	2.6	1.0441	0.0607	6.2	1.0514	0.1095	11.2
5,000	1.0369	0.0270	6.3	1.0434	0.0549	14.7	1.0512	0.1062	25.7
10,000	1.0357	0.0223	15.0	1.0434	0.0531	32.8	1.0508	0.1034	57.0
20,000	1.0356	0.0215	33.9	1.0430	0.0507	76.4	1.0508	0.1022	137.3
50,000	1.0354	0.0205	102.1	1.0430	0.0507	226.3	1.0506	0.1006	411.7

TABLE 7: Results of the computations, $\epsilon = 0.02$. Average over 10 runs.

	$n = 50$ $J^*_\epsilon = 1.0442$			$n = 100$ $J^*_\epsilon = 1.0544$			$n = 200$ $J^*_\epsilon = 1.0630$		
S	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]	ODϵS	PVDϵS	t [s]
100	1.0624	0.2130	0.1	1.0749	0.3086	0.2	1.0808	0.3758	0.7
1,000	1.0494	0.0563	2.2	1.0608	0.0775	9.0	1.0702	0.1030	36.4
2,500	1.0466	0.0346	7.5	1.0576	0.0448	34.3	1.0670	0.0592	140.8
5,000	1.0456	0.0286	19.6	1.0566	0.0351	83.6	1.0652	0.0422	430.6
10,000	1.0450	0.0250	45.8	1.0557	0.0282	204.8	1.0642	0.0308	1,034.4
20,000	1.0444	0.0224	97.5	1.0551	0.0244	444.2	1.0639	0.0268	2,626.2
50,000	1.0442	0.0209	289.1	1.0545	0.0214	1,351.4	1.0633	0.0226	7,662.8

VI. NONLINEAR JOINT CHANCE CONSTRAINED EXAMPLE

In this section we investigate the performance of the algorithm on nonlinear example that appeared in the numerical sections of the state-of-the-art methods in [30] and [31]. Both of these methods are scenarios (or sample) based and use the indicator function approximation (although they approach it in different ways). In the method described in [30], the constraints need to be convex in x and the problem can be a joint chance constrained one. In the method described in [31], the constraints do not have to be convex, but must be continuously differentiable in x and the authors deal with a single chance constraint only. The problem both papers have chosen for the numerical examination is the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

following one:

$$\begin{aligned} \underset{x\geq 0}{\text{minimize}} \quad & -\sum_{j=1}^{n} x_j \\ \text{subject to} \quad & \mathcal{P}\{\sum_{j=1}^{n} \xi_{ij}^2 x_j^2 - b \leq 0,\ i = 1,\dots,m\} \\ & \geq 1-\epsilon, \end{aligned} \tag{13}$$

where $\xi_{ij}, i = 1,\dots,m$ and $j = 1,\dots,n$ are independent and identically distributed standard normal random variables, $b \in \Re$. In the case of [31], $m = 1$ (a single chance constraint).

Optimal solution x^* of the problem (13), derived in [30], is:

$$x_1^* = x_2^* = \cdots = x_n^* = \left[b / F_{\chi_n^2}^{-1}((1-\epsilon)^{\frac{1}{m}}) \right]^{\frac{1}{2}}, \tag{14}$$

where $F_{\chi_n^2}^{-1}$ is the inverse chi-squared distribution function with n degrees of freedom.

Because of the nature of the problem (quadratic and convex), we were able to use the CPLEX solver, and utilize the problem modification feature again. We start the numerical examination with the same setting as [31]: $n = 10, m = 1, b = 10, \epsilon = 0.05$. The parameter γ that controls the selection of scenarios to discard, was set to 10^{-3}. Using the formula (14), the optimal objective value of this problem is -7.390. We generate a number of scenarios S (the values were log-spaced between 10^2 and 10^4) and set the P&D algorithm to discard $\lfloor \epsilon S \rfloor$ of them. After that we estimate the reliability $(1-\epsilon)$ of the obtained solution using 10^5 new scenarios. The results of the computations are summarized in Table 8. Unsurprisingly, the more

TABLE 8: Results of the computation. $J_\epsilon^* = -7.390$. Average values over 10 runs.

S	ODϵS	reliability	P&D [s]
100	-8.042	0.8535	0.44
167	-7.968	0.8700	1.08
278	-7.736	0.9039	1.87
464	-7.564	0.9255	4.17
774	-7.528	0.9322	7.94
1,292	-7.539	0.9356	14.80
2,154	-7.467	0.9422	27.53
3,594	-7.440	0.9454	51.09
5,995	-7.400	0.9491	92.14
10,000	-7.397	0.9497	174.75

scenarios are taken into account, the better (closer to the theoretical optimum) the result. The computational time is quite good, considering [31] report around 500 s as the computational time for their algorithm (that uses 500 scenarios and reports the optimal objective value -7.627) and the big-M mixed-integer formulation does not converge in an hour [31] (again, using "just" 500 scenarios).

The second setting we investigate is from [30]: $n = 10$, $m = 10$, $b = 100$, $\epsilon = 0.1$. The optimal objective value, using (14), is -20.82. Note that in this setting we are dealing with a "proper" joint chance constraint problem, with nonlinear (but convex) constraint functions. In the Pooling part of the algorithm, when we find a scenario with a violated constraint, we have a choice of either adding all of the m constraints that correspond to this scenario to the problem, or to add only the violated ones. In our implementation we chose the former, since it corresponds more closely to the description of the P&D algorithm we gave in the previous sections, although additional computational savings could be gained by properly implementing the latter approach. The number of scenarios used in the examination ranged between 10^2 and 10^4 and the number of scenarios to discard was set to $\lfloor \epsilon S \rfloor$. The results of the computations are listed in Table 9.

TABLE 9: Results of the computation. $J_\epsilon^* = -20.82$. Average values over 10 runs.

S	ODϵS	reliability	P&D [s]
100	-21.46	0.8042	6.67
251	-21.42	0.8305	15.14
631	-21.03	0.8720	51.24
1,585	-20.96	0.8853	162.21
3,981	-20.84	0.8962	494.94
10,000	-20.83	0.8975	1,670.65

To compare the results with the ones achieved in [30], where they used 10,000 scenarios for the computations – the numbers the authors report are a bit vague:
"Our algorithm typically requires less than 10 iterations to converge to the optimal value, and each iteration approximately takes 6 s on average."
The main objections being that their algorithm was presented on two problems with different dimensions (the one presented here and a smaller one), and that, at least judging from the figures (as there is no other way to find the value), their "optimal solution" was around -20.4, which is rather far from the real one. It is important to emphasize again that the P&D algorithm does not produce just one solution – as a sort of a by-product it generates a sequence of decisions, that are "optimal" with respect to an increasing number of discarded scenarios. When we estimate the reliability of these solutions, we get an approximation of the trade-off between the reliability level (1-ϵ) and optimal objective value. Naturally, this approximation gets better as we increase the number of scenarios. The approximation of the trade-off for the setting described above is depicted in Figure 3 – it shows just one run of the P&D algorithm for different number of scenarios and the optimal values computed using the formula (14). The reliability of the solution is estimated using 10^5 different scenarios. If we stopped the algorithm with 200 scenarios once the estimate of the reliability of the solution gets lower than the desired level $1-\epsilon$ and use the previous value, we would discard only 12 scenarios with the objective value -20.47 and estimated reliability 0.9143 – this takes around 6 s. Note that the robust solution for this problem, i.e. the solution for $\epsilon = 0$, is clearly 0, since

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

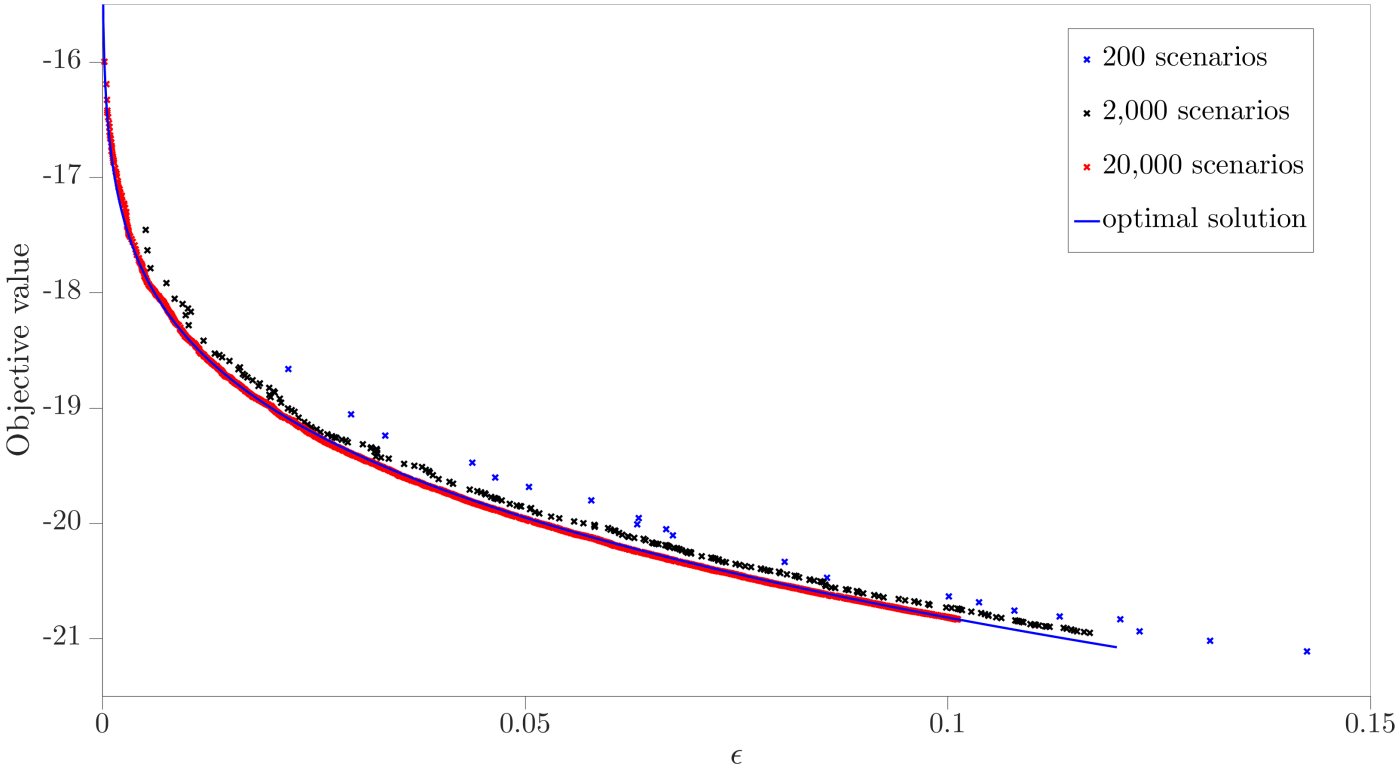


FIGURE 3: Approximation of trade-off between reliability and optimal objective value. Nonlinear joint chance constrained example.

each ξ_{ij}^2 can attain any nonnegative value.

VII. CONCLUSION

The main advantage of the P&D algorithm lies in the exploitation of the structure of the scenario design problem, which is done on two levels:

- The Pooling part of the P&D algorithm utilizes the fact that the number of support scenarios is usually very small compared to the number of sampled scenarios. By iteratively solving much smaller problems we can get the solution faster and need less memory, than we would need to solve the scenario design problem with all scenarios at once (compare the columns CTC and CTPP in Table 1).
- The Discarding part of the algorithm fully utilizes the set $\mathcal{I}$ that contains the current support scenarios, to find the ones that will be discarded (either be the greedy or the randomized algorithm). Since it only solves comparatively much smaller problems (and, each scenario removal should terminate in just a few iterations), it brings additional computational savings (compare the columns PnoD and P&D in Table 2). The combined effect of the two parts of the algorithm is best seen in the difference between columns noPnoD and P&D in Table 2.

The numerical examinations show that P&D algorithm provides a powerful framework for handling certain types of chance constrained optimization problems. When compared with conventional approaches [29] on a linear example (see Table 3), it was several hundred times more efficient in the largest instances. On the nonlinear examples, it was on par with the state-of-the-art methods [31] and [30].

Further investigation are possible – in the Pooling part of the algorithm for joint chance constrained problems, the choice between including all constraints for a violated scenario, or just the ones that are violated, could bring additional computational savings. Also, the use of P&D (or just the use of Pooling) could be applied in other classes of convex optimization problems (e.i., semi-definite problems in control) that need to be set in chance constrained (or robust) setting and require the consideration of large number of scenarios.

REFERENCES

[1] J. Kudela, "Advanced decomposition methods in stochastic convex optimization," Ph.D. dissertation, Brno University of Technology, 2019.

[2] M. C. Campi and S. Garatti, "A sampling-and-discarding approach to chance-constrained optimization: Feasibility and optimality," Journal of Optimization Theory and Applications, vol. 148, pp. 257–280, 2011.

[3] A. Charnes, W. W. Cooper, and G. H. Symonds, "Cost horizons and certainty equivalents: An approach to stochastic programming of heating oil," Management Science, vol. 4, no. 3, pp. 235–263, 1958.

[4] L. B. Miller and H. Wagner, "Chance-constrained programming with joint constraints," Operations Research, vol. 13, no. 6, pp. 930–945, 1965.

[5] A. Prekopa, "On probabilistic constrained programming," in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2990726, IEEE Access

Proceedings of the Princeton Symposium on Mathematical Programming. Princeton University Press, 1970.
[6] ——, "Contributions to the theory of stochastic programming," Mathematical Programming, vol. 4, no. 1, pp. 202–221, 1973.
[7] ——, Stochastic Programming, 1st ed. Kluwer, 1995.
[8] A. Ruszczynski and A. Shapiro, Eds., Stochastic Programming, 1st ed., ser. Handbooks in Operations Research and Management Science. Elsevier, 2003, vol. 10.
[9] R. Henrion and C. Strugarek, "Convexity of chance constraints with independent random variables," Computational Optimization and Applications, vol. 41, no. 2, pp. 263–276, 2008.
[10] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, Robust Optimization, 1st ed. Princeton University Press, 2009.
[11] A. Nemirovski, "On safe tractable approximations of chance constraints," European Journal of Operational Research, vol. 219, no. 3, pp. 707–718, 2012.
[12] A. Nemirovski and S. A., "Convex approximations of chance constrained programs," SIAM Journal on Optimization, vol. 17, no. 4, pp. 959–996, 2006.
[13] R. Henrion and W. Römisch, "Metric regularity and quantitative stability in stochastic programs with probabilistic constraints," Mathematical Programming, vol. 84, no. 1, pp. 55–88, 1999.
[14] ——, "Hölder and Lipschitz stability of solution sets in programs with probabilistic constraints," Mathematical Programming, vol. 100, no. 3, pp. 589–611, 2004.
[15] B. R. Barmish and P. S. Shcherbakov, "On avoiding vertexization of robustness problems: The approximate feasibility concept," IEEE Transactions on Automatic Control, vol. 47, no. 5, pp. 819–824, 2002.
[16] S. P. Boyd and L. Vandenberghe, Convex Optimization, 1st ed. Cambridge University Press, 2004.
[17] G. C. Calafiore and M. C. Campi, "Uncertain convex programs: randomized solutions and confidence levels," Mathematical Programming, Ser. A, vol. 102, no. 1, pp. 25–46, 2005.
[18] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," SIAM Journal of Optimization, vol. 19, no. 3, pp. 1211–1230, 2008.
[19] V. L. Levin, "Application of E. Helly's theorem to convex programming, problems of best approximation and related questions," Mathematics of the USSR-Sbornik, vol. 8, pp. 235–247, 1969.
[20] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A modeling language for mathematical optimization," SIAM Review, vol. 59, no. 2, pp. 295–320, 2017.
[21] J. Bezanson, A. Edelman, S. Karpinski, and S. V. B., "Julia: A fresh approach to numerical computing," SIAM Review, vol. 59, pp. 65–98, 2017.
[22] IBM Corp, IBM ILOG CPLEX Optimization Studio: CPLEX User's Manual. Version 12, Release 7., 2016. [Online]. Available: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/pdf/usrcplex.pdf
[23] C. M. Lagoa, X. Li, and M. Sznaier, "Probabilistically constrained linear programs and risk-adjusted controller design," SIAM Journal on Optimization, vol. 15, no. 1, pp. 938–951, 2005.
[24] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," SIAM Journal on Optimization, vol. 19, no. 2, pp. 674–699, 2008.
[25] J. Luedtke, S. Ahmed, and G. L. Nemhauser, "An integer programming approach for linear programs with probabilistic constraints," Mathematical Programming, Ser. A, vol. 122, no. 2, pp. 247–272, 2010.
[26] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro, "Sample average approximation method for chance constrained programming: Theory and applications," Journal of Optimization Theory and Applications, vol. 142, no. 2, pp. 399–416, 2009.
[27] Y. Song, J. Luedtke, and S. Küçükyavuz, "Chance-constrained binary packing problems," INFORMS Journal on Computing, vol. 26, no. 4, pp. 735–747, 2014.
[28] S. Ahmed, J. Luedtke, Y. Song, and W. Xie, "Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs," Mathematical Programming, vol. 162, no. 1–2, pp. 51–81, 2017.
[29] B. K. Pagnoncelli, D. Reich, and M. C. Campi, "Risk-return trade-off with the scenario approach in practice: A case study in portfolio selection," Journal of Optimization Theory and Applications, vol. 155, no. 2, pp. 707–722, 2012.
[30] F. Shan, L. Zhang, and X. Xiao, "A smoothing function approach to joint chance-constrained programs," Journal of Optimization Theory and Applications, vol. 163, no. 1, pp. 181–199, 2014.
[31] L. Adam and M. Branda, "Nonlinear chance constrained problems: Optimality conditions, regularization and solvers," Journal of Optimization Theory and Applications, vol. 170, no. 2, pp. 419–436, 2016.

JAKUB KŮDELA received his M.S. degree in mathematical engineering from Brno University of Technology in 2014 and received his Ph.D. degree in Applied Mathematics from Brno University of Technology in 2019.

From 2018, he works as a Research Assistant with the Institute of Automation and Computer Science, Brno University of Technology. His research interests includes the development of computational methods for various optimization problems and engineering applications.

PAVEL POPELA received his Ph.D. in 1998 in Econometrics from Charles University Prague.

From 1986, he works as a senior lecturer and researcher at Brno University of Technology. His research is mainly focused on stochastic programming models and methods.

A12

Journal of Cleaner Production 230 (2019) 1294–1304

Contents lists available at ScienceDirect

Journal of Cleaner Production

journal homepage: www.elsevier.com/locate/jclepro

Multi-objective strategic waste transfer station planning

Jakub Kůdela [a, *], Radovan Šomplák [b], Vlastimír Nevrlý [c], Tomáš Lipovský [c], Veronika Smejkalová [c], Ladislav Dobrovský [a]

[a] Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, VUT Brno, Technická 2896/2, 616 69, Brno, Czech Republic
[b] Sustainable Process Integration Laboratory, SPIL, NETME Centre, Faculty of Mechanical Engineering, Brno University of Technology, VUT Brno, Technická 2896/2, 616 69, Brno, Czech Republic
[c] Institute of Process Engineering, Faculty of Mechanical Engineering, Brno University of Technology, VUT Brno, Technická 2896/2, 616 69, Brno, Czech Republic

ARTICLE INFO

Article history:
Received 15 January 2019
Received in revised form
30 April 2019
Accepted 14 May 2019
Available online 16 May 2019

Keywords:
Transfer station
Waste treatment
Stochastic programming
Strategic planning
Multi-objective model

ABSTRACT

The production of mixed municipal waste changes due to the increase of separation rate, urbanization and other factors. The future changes in the legislation and technology development influences the way the waste is being treated. Thus the method, location and capacity of processing sites are unknown. The realization of future projects can be supported by the developing of transportation infrastructure. Such a feature may be represented by a robust transfer station grid, which can be designed to handle all possible future realizations and technological solutions (establishment of waste treatment facilities). The paper presents an approach utilising a mathematical model for the design of transfer stations. It is formulated as a multi-objective two-stage mixed-integer stochastic programming problem, where the trade-off between the environmental aspect and the economic viability is considered. The model is tested through a case study for the Czech Republic, where the waste treatment of over 6,000 municipalities is analysed. The solutions for different preferences are assessed throughout the principle called out-of-sample stability with 10,000 scenarios. The optimal decision consists of the robust transfer station grid with selected locations and their respective capacity. The particular solution with respect to the potential trade-off suggests to save 1.148 mil of travelled kilometres with only 0.77 mil EUR. The output in the form of decision support can serve possible stakeholders from the field of waste management to plan more sustainable projects.

1. Introduction

Most countries are struggling with changing demographic conditions. The population growth can be identified in developing countries, while in developed nations the age of the population is steadily growing (World Bank Open Data, 2018). The paper by Klemeš et al. (2017) is worth mentioning, because they summarized the progress in the sustainability applications from the recent years. At the same time, there is an ubiquitous technological boom and a change in the lifestyle associated with it. These aspects have an impact on products that are reflected in the waste produced, in terms of the absolute amount (Lou et al., 2017) and the waste composition (Chen, 2018a). Apart from the economic aspect, also the areas focusing on saving primary raw materials (Chen, 2018b) and reducing the emissions (Fan et al., 2018a) are becoming more prominent. An example may be a package for circular economy issued in May 2018 by EU (Directive (EU) 2018/849, 2018/850 2018/851 and 2018/852 – required to transpose the directives into national law of the member states by 5 July 2020) or an earlier document (from the year 2008), that anchors the preference of waste management in "waste hierarchy" (Directive (EU) 2008/98/EC). This hierarchy has been analysed (Gharfalkar et al., 2015) in more detail. The EU legislation is reflected by the local legislation of the EU member states, where binding milestones with fixed deadlines are defined.

Changing the way of waste treatment is only possible with the corresponding development of the waste processing infrastructure. Experts in this field are focused on the development of new sophisticated approaches for complex planning, where the main decision criterion is not only the cost but also the environmental point

* Corresponding author.
E-mail address: jakub.kudela@vutbr.cz (J. Kůdela).

https://doi.org/10.1016/j.jclepro.2019.05.167
0959-6526/

of view (Barbosa-Póvoa et al., 2018). The optimization models regarding the waste management can be divided into main categories:

- location problem, see Wichapa and Khokhajaikiat (2017),
- allocation problem, see Boonmee et al. (2018),
- network flow, see Tian et al. (2018),
- supply chain, see Islam and Huda (2018),
- and other.

The tasks mentioned in the bullet points are significantly affected by transportation cost. The other point of view represents the carbon emission production, which can be also used as a criterion in such tasks. The transition to high levels of renewable energy is included in future plans (Walmsley et al., 2015). Infrastructure building is influenced by local trends that are evolving due to legislative changes. For new projects, the economic sustainability is the essential aspect. The planning of the facility commissioning in waste management is difficult due to a long approval period (Putna et al., 2018). The projects are significantly affected by local environmental organizations, that often stop the project (Hsu, 2006). For these reasons, the forecasting of future waste handling is very difficult. Specific forecast contains many indeterminate factors which are projected in the main indicators (Cervantes et al., 2018). The current state of the waste handling, which is an important input for the simulations of future development (Šomplák et al., 2017) was analysed.

Since, the transportation is the key part of the whole chain of waste flow, deep evaluations of its component are needed. Most mathematical models solve the transport within the optimization of processing capacities, but their locations are set (Peri et al., 2018). However, the resulting flow allocation is not effective when changing the parameters of the terminal facility. On the other hand, transport and infrastructure models are usually solved with fixed processing grid (existing processing infrastructure). There exist several location allocation models in waste management. Bojic et al. (2013) examined the problem of allocating solid biomass power plants in Serbia. Sarker et al. (2018) focused on designing a logistic system for bio-methane gas production. However, these models do not include any sources of uncertainty and focus only on a single objective.

In the case of strategic decision-making in transport (Boonmee et al., 2018), all possible networks in the debris operation process were considered. It consists of waste collection and separation sites, processing and recycling sites, disposal sites and market sites to solve the post-disaster supply chain. Gambella et al. (2018) addressed a tactical problem of waste flow allocation from a waste operator point of view with the aim of minimizing the total management cost with considered profits from special subproducts. Coban et al. (2018) studied the possible disposal techniques for municipal authorities that could be applicable to Turkey with regards to the ever-growing amounts of the municipal solid waste. Another point of view was proposed by Zhao et al. (2016), where the complex network design problem was investigated. It considers the regional hazardous waste management system and searches for the transfer routes. The goal was to minimize the total cost and the inherent risk at the same time. Waste transfer stations and their locations were examined in the city of Nashik (India) by Yadav et al. (2016). The introduced study considers various waste treatment options, but the evaluation of new processing capacities is not included. The multi-objective approach for eco-design was proposed by Ji et al. (2016) and solved by Pareto optimization to obtain optimal transportation strategy. None of these approaches reflect the future possible legislation changes which can be projected in the development of certain technology while due to the local conditions, different locations are suitable for a realization of the different projects. This results in unstable optimal solutions when the corresponding legislation changes. Even when some similar thoughts were provided, big simplifications were considered, which limit the application to real problems.

This paper introduces the planning of transport infrastructure for municipal waste processing, specifically the location of the transfer stations with the capacity selection. The new approach is based on two-stage stochastic programming. The uncertainties are projected through the model parameters. The result is the suggestion of transfer stations placement, which is robust for future realization of unknown parameters. These unknown parameters were analysed separately in the following papers:

- The possibility of different processing facilities (Asefi and Lim, 2017) – sorting line, Waste-to-Energy plant, Mechanical-biological treatment plant, Monoblock, co-incineration with coal.
- The capacity of facilities (Rudi et al., 2017).
- The price variability – a requirement for the return of the investment – municipal or private investor (Ferdan et al., 2015), the development of the price and the demanded heat (Putna et al., 2018).
- The competitive waste market – construction of new facilities (Šomplák et al., 2014).
- Exporting/importing the waste (as a raw material) abroad (Botello-Álvarez et al., 2018).
- The legislation changes (Tomić et al., 2017).

From the above-mentioned points it is clear, that it is advantageous to consider a large number of scenarios for the description of the possible future realizations of the indeterminate parameters. The above-mentioned papers mostly focus on the testing instances or highly aggregated task with NUTS 3 (Nomenclature of Territorial Units for Statistics), which is very limited for practical use, especially for design of low-capacity facilities. This paper considers the analysed area on the more detailed level, which corresponds to LAU 2 (Local Administrative Unit). Such an approach entails significant demands on the compilation and the implementation of the mathematical model. The solvability and acceptable time and resources for computations play a crucial role. The approach will be described in detail in Section 5. The developed optimization model is multi-objective, providing a desired level of trade-off between the ecological and the economic objectives. This is achieved by minimizing the overall building and managing costs, and minimizing the total distance travelled by all the vehicles (and, hence, the emissions produced by those vehicles, see Fan et al. (2018b) for extensive review on air emission assessment) simultaneously.

2. Problem description

The main focus of the developed optimization model is to serve as a decision support on the selection of the location and the capacity of waste transfer stations. The purpose of these transfer stations is to be a transportation hub where the waste from the neighbouring municipalities is gathered, compressed, and loaded on a more cost-efficient vehicle before it is shipped to a waste treatment facility. See Gregor et al. (2017) for the evaluation of related costs. The decision on the placement and the capacity of these stations is a strategic one, as it needs to be made in advance, and has a lasting impact on the behaviour of the system (in this case, on the flows through the transportation network and the resulting costs and emissions). In the language of stochastic programming (Birge and Louveaux, 1997), these decisions are called the “first-stage” decisions, as they need to be made without the

knowledge of the particular realization of the uncertain parameters. These decisions must be made robust enough to be suitable for a wide range of possible future values of the uncertain parameters.

The optimization variable representing the decision on building the transfer station in a specific place is inherently binary, making the problem a mixed-integer one and adding extensive computational complexity. The relation between the cost of building a transfer station and its capacity is nonlinear, and (what is even more important) nonconvex, making the problem even more challenging. The non-linearity is caused by the purchasing of press equipment which decreases per unit of processed waste. This difficulty is overcome using the special ordered set of type 1 (or SOS1) variables (Williams, 2013), that linearize the cost function and help to decrease the computational complexity.

The other decisions, namely the transportation of waste, the usage of the transfer stations, and the choice of the waste treatment facility are all operational ones – they can adjust to the uncertain prices at the waste treatment facilities. In stochastic programming terms, these decisions are called "second-stage". To describe the transportation of waste (which essentially is a network flow) between the municipalities, a fitting mathematical structure is needed. In this case, because of the two different modes of transport ("normal one" and the one from transfer stations), two separate graphs are used. The first one has each municipality represented by a node and arcs describing the available road network between those municipalities (cf. Fig. 1 in the Case Study). The second graph describes the network using the transfer stations – the nodes are only the municipalities where there is either a possible transfer station location, or a waste treatment facility. The arcs describe the shortest paths between all pairs of possible transfer stations and waste treatment facilities.

The multi-objective nature of the model stems from the desire to design systems that are both economical and with as small environmental impact as possible. In this case, the environmental impact of a solution is measured in terms of the total distance travelled by all the vehicles used in the transportation of waste. The modelling technique employed to tackle the multi-objectivity of the problem is the standard scalarization one (Boyd and Vandenbarghe, 2004), where the two objectives are given different weights. These weights are used to construct a new single objective function that is minimized. By appropriately changing the weights, one obtains the desired trade-off curve (or a Pareto frontier) between the two objectives, as well as the corresponding optimal decisions.

As already stated in Section 1, there is a multitude of uncertain factors that need to be accounted for. This uncertainty in data is modelled as different possible scenarios, with the objective computed as an average (trade-off between costs and distance) over these scenarios. Intuitively, as the number (and the quality) of the considered scenarios grows, so does the quality of the decision that is based upon them. However, with rising number of scenarios comes an increase of computational difficulty, as to each scenario corresponds a separate set of second-stage decisions. This difficulty is addressed by the use of an appropriate optimization method described in Section 5.

3. Model formulation

All the necessary notation used to describe the mathematical model is summarized in Table 1. The mathematical model of the problem is developed using a combination of description styles to ease the notation. Most notably, the scalar product of two column vectors x, y is denoted as $x^T y$ (all vectors considered are column vectors). In the equations, some subscripts are hidden, meaning that the full vector of values is used – e.g., in Eq (3), the subscript j is missing to indicate that the inequality should hold for all of the corresponding values (component-wise for the two vectors).

The weighted objective function is given by Eq (1). The first

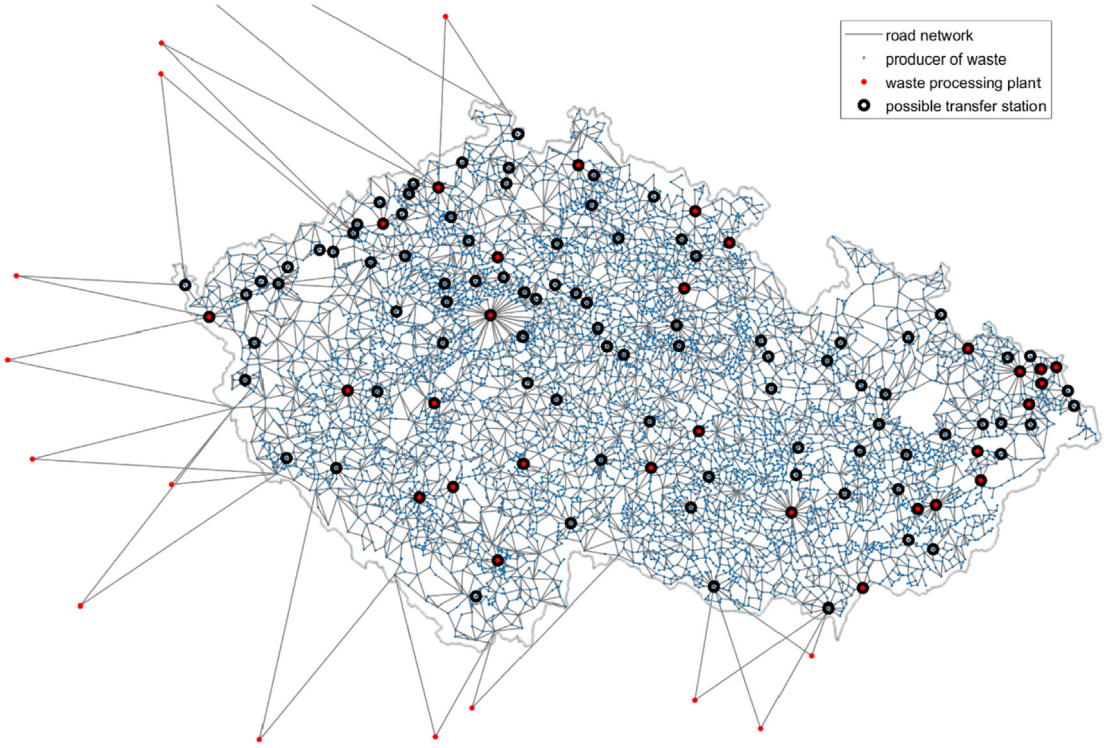


Fig. 1. A map showing the layout of the case study.

Table 1
The notation.

Type	Symbol	Description [unit]
Sets	$s \in S$	Set of scenarios
	$j \in J$	Set of nodes (municipalities)
	$i \in I \subset J$	Set of possible transfer stations
	$t \in T$	Set of possible options for transfer station capacities
Parameters	A_1	The first incidence matrix (connections between municipalities) [-]
	A_2	The second incidence matrix (transfer stations – treatment facilities) [-]
	d_1	Distances on the first incidence matrix (on A_1) [km]
	d_2	Distances on the second incidence matrix (on A_2) [km]
	c_1	Transfer costs, without the transfer stations (on A_1) [EUR/t]
	c_2	Transfer costs, using transfer stations (on A_2) [EUR/t]
	b_1	Capacity of vehicles on A_1 [t]
	b_2	Capacity of vehicles on A_2 [t]
	p_s	Probability of a scenario s [-]
	$e_{i,t}$	Cost of a construction of a transfer station at location i, with capacity option t [EUR]
	$k_{i,t}$	Capacity of a transfer station at location i, with capacity option t [t]
	$f_{j,s}$	Cost of processing waste at node j, scenario s [EUR/t]
	r_j	Production of waste at node j [t]
	q_j	Waste processing capacity of node j [t]
	λ	Scalarization parameter [-]
Variables	$\delta_{i,t}$	Decision on building the transfer station at location i, with capacity option t; binary (SOS1), first-stage [-]
	$x_{1,s}$	Flows on A_1 in scenario s; continuous, second-stage [t]
	$x_{2,s}$	Flows on A_2 in scenario s; continuous, second-stage [t]
	$y_{j,s}$	Amount of processed waste in node j, scenario s; continuous, second-stage [t]

addend denotes the expected costs, that are associated with the construction of the transfer stations, waste processing, and transportation. The second addend denotes the expected distance travelled by the vehicles transporting waste. Its computation is based on the amount of transported waste on the arcs of the networks and the capacity of the vehicles used on the two different networks (with and without the use of the transfer stations). The scalarization parameter $\lambda \in [0, 1]$ is used to describe the level of trade-off between the two objectives.

$$\begin{aligned} \textit{minimize}\ \ & \lambda \cdot \left(\sum_{i \in I, t \in T} e_{i,t}\delta_{i,t} + \sum_{s \in S} p_s \left(c_1^T x_{1,s} + c_2^T x_{2,s} + f_s^T y_s \right) \right) \\ & + (1-\lambda) \cdot \left(\sum_{s \in S} p_s \left(d_1^T x_{1,s}/b_1 + d_2^T x_{2,s}/b_2 \right) \right) \end{aligned} \tag{1}$$

The constraints describing the model take the following form:

$$A_1 x_{1,s} + A_2 x_{2,s} + y_s - \mathrm{r} = 0, \ \forall s \in S \tag{2}$$

$$y_s \leq q, \ \forall s \in S \tag{3}$$

$$\sum_{\textit{flows from } i \in I} x_{2,s} \leq \sum_{t \in T} k_{i,t}\delta_{i,t}, \ \forall s \in S, \ \forall i \in I \tag{4}$$

$$x_{1,s}, x_{2,s}, y_s \geq 0, \ \forall s \in S \tag{5}$$

$$\sum_{t \in T} \delta_{i,t} \leq 1, \ \forall i \in I \tag{6}$$

$$\delta_{i,t} \in \{0, 1\}, \ \forall i \in I, \ \forall t \in T \tag{7}$$

The constraint Eq (2) describes the "conservation of waste" – the net balance of the amount of waste that is produced in a municipality, transported (by the two different networks) in and out of a municipality, and processed in a municipality must be equal to zero. The constraint Eq (3) denotes the waste-processing capacities of the different municipalities (with the ones without a waste processing facility having $q_j = 0$). The constraint Eq (4) links the decision of building the transfer stations with the use of the associated network – the sum of the transported waste from a node i by the transfer station network must be less than the installed transfer station capacity in that node. The constraint Eq (5) ensures that the transportation and processing variables will be nonnegative. The two last constraints Eq (6) and Eq (7) together define the SOS1 variable – at most one of the possible capacities of the considered transfer stations must be chosen (with the possibility not to build any).

4. Case study

The case study involves the transfer station planning in the Czech Republic for the mixed municipal waste. In terms of scale, it deals with the most detailed description of the road networks and municipality structure available. In total, 6258 nodes (municipalities producing waste), 44 waste processing plants (15 of which were foreign, allowing a potential export of the waste to Germany or Austria) and 116 possible places for the transfer stations were considered (these sets are not mutually exclusive). For every possible transfer station 6 options for its capacity were considered.

The waste processing sites have the possibility to utilise potentially produced heat from waste, i.e. existing district heating systems, see Putna et al. (2018). In such locations, the construction of new facilities (Waste-to-Energy, Mechanical Biological Treatment, Processing of Refuse Derived Fuels) is considered. These technologies can effectively utilise residual municipal waste (Šomplák et al., 2014). The newly created facilities comply with (Directive (EU) 2018/850), which aims to move away from landfilling through the use of waste (material or energy recovery). The robust design of transfer stations promotes the financial sustainability of new projects and is therefore an important aspect of the transition to a more efficient waste management anchored in (Directive (EU) 2008/98/EC).

The first road network (connecting the municipalities, described by the incidence matrix A_1) had 24,770 arcs and is depicted in Fig. 1. In order to differentiate between the transportation of waste that

does or does not use the transfer stations, a separate road network was computed – for each possible transfer station was found the shortest path to each waste-processing plant. In this pre-processing step, 5075 shortest path optimization problems were solved, resulting in the second network (described by the incidence matrix A_2) with 5075 arcs (omitting the ones that started and ended at the same place). The transfer of waste when using the transfer stations is assumed according to (Gregor et al., 2017). Flows on the first network are considered to be serviced by vehicles with capacity $b_1 = 10$ t, and the flows from transfer stations are serviced by vehicles with capacity $b_2 = 24$ t. For the purpose of simplification, the vehicles are assumed fully loaded. This assumption has an undeniable effect on the resulting optimal solution. The model could be refined by considering less aggregated data (on a weekly/monthly basis instead on the yearly basis) and by obtaining the information about the utilization of the vehicles collecting the waste. Additionally, as pointed out by How et al. (2016) adding both weight and volume constraints significantly increases the precision of the transportation model.

The first-stage of the optimization problem consisted only of the planning decisions (on where to build the transfer stations) and is described by 696 binary variables. The second-stage of the optimization problem used 29,889 continuous decision variables.

The uncertain parameter that is considered in the model is the cost for processing the waste at the 44 different plants, which correspond with the legislation development and local conditions (such as the demand for heat, etc.). To appropriately capture the nature of the inherent uncertainty, 1000 possible scenarios for the waste treatment costs were constructed to be used within the optimization. The resulting optimization model had almost 30 million variables. The total number of constraints that depend on scenarios was 36,307, meaning that the optimization model had over 36 million constraints.

5. Optimization method

Because of the enormous number of variables of the considered optimization model, a specialized optimization method had to be employed. The particular block angular structure (see Birge and Louveaux, 1997) of this two-stage stochastic optimization problem is very well suited for the so-called Benders decomposition algorithm. This algorithm was originally developed as a method of solving large mixed integer optimization problems and it is thoroughly described by Kůdela et al. (2017). The variant of the algorithm that was used to solve the optimization problem further utilized the warm-start cuts developed by Kůdela and Popela (2017). The method works by decomposing the optimization problem into two different ones, namely the master problem and the subproblem. In the considered case, the master problem consists of all the first-stage variables, constraints that contain only the first-stage variables (constraints Eq (6) and Eq (7)) and a part of the objective function with only the first-stage variables. An additional continuous variable is attached to the master problem and is used as a link between the master problem and the subproblem.

The subproblem then includes every other variable and constraints Eq (2)-Eq (5). The algorithm progresses by alternating between solving the master problem and solving the subproblem, where the value of the first-stage variables is being fixed (i.e. the first-stage variables appear as constants in the subproblem). Depending on the solution of the subproblem, the master problem is augmented by the so-called feasibility and optimality cuts until an optimality criterion is met and the solution (the first-stage decision) is declared to be the optimal one (or to be within a specified optimality gap).

The real strength of the algorithm comes into light when solving the subproblem. The structure of the optimization model is such that the subproblem is naturally separable by scenarios, once the first-stage decisions are fixed. This means that instead of solving one large optimization problem with 30 million variables, one needs to solve 1000 problems with 29,889 variables instead – this can be done with the help of modern solvers and equipment in a reasonable time frame.

To further take the advantage of modern solvers, the concept of lazy constraints, proposed by IBM (International Business Machines Corporation) that develops the CPLEX solver (CPLEX, 2019), was utilized. Instead of solving the mixed-integer master problem completely for each newly generated optimality or feasibility cut, the cut-generation is moved to be within the solution procedure of the master problem. This is achieved by using the lazy cuts – each time a new incumbent solution for the mixed-integer master problem is found, the subproblems are solved and a depending on the result, new cuts are added (the incumbent is rejected), or the solution is deemed optimal (the incumbent is accepted). A flowchart describing the algorithm is depicted in Fig. 2.

The optimization was carried out for 7 different values of λ to obtain a set of advantageous and diverse trade-off decision. To test the computed optimal first-stage decision on building the transfer station network, a new set of 10,000 scenarios was generated. On these new scenarios, the transfer station network was fixed, and the optimization was carried out only with respect to the operational decision (transportation and waste treatment), which was done separately for each scenario. This evaluation is based upon a principle called out-of-sample stability (King and Wallace, 2012). Using this method, it was found that the objective function values from the optimization were less than 1% off from the results of the evaluation (suggesting that the 1000 scenarios used for optimization were "enough" to obtain out-of-sample stable solutions). The computations took about 1 h for each value of λ.

The optimization model and the decomposition algorithm were programmed in the high-performance dynamic language JULIA (Bezanson et al., 2017) with the JuMP package for mathematical optimization (Dunning et al., 2017), that is very well suited for large-scale scientific computing. The solution of the mixed-integer master problem was obtained using a branch-and-cut method (with the aforementioned lazy cuts), calling the CPLEX 12.6.3 solver

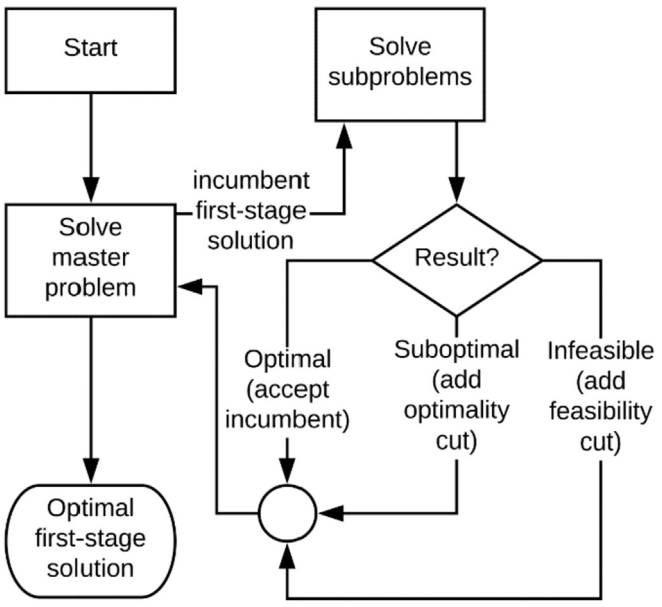


Fig. 2. A flow chart describing the Benders decomposition with lazy constraints.

(CPLEX, 2019). The MIP gap parameter was set at 1.5%, which was decided to be sufficiently low for this application. The individual subproblems in the second stage were solved by the primal-dual simplex method, calling the GUROBI 7.5 solver (GUROBI, 2019). This combination of solvers and algorithms achieved the best overall performance – the scheme reached the 1.5% optimality gap for the problem formulation with 1000 scenarios within 24 h for each considered value of the scalarization parameter λ. The computations were carried out on an ordinary computer (3.2 GHz i5-4460 CPU, 16 GB RAM).

6. Results and discussion

The results of the computations for 7 different values of λ are summarized in Table 2 – the mean and standard deviation (Std) for both objectives and the number and total capacity of transfer stations is presented. The trade-off between the overall costs and the total travelled distance, based on the value of the scalarization parameter λ, is best exemplified by the Pareto frontier graph in Fig. 3. The decisions based only on one of the objectives (corresponding to $\lambda = 0$ and $\lambda = 1$) are rather "extreme" to be used in as a support for decision making. Instead they serve as very useful reference points for comparing the possible trade-offs. A very important feature of the results is the "very steep" and "very shallow" slopes of the graph near the extreme values of λ (close to the single-objective optimal decisions). This means that for a very small compromise in one objective, large gains can be achieved in the other objective. This can be clearly seen, for example, on the solution for $\lambda = 0.05$, that is just very marginally worse in terms of costs than the decision that focuses solely on costs (for $\lambda = 1$, results for this setting were basically the same as the ones obtained by Kůdela et al. (2018)) but has 8.7% lower total travelled distance. For practical purposes, the best trade-off solutions seem to be obtained between $\lambda = 0.001$ and $\lambda = 0.03$. For $\lambda = 0.03$, the costs are only 0.3% higher than the best possible and the total distance travelled is 42.2% higher than the best possible. For $\lambda = 0.001$, the costs are 3.97% higher than the best possible and the total distance travelled is 1.35% higher than the best possible.

Another interesting aspect of the decision is the variability in the values of the two objectives. The more "focus" is shifted towards one of the objectives, the lower are the standard deviations in this objective and the higher are the standard deviation in the other objective. This can be clearly seen in Fig. 4. For $\lambda = 0$, the resulting plan was the same for all the considered scenarios which means Std $= 0$. It is caused by the considered random variable, which was the cost of waste treatment at the different facilities. Thus, the only objective was to use the operating decisions (transfer and treatment) with least amount of total travelled distance.

The planning decisions on the placement and capacity of the transfer stations are rather similar, ranging between 75 and 90 for the number of stations, and between $4.25 \cdot 10^6$ t to $5.06 \cdot 10^6$ t in terms of the installed capacity. This signifies that the use of the transfer stations is preferable for both objectives. There is, however, no apparent simple relationship between the optimal number/capacity of the built transfer stations and the values of λ, which indicates that the optimal decisions are rather complex (as opposed to a straightforward "higher λ means more capacity" or similar ones). Many of the optimal decision for placement of the transfer stations for different values of λ overlap, which can be seen in Fig. 5 (with $\lambda = 0$, focusing on distance), Fig. 6 (with $\lambda = 0.01$, describing a trade-off between distance and costs), and Fig. 7 (with $\lambda = 1$, focusing on costs). These places can be seen as the best potential candidates for the construction of transfer stations, regardless of the particular objective.

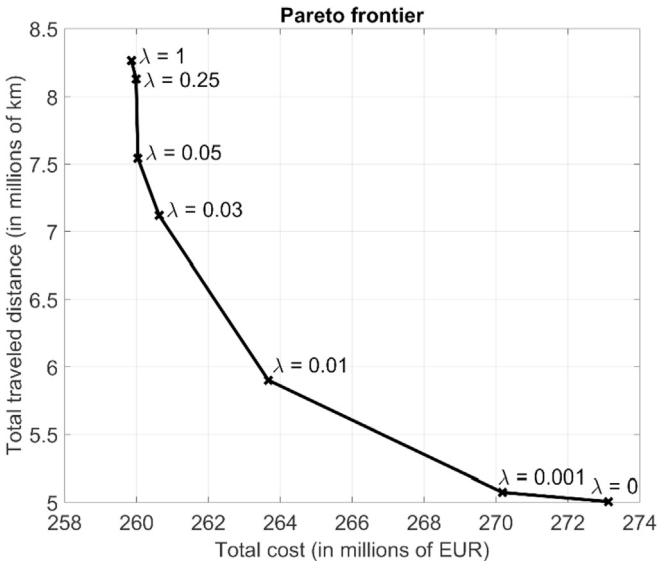


Fig. 3. Pareto frontier describing the trade-off between cost and distance for different values of λ.

The operating decision, however, depend on the value of λ to a much higher degree. As can be seen in Fig. 5, Fig. 6, Fig. 7, and, in more detail, in Fig. 8, the transport of waste from different municipalities to transfer stations and waste treatment plants naturally divides the map into (mostly unconnected) areas of influence. These areas change just slightly depending on the values of λ, where the biggest differences are caused by decision on building additional/different transfer stations, illustrated in Fig. 8. Additionally, the areas of influence also depend on the particular scenario, as the flow on both of the transport networks can vary (but the layout of transfer station network for a selected value of λ remains fixed).

The decision that depends on the desired trade-off between the overall costs and total distance travelled the most is the transport

Table 2
The results of the evaluation with 10,000 scenarios.

λ	Total cost		Total travelled distance		Stations	
	Mean (in EUR)	Std (in EUR)	Mean (in km)	Std (in km)	number built	Capacity (in t)
0	$273.13 \cdot 10^6$	$6.036 \cdot 10^6$	$5.006 \cdot 10^6$	0	90	$5.06 \cdot 10^6$
0.001	$270.18 \cdot 10^6$	$5.909 \cdot 10^6$	$5.074 \cdot 10^6$	$0.108 \cdot 10^5$	84	$5.02 \cdot 10^6$
0.01	$263.67 \cdot 10^6$	$5.468 \cdot 10^6$	$5.901 \cdot 10^6$	$2.519 \cdot 10^5$	75	$4.45 \cdot 10^6$
0.03	$260.63 \cdot 10^6$	$5.238 \cdot 10^6$	$7.119 \cdot 10^6$	$6.571 \cdot 10^5$	83	$4.25 \cdot 10^6$
0.05	$260.04 \cdot 10^6$	$5.196 \cdot 10^6$	$7.543 \cdot 10^6$	$7.533 \cdot 10^5$	81	$4.38 \cdot 10^6$
0.25	$259.99 \cdot 10^6$	$5.157 \cdot 10^6$	$8.131 \cdot 10^6$	$8.953 \cdot 10^5$	79	$4.43 \cdot 10^6$
1	$259.86 \cdot 10^6$	$5.158 \cdot 10^6$	$8.267 \cdot 10^6$	$9.084 \cdot 10^5$	79	$4.68 \cdot 10^6$

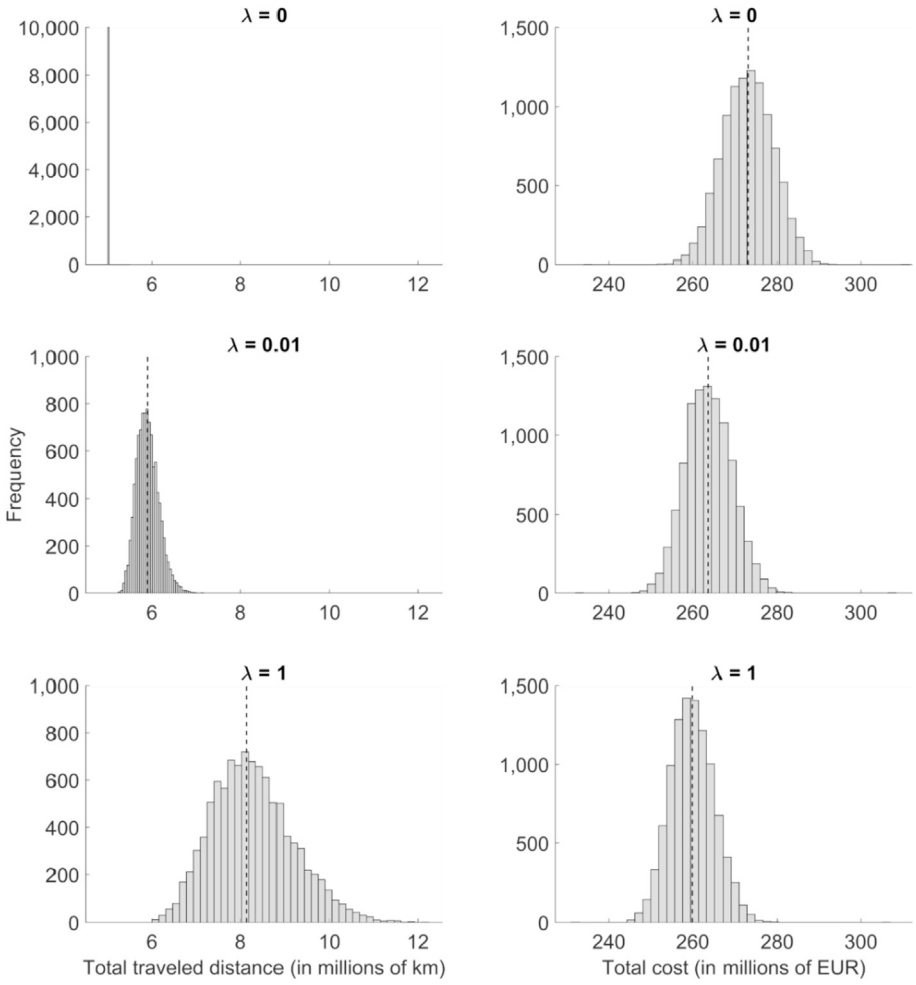


Fig. 4. Histograms of total travelled distance and total cost for different values of parameter λ. Mean values are denoted by a dashed line.

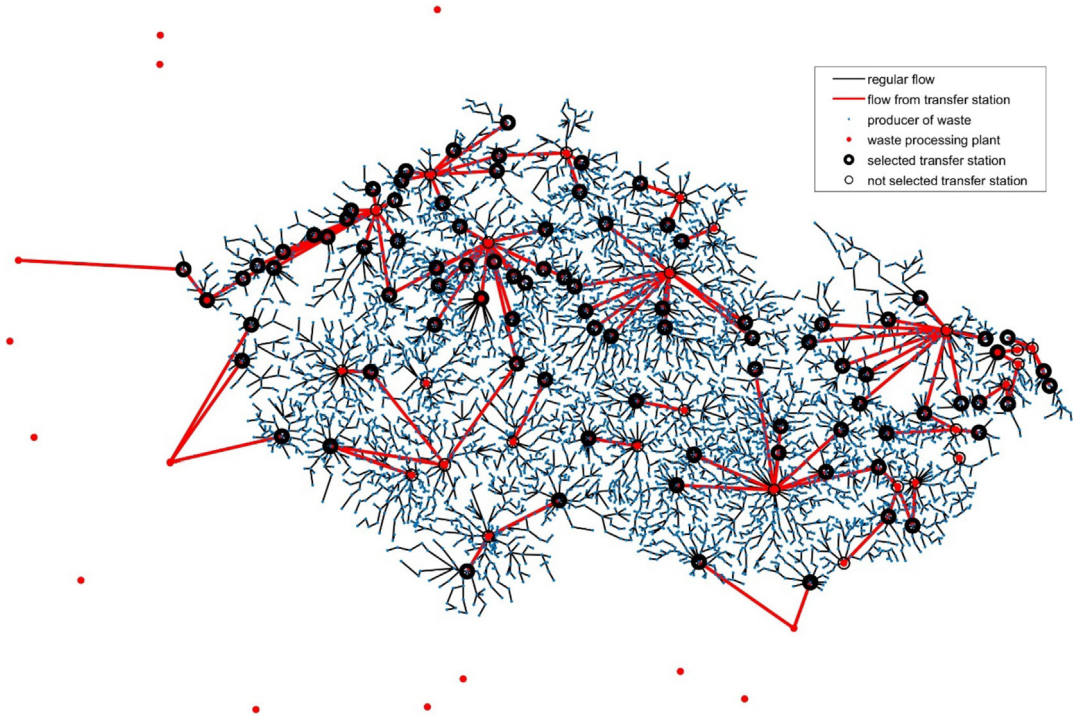


Fig. 5. A map showing the results for a baseline scenario. $\lambda = 0$.

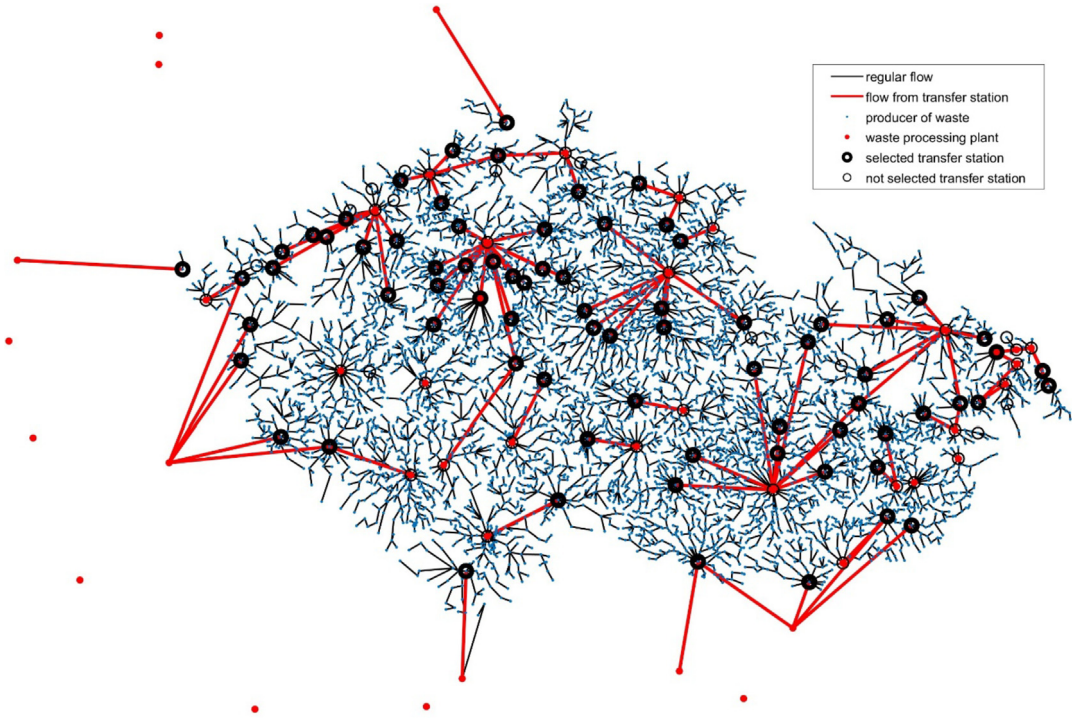


Fig. 6. A map showing the results for a baseline scenario. $\lambda = 0.01$.

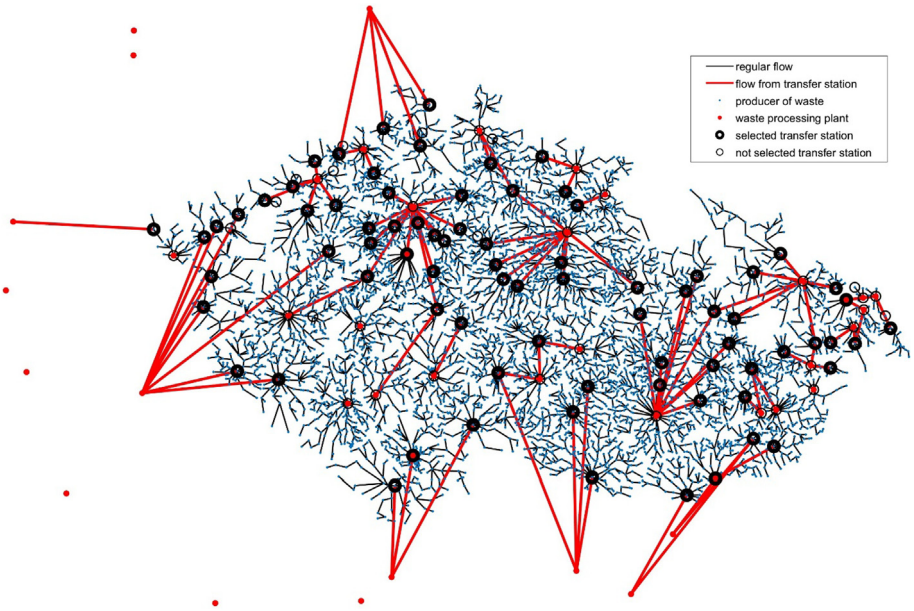


Fig. 7. A map showing the results for a baseline scenario. $\lambda = 1$.

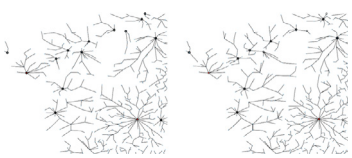


Fig. 8. A close-up on the map (western part) showing the transport only on the first network. The map naturally decomposes into areas of influence. $\lambda = 0$ (left) and $\lambda = 1$ (right).

from the selected transfer stations to the waste treatment plants. For lower values of λ (exemplified in Figs. 5 and 6), this transport is much more regionally focused, preferring the nearby waste treatment plants that have more expensive treatment costs. For higher values of λ (Fig. 7) the solution shifts towards the utilization of much more of the further placed and the foreign waste treatment plants that offer lower waste treatment costs.

The results suggest that each municipality has its preferred "transport destination" – either a transfer station or a waste treatment plant that the municipality utilized for most scenarios. In

Fig. 10, the percentage utilization of the most used transport destination for the municipalities is depicted for $\lambda = 0.01$. Over 34% of the municipalities had a singular transfer destination that did not change over the 10,000 scenarios. For 86% municipalities, the most used transfer destination was used at least 91% of the time – or, to rephrase, the claim: "This municipality uses the same transfer destination in at least 91% scenarios", was true for 86% municipalities. For 97.5% municipalities, the most used transfer destination was used at least 70% of the time. For the municipalities, this analysis can serve as a foundation for their support of the

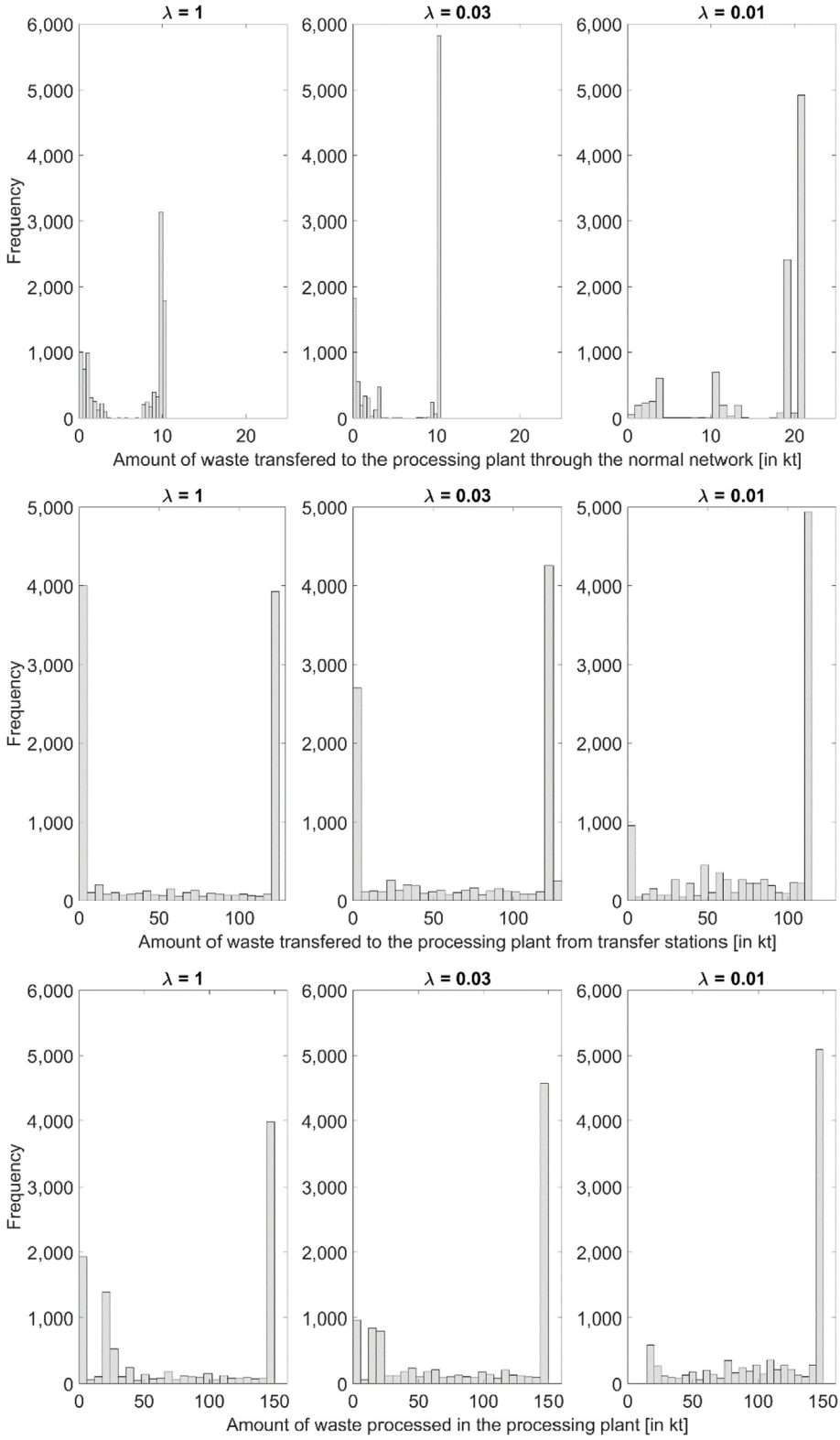


Fig. 9. Histograms of waste processing and transportation in processing plant in "Most".

plan to build a particular transfer station.

An additional perspective can be gained by analysing the effect of choosing λ on individual waste processing plants. In Fig. 9 are the histograms for the transportation and processing of waste in the processing plant in a municipality called “Most”. This municipality is the capital city of the “Most District” located in the northwest of the Czech Republic. It has approximately 67 thousand inhabitants, and the waste processing plant located there has a capacity of 150 kt. As the value of the parameter λ decreases and the optimal waste processing plan gets more locally focused, the utilization of the processing plant in “Most” increases. This can be clearly seen in the histogram of amount of processed waste – for $\lambda = 1$ the plant in “Most” is not used in almost 20% of the scenarios, while for $\lambda = 0.01$ the plant is always in use. The percentage of times the plant runs at full capacity also increases from almost 40% for $\lambda = 1$ to over 50% for $\lambda = 0.01$. However, the increase in the utilization of the plant naturally increases the traffic from both the normal network and from the transfer stations. The overall effect on the traffic situation in individual municipalities should be carefully analysed.

7. Conclusions

Decision making in an uncertain environment is always a delicate task and requires proper handling and careful consideration. In this paper, a mathematical model for an optimal transfer station grid is developed, taking into account the uncertain development in the waste processing costs, caused by the unknown future development in legislation and technology. The multi-objective nature of the model provides a ground for evaluation of the advantages and disadvantages of the trade-off between the environmental aspects and the economic viability attained by the different solutions.

A case study demonstrating the applicability and scalability of the model is presented. This study describes in high detail (considering over 6,000 municipalities) the mixed municipal waste management situation in the Czech Republic. The results show a range of viable option and strategies for the planning and managing the transfer station grid.

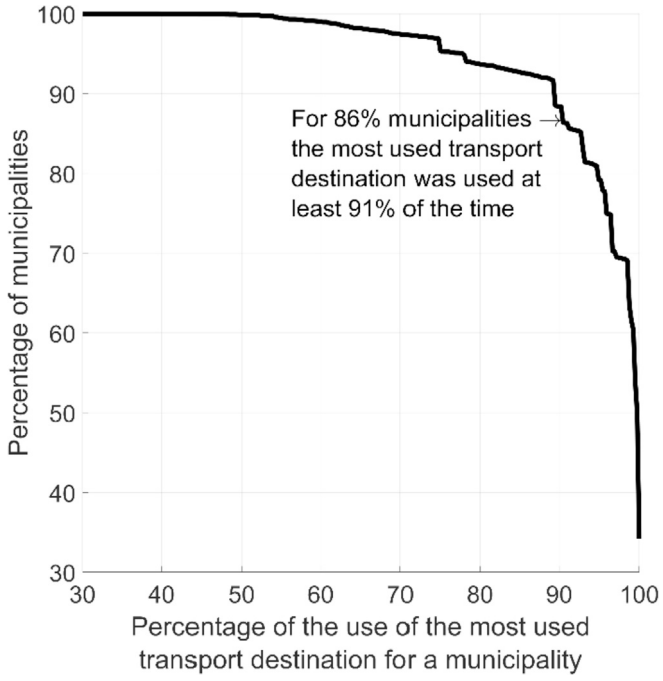


Fig. 10. Utilization of the most frequently used transport destinations (transfer stations and waste treatment facilities).$\lambda = 0.01$.

Because of the large scale of the resulting model, a suitable optimization algorithm was needed to process and solve the problem. The Benders decomposition algorithm with the utilization of lazy constraints and warm start cuts was chosen, as it is highly scalable and relatively straightforward to implement. A high-level description of the algorithm was presented.

From the macro-level perspective, the mathematical model can be used for the assessment of the optimal strategies (both tactical and operational), exemplified in Figs. 3, Fig. 4, and Fig. 10. It also provides means for the micro-level analysis of the impacts of the selected strategies on the individual municipalities and waste processing plants.

However, the most difficult task is still left for the specific decision maker (possible investors, municipalities and/or stakeholders from the field of waste management). The proposed results serve as the support and recommendation. The optimal trade-off probably lies between values 0.001 and 0.03 of the scalarization parameter λ. For λ equal to 0.03, it is possible to save 1.148 mil of travelled kilometres with only 0.77 mil EUR as extra costs. It corresponds with establishment of 83 transfer stations with total capacity 4.25 mil tones. There can be suggested locations for transfer stations, which are robust both with regards to the objective functions and uncertainties. These transfer stations can represent the first step for stakeholders in supporting sustainable waste management. The consideration of the environmental and economic aspects of the different solutions must be further examined in a much broader context. Another factor that could improve the model in the future is the addition of more decision stages, providing the possibility to plan the opening/closure of the transfer station alongside with the development of the waste processing infrastructure. Also, it is possible to merge the transfer station planning model with a model for the planning of the waste processing infrastructure, which would offer a more holistic view. However, such a model is likely to cause severe tractability/computational issues, that need to be addressed by the development of appropriate algorithms.

Acknowledgments

The authors gratefully acknowledge the financial support provided by The Ministry of Education, Youth and Sports of the Czech Republic INTER-COST project LTC18053, by the project Sustainable Process Integration Laboratory – SPIL, funded as project No. CZ.02.1.01/0.0/0.0/15_003/0000456, by Czech Republic Operational Programme Research and Development, Education, Priority 1: Strengthening capacity for quality research, and by the project “Computer Simulations for Effective Low-Emission Energy” funded as project No. CZ.02.1.01/0.0/0.0/16 026/0008392 by Operational Programme Research, Development and Education, Priority axis 1: Strengthening capacity for high-quality research, and by IGA BUT: No. FSI–S-17-4785.

References

Asefi, H., Lim, S., 2017. A novel multi-dimensional modeling approach to integrated municipal solid waste management. J. Clean. Prod. 166, 1131–1143. https://doi.org/10.1016/j.jclepro.2017.08.061.

Barbosa-Póvoa, A.P., Silva, C., Carvalho, A., 2018. Opportunities and challenges in sustainable supply chain: an operations research perspective. Eur. J. Oper. Res. 268 (2), 399–431. https://doi.org/10.1016/j.ejor.2017.10.036.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B., 2017. Julia: a fresh approach to numerical computing. SIAM Rev. 59, 65–98.

Birge, J.R., Louveaux, F., 1997. Introduction to Stochastic Programming. Springer, New York, USA, ISBN 978-0-3879-8217-5.

Bojic, S., Datkov, D., Brcanov, D., Georgievic, M., Martinov, M., 2013. Location

allocation of solid biomass power plants: case study of Vojvodina. Renew. Sustain. Energy Rev. 26, 769–775. https://doi.org/10.1016/j.rser.2013.06.039.
Boonmee, Ch, Arimura, M., Asada, T., 2018. Location and allocation optimization for integrated decisions on post-disaster waste supply chain management: on-site and off-site separation for recyclable materials. Int. J. Disaster Risk Reduct. 31, 902–917. https://doi.org/10.1016/j.ijdrr.2018.07.003.
Botello-Álvarez, J.E., Rivas-García, P., Fausto-Castro, L., Estrada-Baltazar, A., Gomez-Gonzalez, R., 2018. Informal collection, recycling and export of valuable waste as transcendent factor in the municipal solid waste management: a Latin-American reality. J. Clean. Prod. 182, 485–495. https://doi.org/10.1016/j.jclepro.2018.02.065.
Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press, UK, ISBN 978-0-521-83378-3.
Cervantes, D.E.T., Martínez, A.L., Hernández, M.C., García de Cortázar, A.L., 2018. Using indicators as a tool to evaluate municipal solid waste management: a critical review. Waste Manag. 80 (51–63) https://doi.org/10.1016/j.wasman.2018.08.046. ISSN 0956-053X.
Chen, Y.-Ch, 2018. Effects of urbanization on municipal solid waste composition. Waste Manag. 79, 828–836. https://doi.org/10.1016/j.wasman.2018.04.017.
Chen, Y.-Ch, 2018. Potential for energy recovery and greenhouse gas mitigation from municipal solid waste using a waste-to-material approach. Waste Manag. 58, 408–414. https://doi.org/10.1016/j.wasman.2016.09.007.
Coban, A., Ertis, I.F., Cavdaroglu, N.A., 2018. Municipal solid waste management via multi-criteria decision making methods: a case study in Istanbul, Turkey. J. Clean. Prod. 180, 159–167, 10 16/j.jclepro.2018.01.130.
CPLEX, 2019. IBM ILOG CPLEX Optimization Studio accessed 26 March 2019,. https://www.ibm.com/analytics/cplex-optimizer.
Directive EU, 2008/98. EC of the european parliament and of the council on waste and repealing certain directives. Off. J. Eur. Union L312, 3–30.
Directive EU, 2018/849. European Parliament and of the Council of 30 May 2018 Amending Directives 2000/53/EC on End-Of-Life Vehicles, 2006/66/EC on Batteries and Accumulators and Waste Batteries and Accumulators, and 2012/19/EU on Waste Electrical and Electronic Equipment. of the, pp. 93–99.
Directive EU, 2018/850. of the. In: European Parliament and of the Council of 30 May 2018 Amending Directive 1999/31/EC on the Landfill of Waste (Text with EEA Relevance), vols. 100–108.
Directive EU, 2018/851. European Parliament and of the Council of 30 May 2018 Amending Directive 2008/98/EC on Waste. of the, pp. 109–140.
Directive EU, 2018/852. European Parliament and of the Council of 30 May 2018 Amending Directive 94/62/EC on Packaging and Packaging Waste. of the, pp. 141–154.
Dunning, I., Huchette, J., Lubin, M., 2017. JuMP: a modeling language for mathematical optimization. SIAM Rev. 59, 295–302.
Fan, Y.V., Klemeš, J.J., Lee, ChT., Perry, S., 2018. Anaerobic digestion of municipal solid waste: energy and carbon emission footprint. J. Environ. Manag. 223, 888–897. https://doi.org/10.1016/j.jenvman.2018.07.005.
Fan, Y.V., Perry, S., Klemeš, J.J., Lee, C.T., 2018. A review on air emissions assessment: Transportation. J. Clean. Prod. 194, 673–684. https://doi.org/10.1016/j.jclepro.2018.05.151.
Ferdan, T., Šomplák, R., Zavíralová, L., Pavlas, M., Frýba, L., 2015. A waste-to-energy project: a complex approach towards the assessment of investment risks. Appl. Therm. Eng. 89, 1127–1136. https://doi.org/10.1016/j.applthermaleng.2015.04.005.
Gambella, C., Maggioni, F., Vigo, D., 2018. A stochastic programming model for a tactical solid waste management problem. Eur. J. Oper. Res. 273 (2), 684–694. https://doi.org/10.1016/j.ejor.2018.08.005.
Gharfalkar, M., Court, R., Campbell, C., Ali, Z., Hillier, G., 2015. Analysis of waste hierarchy in the European waste directive 2008/98/EC. Waste Manag. 39, 305–313. https://doi.org/10.1016/j.wasman.2015.02.007.
Gregor, J., Šomplák, R., Pavlas, M., 2017. Transportation cost as an integral part of supply chain optimisation in the field of waste management. Chem. Eng. Trans. 56, 1927–1932. https://doi.org/10.3303/CET1756322.
GUROBI, 2019. Gurobi Optimizer 7.5 accessed 26 March 2019,. http://www.gurobi.com/.
How, B.S., Tan, K.Y., Lam, H.L., 2016. Transportation decision tool for optimisation of integrated biomass flow with vehicle capacity constraints. J. Clean. Prod. 136 (10), 197–223. https://doi.org/10.1016/j.jclepro.2016.05.142.
Hsu, S.-H., 2006. NIMBY opposition and solid waste incinerator siting in democratizing Taiwan. Soc. Sci. J. 43 (3), 453–459. https://doi.org/10.1016/j.soscij.2006.04.018.
IBM. User Cuts and Lazy Constraints. Accessed 08.01.2019. https://www.ibm.com/support/knowledgecenter/en/SS9UKU_12.4.0/com.ibm.cplex.zos.help/UsrMan/topics/progr_adv/usr_cut_lazy_constr/01_uc_lc_title_synopsis.html.
Islam, T.M., Huda, N., 2018. Reverse logistics and closed-loop supply chain of Waste Electrical and Electronic Equipment (WEEE)/E-waste: a comprehensive literature review. Resour. Conserv. Recycl. 137, 48–75. https://doi.org/10.1016/j.resconrec.2018.05.026.
Ji, X., Wu, J., Zhu, Q., 2016. Eco-design of transportation in sustainable supply chain management: a DEA-like method. Transport. Res. Transport Environ. 48, 451–459. https://doi.org/10.1016/j.trd.2015.08.007.
King, A., Wallace, S.W., 2012. Modeling with Stochastic Programming. Springer, New York, ISBN 978-0-387-87816-4.
Klemeš, J.J., Varbanov, P.S., Fan, V.Y., Lam, H.L., 2017. Twenty years of PRES: past, present and future – process integration towards sustainability. Chem. Eng. Trans. 61, 1–24. https://doi.org/10.3303/CET1761001.
Kůdela, J., Popela, P., 2017. Warm-start cuts for generalized Benders decomposition. Kybernetika 53, 1012–1025. https://doi.org/10.14736/kyb-2017-6-1012.
Kůdela, J., Popela, P., Šomplák, R., Málek, M., Rychtář, A., Hrabec, D., 2017. The L-shaped method for large-scale mixed-integer waste management decision making problems. Chem. Eng. Trans. 61, 1087–1092. https://doi.org/10.3303/CET1761179.
Kůdela, J., Šomplák, R., Nevrlý, V., Lipovský, T., 2018. Robust waste transfer station planning by stochastic programming. Chem. Eng. Trans. 70, 889–894. https://doi.org/10.3303/CET1870149.
Lou, Z., Cai, B.-F., Zhu, N., Zhao, Y., Geng, Y., Yu, B., Chen, W., 2017. Greenhouse gas emission inventories from waste sector in China during 1949–2013 and its mitigation potential. J. Clean. Prod. 157, 118–124. https://doi.org/10.1016/j.jclepro.2017.04.135.
Peri, G., Ferrante, P., La Gennusa, M., Pianello, C., Rizzo, G., 2018. Greening MSW management systems by saving footprint: the contribution of the waste transportation. J. Environ. Manag. 219, 74–83. https://doi.org/10.1016/j.jenvman.2018.04.098.
Putna, O., Janošťák, F., Šomplák, R., Pavlas, M., 2018. Demand modelling in district heating systems within the conceptual design of a waste-to-energy plant. Energy 163, 1125–1139. https://doi.org/10.1016/j.energy.2018.08.059.
Rudi, A., Müller, A.K., Fröhling, M., Schultmann, F., 2017. Biomass value chain design: a case study of the upper rhine region. Waste Biomass Valorizat. 8 (7), 2313–2327. https://doi.org/10.1007/s12649-016-9820-x.
Sarker, B.R., Wu, B., Paudel, K.P., 2018. Optimal number and location of storage hubs and biogas production reactors in farmlands with allocation of multiple feedstocks. Appl. Math. Model. 55, 447–465. https://doi.org/10.1016/j.apm.2017.11.010.
Šomplák, R., Pavlas, M., Kropáč, J., Putna, O., Procházka, V., 2014. Logistic model based tool for policy-making towards sustainable waste management. Clean Technol. Environ. Policy 16, 1275–1286. https://doi.org/10.1007/s10098-014-0744-5.
Šomplák, R., Nevrlý, V., Málek, M., Pavlas, M., Klemeš, J.J., 2017. Network flow based model applied to sources, sinks and optimal transport of combustible waste. Chem. Eng. Trans. 61, 991–996. https://doi.org/10.3303/CET1761163.
Tian, X., Wu, Y., Qu, S., Liang, S., Chen, W., Xu, M., Zuo, T., 2018. Deriving hazardous material flow networks: a case study of lead in China. J. Clean. Prod. 199, 391–399. https://doi.org/10.1016/j.jclepro.2018.07.132.
Tomić, T., Dominković, D.F., Pfeifer, A., Schneider, D.R., Pedersen, A.S., Duić, N., 2017. Waste to energy plant operation under the influence of market and legislation conditioned changes. Energy 137, 1119–1129. https://doi.org/10.1016/j.energy.2017.04.080.
Walmsley, M.R.W., Walmsley, T.G., Atkins, M.J., Kamp, P.J.J., Neale, J.R., Chand, A., 2015. Carbon Emissions Pinch Analysis for emissions reductions in the New Zealand transport sector through to 2050. Energy 92, 569–576. https://doi.org/10.1016/j.energy.2015.04.069.
Wichapa, N., Khokhajaikiat, P., 2017. Solving multi-objective facility location problem using the fuzzy analytical hierarchy process and goal programming: a case study on infectious waste disposal centers. Operat. Res. Perspect. 4, 39–48. https://doi.org/10.1016/j.orp.2017.03.002.
Williams, H.P., 2013. Model Building in Mathematical Programming, fifth ed. John Wiley & Sons, UK, ISBN 978-1-118-44333-0.
World Bank Open Data. Accessed date 8 October 2018. https://data.worldbank.org/.
Yadav, V., Karmakar, S., Dikshit, A.K., Vanjari, S., 2016. A feasibility study for the locations of waste transfer stations in urban centers: a case study on the city of Nashik, India. J. Clean. Prod. 126, 191–205. https://doi.org/10.1016/j.jclepro.2016.03.017.
Zhao, J., Huang, L., Lee, D.-H., Peng, Q., 2016. Improved approaches to the network design problem in regional hazardous waste management systems. Transport. Res. E Logist. Transport. Rev. 88, 52–75. https://doi.org/10.1016/j.tre.2016.02.002.

A13

The Quadratic Assignment Problem: Metaheuristic Optimization Using HC12 Algorithm

Radomil Matousek
Brno University of Technology
Institute of Automation and Computer Science
rmatousek@vutbr.cz

Ladislav Dobrovsky
Brno University of Technology
Institute of Automation and Computer Science
dobrovsky@fme.vutbr.cz

Jakub Kudela
Brno University of Technology
Institute of Automation and Computer Science
Jakub.Kudela@vutbr.cz

ABSTRACT

The Quadratic Assignment Problem (QAP) is a classical NP-hard combinatorial optimization problem. In the paper will be presented suitable metaheuristic algorithm HC12. The algorithm is population based and uses a massive parallel search of the binary space which represents the solution space of the QAP. The presented implementation of the metaheuristic HC12 utilizes the latest GPU CUDA platform. The results are presented on standard test problems from the QAP library.*

CCS CONCEPTS

• **Design and analysis of algorithms** → Approximation algorithms analysis, Parallel algorithms • **Mathematical optimization** → Optimization with randomized heuristics

KEYWORDS

Quadratic assignment problem, Massively parallel algorithm

ACM Reference format:

R. Matousek, L. Dobrovsky, and J. Kudela. 2019. The Quadratic Assignment Problem: Metaheuristic Optimization Using HC12 Algorithm. In *Companion Proceedings of ACM GECCO conference, Prague, Czech Republic, July 2019 (GECCO'19),* 2 pages.
DOI: 10.1145/3319619.3322088

1 INTRODUCTION

The NP-hard quadratic assignment problem (QAP), in its Koopmans and Beckmann form [1], can be described as follows: The problem is structured on a complete directed graph with n nodes and n^2 arcs, together with a set of n facilities, that have to be assigned to the nodes. The indices i, j correspond to the nodes, the indices f, g correspond to the facilities, $b_{i,j} \geq 0$ is a given (directed) distance from node i to node j, $a_{f,g} \geq 0$ is a flow from facility f to facility g, and $c_{i,j}$ is a cost of assigning facility f to node i. By using binary variables $x_{i,f} = 1$ if facility f is assigned to node i, and 0 otherwise, the QAP can be stated as the following 0-1 optimization problem:

$$min \quad \sum_i \sum_f \sum_j \sum_g a_{f,g}\, b_{i,j} x_{i,f} x_{j,g} + \sum_i \sum_f c_{i,f}\, x_{i,f} \qquad (1)$$

$$s.t. \quad \sum_i x_{i,f} = 1, \quad \sum_f x_{i,f} = 1, \qquad \forall f, \ \forall i \qquad (2)$$

$$x_{i,f} \in \{0,1\}, \qquad \forall i, \forall f. \qquad (3)$$

Several directions for enriching the QAP formulation have been proposed – among the most notable of these are the multi-objective formulation [2] and stochastic formulation [3].

2 ALGORITHM HC12

The binary HC12 algorithm [4] is a stochastic heuristic searching algorithm which belongs to the class of pseudo global search methods. The basic step of the algorithm is a generation of a neighborhood of the current solution, which serves as a base for the new population. The method of generating the neighborhood is the pivotal characteristic of HC12. The paradigm of the algorithm is the search of the optimal solution in the binary (Hamming) space, that encodes the solution. In this context, it is a parallel approach to genetic algorithms, where the solution is encoded as a binary vector. The best individual of the $i-$th generation (or iteration) is chosen as the base for the following $(i+1)$ generation. The approach is depicted in Fig. 1.

3 RESULTS AND DISCUSSION

The HC12 algorithm is extremely suitable for parallel implementation. In the presented experiments, it was implemented for HPC computations on NVIDIA RTX 2080 (8GB). Even the larger memory requirements of the QAP problems, not more than 6GB were used. The implementation searches for the best solution in multiple runs (restarts of the algorithm). The effectivity of the algorithm (in regard to the number of found optimal solutions) can be determined as a success rate (the ration of runs that ended in an optimal solution).

GECCO'19, July 13-17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3322088

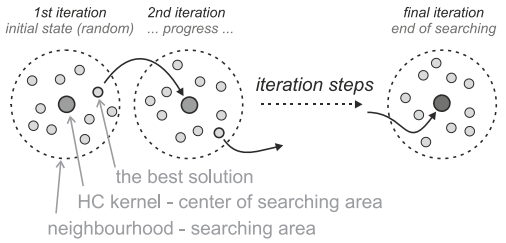


Figure 1: The scheme of HC12 iterations.

A rather interesting insight is provided by the dependence of the number of used swaps on the number of found optimal solutions. More swaps also result in a higher computation time. There appear to be "optimal" number of swaps for the given problem (the number of swaps that results in the most successful runs).

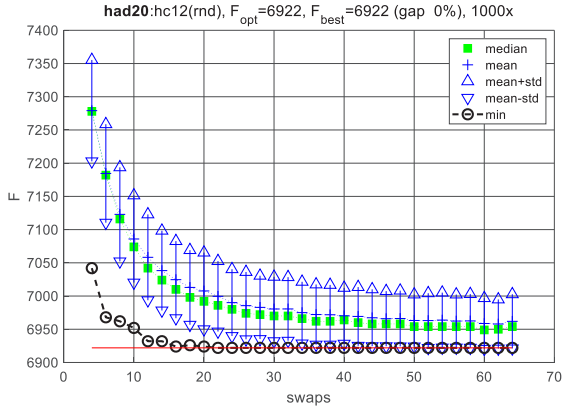


Figure 2: An influence of swap operator to convergence features of "had20" test problem.

The computational comparison of HC12 with other state-of-the-art metaheuristics is done on the standard test problems from the QAPLIB library [5].

The selected metaheuristics are the hybrid teaching-learning optimization implemented on a cluster [6], the parallel implementation of hybrid algorithms [7,8], and the bee algorithm implemented on a CUDA platform [9]. The results of the computation and the comparison are reported in Table 1.

Although the running times of the HC12 algorithm are extremely fast (compare to the other heuristics), the robustness of the resulting solutions is still rather low and requires additional research and tuning.

ACKNOWLEDGMENTS

This work was also supported by The Ministry of Education, Youth and Sports of the Czech Republic, INTER-COST project LTC18053 and European COST Action CA15140 and by IGA BUT: No. FSI-S-17-4785.

REFERENCES

[1] T.C. Koopmans and M.J. Beckmann. 1957. Assignment problems and the location of economic activities. *Econometrica* 25, 1 (1957), 53–76.
[2] C. Sanhueza, F. Jimenez, R. Berretta, and P. Moscato. 2017. PasMoQAP: A parallel asynchronous memetic algorithm for solving the Multi-Objective Quadratic Assignment Problem. In: *2017 IEEE Congress on Evolutionary Computation (CEC).* San Sebastian, Spain, 17013939.
[3] R. Matousek, P. Popela, and J. Kudela. 2017. Heuristic approaches to stochastic quadratic assignment problem: VaR and CVaR cases. *MENDEL Journal Series* 23 (2017), 73–78.
[4] R. Matousek and E. Zampachova. 2011. Promising GAHC and HC12 algorithms in global optimization tasks. *Optimization Methods and Software* 26, 3 (2011), 405–419.
[5] R.E. Burkard, S. Karisch, and F. Rendl. 1991. QAPLIB–A quadratic assignment problem library. *Eur. J. Oper. Res*. 55, 1 (1991), 115–119.
[6] T. Dokeroglu. 2015 Hybrid teaching–learning-based optimization algorithms for the Quadratic Assignment Problem. *Computers & Industrial Design* 85 (2015), 86–101.
[7] U. Tosun. 2015. On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering Applications of Artificial Intelligence* 39 (2015), 267–278.
[8] U. Tosun, T. Dokeroglu, and A. Cosar. A robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem. *International Journal of Production Research* 51, 14 (2013), 4117–4133.
[9] W. Chmiel and P. Szwed. 2015. Bees Algorithm for the Quadratic Assignment Problem on CUDA Platform. In: *Man–Machine Interactions 4*, Gruca A., Brachman A., Kozielski S., Czachorski T. (Eds). Advances in Intelligent Systems and Computing, vol 391, pp 615–625. Springer, Cham.

Table 1: Comparison with other results

problem instance	optimal solution	HC12 (GPU implementation)			[6]		[7]		[8]		[9]
		swaps	success*	time [s]	APD	time	APD	time	APD	time	APD
esc16a	68	62	1	0.0014	0	6	0	151.8	0	702	0
esc32a	130	52	0.001	5.9462	0	72	-	-	-	-	10.77
had16	3720	64	0.348	0.0288	0	6	0	149.4	0	594	0
had18	5358	64	0.123	0.1476	0	12	0	183.6	0	618	-
had20	6922	62	0.067	0.3072	0	18	0	223.8	0	600	-
rou12	235528	60	0.116	0.0338	0	6	0	87.6	0	90	-
rou15	354210	50	0.02	0.2378	0	6	0	133.8	0	600	-
rou20	725522	44	0.001	8.4109	0	18	-	-	-	-	0.25

* success: The effectivity of the algorithm (in regard to the number of found optimal solutions) is determined as ratio of number of optimal solutions to number of algorithms runs.

A14

International Journal of Industrial Engineering Computations 13 (2022) 151–164

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

How to start a heuristic? Utilizing lower bounds for solving the quadratic assignment problem

Radomil Matousek[a], Ladislav Dobrovsky[a] and Jakub Kudela[a*]

[a]*Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Brno, Czech Republic*

C H R O N I C L E

Article history:
Received July 30 2021
Received in Revised Format September 21 2021
Accepted December 31 2021
Available online December, 31 2021

Keywords:
Heuristics
Lower bounds
Metaheuristics
Quadratic assignment problem
Starting values

A B S T R A C T

The Quadratic Assignment Problem (QAP) is one of the classical combinatorial optimization problems and is known for its diverse applications. The QAP is an NP-hard optimization problem which attracts the use of heuristic or metaheuristic algorithms that can find quality solutions in an acceptable computation time. On the other hand, there is quite a broad spectrum of mathematical programming techniques that were developed for finding the lower bounds for the QAP. This paper presents a fusion of the two approaches whereby the solutions from the computations of the lower bounds are used as the starting points for a metaheuristic, called HC12, which is implemented on a GPU CUDA platform. We perform extensive computational experiments that demonstrate that the use of these lower bounding techniques for the construction of the starting points has a significant impact on the quality of the resulting solutions.

1. Introduction

The NP-hard quadratic assignment problem (QAP) in its Koopmans and Beckmann form (Koopmans & Beckmann, 1957), which is notoriously difficult in practice, can be described as follows (Cela, 2013): The problem is structured on a complete directed graph $G = (V, A)$ with n nodes and n^2 arcs, together with a set of n facilities that have to be assigned to the nodes. The indices $i, j \in V$ correspond to the nodes, the indices $f, g \in N = \{1, \ldots, n\}$ correspond to the facilities, $b_{i,j} \geq 0$ is a given (directed) distance from node i to node j, $a_{f,g} \geq 0$ is a given flow from facility f to facility g. By using binary variables $x_{i,f} = 1$ if facility f is assigned to node i, and 0 otherwise, the QAP can be stated as the following quadratic 0-1 optimization problem:

$$\text{min} \quad \sum_{i \in V} \sum_{f \in N} \sum_{j \in V} \sum_{g \in N} a_{f,g} b_{i,j} x_{i\,f} x_{j,g} \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in V} x_{i,f} = 1 \quad \forall f \in N \tag{2}$$

$$\sum_{f \in N} x_{i,f} = 1 \quad \forall i \in V \tag{3}$$

$$x_{i\,f} \in \{0\ , 1\} \quad \forall i \in V\ \forall f \in N, \tag{4}$$

It is quite well known that the constraint matrix, defined by (2)-(3) is totally unimodular, implying that the optimization of any linear objective function over the QAP feasible set is just a relatively easy linear programming problem, known as the linear assignment problem (LAP) (Burkard et al., 2012).

* Corresponding author
E-mail: Jakub.Kudela@vutbr.cz (J. Kudela)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.12.003

Despite its rather simple definition, QAP is among the most difficult optimization problems that arise in practice – campus planning problem (Dickey & Hopkins, 1972), backboard wiring problem (Steinberg, 1961), hospital layout problem (Helber et al., 2016), airport gate assignment (Haghani & Chen, 1998), turbine runner in electricity generation (Laporte & Mercure, 1988), statistical analysis (Hubert & Schultz, 1976), and optimal placing of letters on touchscreen devices (Dell'Amico et al., 2009) have all been modeled as a QAP. There are several other well-known combinatorial optimization problems which can be formulated as QAPs with specific coefficient matrices (Cela, 2013) – e.g., the travelling salesman problem, graph partitioning and maximum clique, the linear arrangement problem, and packing problems in graphs, to name a few. An intriguing feature of the QAP is that even for some problems of size $n \leq 50$, such as sko42 or tai30a from the QAPLIB problem library (Burkard et al., 1997), the optimal solution is still not confirmed. Even finding an ε-optimal solution is a difficult problem. There are, however, several QAP instances/structures for which the optimal solution is attainable in polynomial time (Cela et al., 2018) or which were generated in such a way that the optimal solution is known (Li & Pardalos, 1992). Furthermore, several directions for enriching the QAP formulation have been proposed – among the most notable of these are the multi-objective formulation (Samanta et al., 2018; Sanhueza et al., 2017) and stochastic programming formulation (Popela et al., 2016; Matousek et al., 2017). For these reasons, the study of the QAP attracted quite a large amount of research from both mathematical programming and heuristics communities.

In this paper, we show that the approaches from these two communities can be successfully combined. We will utilize the lower-bounding techniques for the construction of advantageous starting points for a hill climbing metaheuristic and show on extensive computational experiments that starting the heuristic at these points yields significant improvements over the usual random starting points.

The remained of the article is structured as follows: In Section 2, the state-of-the-art in both mathematical programming and heuristics approaches to the QAP is reviewed. In Section 3 a suitable metaheuristic algorithm HC12 is described. Section 4 provides computational comparison of the mathematical programming approaches, and the results obtain from HC12. Conclusions and future research direction are summarized in Section 5.

2. Methods for approaching QAP

2.1. (Meta)Heuristics

Because of the computational difficulty of the QAP, myriads of heuristics were proposed to tackle this combinatorial problem. Among the first ones were simulated annealing (Burkard & Rendl, 1984), robust tabu search (Taillard, 1991) and genetic hybrids (Fleurent & Ferland, 1994) – although these are no longer the most efficient methods, they were able to find the best known solutions for some of the QAPLIB instances, that are yet to be beaten or proven optimal. The comparison between tabu search and simulated annealing based on a size of the QAP was conducted in (Hussin & Stützle, 2014). A local search heuristic called breakout local search enhanced by a Levenshtein Distance metric for checking solutions for similarity was described in (Aksan et al., 2017). The state-of-the-art in metaheuristics for the QAP includes population based memetic algorithms (Benlic & Hao, 2015), genetic algorithms (Ahmed, 2015), differential evolution (Hameed et al., 2020) and particle swarm algorithms (Hafiz & Abdennour, 2016). Hybrid algorithms, combining several heuristics and metaheuristics are also very prevalent. A hybrid teaching-learning based algorithm integrating tabu search within a swarm intelligence metaheuristic was described in (Dokeroglu, 2015). A memetic algorithm that uses a ternary tree structure for its population and the tabu search algorithm, which runs simultaneously, for its local search mechanism was proposed in (Harris et al., 2015). A parallel hybrid algorithm with three phases was proposed by (Tosun, 2015) – this algorithm initially benefits from a genetic algorithm to obtain a high-quality initial seed on which a diversification mechanism is run. Finally, this modified solution is used for a robust tabu search to find a near-optimal result. In (Abdel-Basset et al., 2018) the authors describe an algorithm integrating the whale optimization algorithm with a tabu search. A multistart hyper-heuristic algorithm on the grid is proposed in (Dokeroglu & Cosar, 2016) – it makes use of different metaheuristics (simulated annealing, robust tabu search, ant colony optimization, and breakout local search) and reports computations on a high-performance cluster with 368 cores and 736 GB of RAM.

Since it offers speed-up opportunity that can outperform current multicore processors, (Tsutsui & Fujimoto, 2009) applied Graphics Processing Unit (GPU) computation with compute unified device architecture (CUDA) to solve the QAP. In (Czapiński, 2013) is proposed a Parallel Multistart Tabu Search (PMTS) algorithm. It is implemented on a highly powerful GPU hardware intended for high-performance computing with the CUDA platform. Therefore, PMTS is shown to perform competitively with a single-core or a parallel CPU implementation on a high-end six-core CPU. Another GPU based algorithm is described in (Mohammadi et al., 2015) – a parallel genetic algorithm, that (as the authors report) can run up to 30 times faster than its serial counterpart. Finally, the bees algorithm implemented on the CUDA platform is proposed in (Chmiel & Szwed, 2016).

2.2. MIP Reformulations

One common mathematical programming approach for solving the QAP is to "linearize" it, that is, reformulate it as a pure or mixed integer linear programming problem. This was first done in (Gilmore, 1962) by replacing the terms $x_{i,f}x_{j,g}$ in the

objective function by n^4 variables $y_{i,f,j,g} = x_{i,f}x_{j,g}$. This reformulation was further improved upon in (Adams & Johnson, 1994), calling it the level-1 reformulation-linearization technique (RLT-1). The reformulated problem than has the following form:

$$\min \quad \sum_{i\in V}\sum_{f\in N}\sum_{j\in V}\sum_{g\in N} a_{f,g}b_{i,j}y_{i,f,j,g} \tag{5}$$

$$\text{s.t.} \quad \sum_{i\in V} y_{i,f,j,g} = x_{j,g} \quad \forall j \in V \ \forall f,g \in N \tag{6}$$

$$\sum_{f\in N} y_{i,f,j,g} = x_{j\,g} \quad \forall i \ j \in V \ \forall g \in N \tag{7}$$

$$y_{i,f,j,g} = y_{j\,g\,i\,f} \geq 0 \quad \forall i \ j \in V \ \forall f \ \ g \in N \tag{8}$$

$$\sum_{i\in V} x_{i,f} = 1 \quad \forall f \in N \tag{9}$$

$$\sum_{f\in N} x_{i,f} = 1 \quad \forall i \in V \tag{10}$$

$$x_{i\,f} \in \{0,1\} \quad \forall i \in V \ \forall f \in N, \tag{11}$$

By relaxing the binary constraint (11) (using a LP relaxation), the above formulation can be used to obtain a valid lower bound on the QAP (1)-(4). The RLT-1 reformulation was further strengthened by introducing additional n^6 variables, called RLT-2 in (Adams et al., 2007), and even further with additional n^8 variables, called RLT-3 in (Hahn et al., 2012), which for the time being is still too large even for modern day computers – for problems of size $n = 25$, the computations needed to be done on a server with 384 GB of RAM. A different mixed integer linearization scheme, called the Kaufman-Broeckx formulation, was proposed in (Kaufman & Broeckx, 1978) with $O(n^2)$ additional variables. Although this is the smallest QAP linearization, its LP relaxation is known to be usually weak. This relaxation was tightened in (Xia & Yuan, 2006) using the Gilmore-Lawler bound (GLB) (Gilmore, 1962; Lawler, 1963) and further enhanced in (Zhang et al., 2013). A formulation based in the Kaufman-Broeckx family was used in (Fischetti et al., 2012) to solve (prove optimality) all the esc instances (Eschermann & Wunderlich, 1990) (including the one of size $n = 128$).

2.3. Lower Bounding Techniques

Exact solution of a QAP typically requires the use of a branch-and-bound framework (Anstreicher, 2003). In practice, the lack of efficiently computable, tight lower bounds for the QAP has been the key factor in the problem's difficulty, as the tighter the bound is, the more difficult it generally is to compute. There are various approaches for obtaining lower bounds. One of the oldest methods, the Gilmore-Lawler bound (GLB), is still widely used. A comparison of older bounds based on linearization of the QAP can be found in (Karisch et al., 1999). A great success in solving previously unsolved QAP instances was achieved using the convex quadratic programming bound introduced in (Anstreicher & Brixius, 2001).

A seminal breaking point in combinatorial optimization was the emergence of semidefinite programming (SDP). The SDP bounds for the QAP were first studied in (Zhao et al., 1998). The problem with this relaxation was that it involved a matrix variable of order n^2, and can therefore only be solved efficiently by interior point methods for, say, $n \leq 20$. This limitation has encouraged research into exploiting group symmetry of the QAP data matrices to obtain smaller and more tractable SDP problems (de Klerk & Sotirov, 2012). It has also prompted recent research into SDP relaxations of QAP where the matrix variables are of order n; see (Peng et al., 2010) and (Peng et al., 2015). In both these lines of research the authors were able to compute the best-known lower bounds for some QAPLIB instances. As we will use the lower bounding techniques for the construction of starting points for our metaheuristic, we will describe these techniques in greater detail. The computation of the GLB can be done in the following way: Denote the row vectors of matrices A and B by a_i and $b_i, i = 1,2,\dots,n,$. Let $\hat{a}_i$ be the vector consisting of the $(n-1)$ components of a_i, without $a_{i,i}$, and let $\hat{b}_i$ be the vector consisting of the $(n-1)$ components of b_i, without $b_{i,i}$. Define a matrix $L = (l_{i,j})$ as follows:

$$l_{ij} = \langle \hat{a}_i, \hat{b}_j \rangle_-, \quad i,j = 1,2,\dots,n,$$

where $\langle a,b\rangle_-$ is the minimal scalar product, which can be computed by ordering the vector a nondecreasingly and b nonincreasingly. The GLB is given by the optimal value of the n-dimensional LAP with cost matrix $l_{i,j} + a_{i,i}b_{j,j}$:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}(l_{i\,,j} + a_{i,\,i}b_{j,\,j})x_{i\,j} \tag{12}$$

$$\text{s.t.} \quad \sum_{i\in V} x_{i,\,f} = 1 \quad \forall f \in N \tag{13}$$

$$\sum_{f \in N} x_{i,f} = 1 \quad \forall i \in V \tag{14}$$

$$x_{i\,f} \in \{0,\ 1\} \quad \forall i \in V\ \forall f \in N \tag{15}$$

which requires only $O(n^3)$ computational time.

The second type of the QAP lower bounds are the eigenvalue bounds. These use the fact that the set of permutation matrices Π_n can be characterized as:

$$\Pi_n = \mathcal{Q}_n \cap \mathcal{E}_n \cap \mathcal{N}_n,$$

where $\mathcal{Q}_n$ is the set of orthogonal matrices, $\mathcal{E}_n$ is the set of doubly stochastic matrices and $\mathcal{N}_n$ is the set of matrices with positive elements of size $n \times n$.

The QAP can then be equivalently formulated as:

$$\min_{X \in \Pi_n} tr\,(AXBX^T),$$

where tr(·) is the trace of a matrix. The first eigenvalue bound that uses this QAP formulation was introduced in (Hoffman & Wielandt, 2003) and is based on the relaxation of the feasible region:

$$\min_{X \in \mathcal{Q}_n} \mathrm{tr}\,(AXBX^T) = \langle \lambda(A), \lambda(B) \rangle_-,$$

where $\lambda(\cdot)$ denotes the vector of eigenvalues of the matrix. This bound can be computed with very little effort but tends to be extremely weak. The improvement of this bound was done in (Hadley et al., 1992) and is called the Hadley-Rendl-Wolkowitz (HRW): Let u_n be a vector of all ones and let V be an $n \times (n-1)$ matrix with $V^T u_n = 0$ and $\mathrm{rank}(V) = n - 1$. Then

$$\{X \in \mathcal{R}^{n \times n} : Xu_n = X^T u_n = u_n\} = \{\frac{1}{n} u_n u_n^T + VMV^T : M \in \mathcal{R}^{(n-1)\times(n-1)}\}$$

which can be used to reparametrize the trace formulation as:

$$\mathrm{tr}(AXBX^T) = tr\left((V^TAV)\hat{X}(V^TBV)\hat{X}^T\right) + \frac{2}{n}\mathrm{tr}(AJ_nB)X^T - \mathrm{const}$$

where $J_n = u_n u_n^T$ is a matrix of all ones, and use the eigenvalue bound to obtain the improved HRW bound:

$$\langle \lambda(V^TAV)\ \lambda(V^TBV) \rangle_- + \mathrm{LAP}\left(\frac{2}{n} AJ_nB\right) - \mathrm{const} \tag{16}$$

The third type of the QAP lower bounds are based on a convex quadratic programming relaxation of the trace reparameterization shown above. (Anstreicher & Brixius, 2001) used the above-mentioned parametrization and showed that the following convex quadratic optimization problem gives a lower bound on the QAP:

$$\min \quad \mathrm{vec}(X)^T Q \mathrm{vec}(X) + \langle \lambda(V^TAV)\ \lambda(V^TBV) \rangle_- \tag{17}$$

$$\text{s.t.} \quad Xu_n = X^T u_n = u_n \quad X \geq 0, \tag{18}$$

where

$$Q = (B \otimes A) - \left(I \otimes V\hat{S}V^T\right) - \left(V\hat{T}V^T \otimes I\right)$$

and $\hat{S}$ and $\hat{T}$ can be obtained from the spectral decomposition of V^TAV and V^TBV. The last type of the QAP lower bounds we consider are based on a SDP relaxation developed by (Peng et al., 2015): Let (B_1, B_2) be a minimal trace matrix splitting of the matrix B and compute a decomposition $B_i = \hat{B}_i^T \hat{B}_i$. Let $B_s = B_1 + B_2$ the SDP relaxation model of QAP based on minimal trace matrix splitting is the following:

$$\min \quad \mathrm{tr}(\mathrm{AY}) \tag{19}$$

$$\text{s.t.} \quad Y = Y_1 - Y_2 \quad Y_s = Y_1 + Y_2 \tag{20}$$

$$\begin{pmatrix} I & \hat{B}_1 X^T \\ X\hat{B}_1 & Y_1 \end{pmatrix} \geq 0 \begin{pmatrix} I & \hat{B}_2 X^T \\ X\hat{B}_2 & Y_2 \end{pmatrix} \geq 0 \tag{21}$$

$$\mathrm{diag}(Y_1) = X\mathrm{diag}(B_1) \quad Y_1 e = XB_1 e \tag{22}$$

$$\mathrm{diag}(Y_2) = X\mathrm{diag}(B_2) \quad Y_2 e = XB_2 e \tag{23}$$

$$\left(X \min\left([B_1]_{off}\right)\right)_i \leq [Y_1]_{ij} \quad \forall i \neq j \tag{24}$$

$$[Y_1]_{ij} \leq \left(X \max\left([B_1]_{off}\right)\right)_i \quad \forall i \neq j \tag{25}$$

$$\left(X \min\left([B_2]_{off}\right)\right)_i \leq [Y_2]_{ij} \quad \forall i \neq j \tag{26}$$

$$[Y_2]_{i\,j} \le \left(X \max\left([B_2]_{off}\right)\right)_i \quad \forall i \ne j \tag{27}$$

$$\left(X \min\left([B]_{off}\right)\right)_i \le [Y]_{i\,j} \quad \forall i \ne j \tag{28}$$

$$[Y]_{i\,j} \le \left(X \max\left([B]_{off}\right)\right)_i \quad \forall i \ne j \tag{29}$$

$$\left(X \min\left([B_s]_{off}\right)\right)_i \le [Y_s]_{i\,j} \quad \forall i \ne j \tag{30}$$

$$[Y_s]_{i\,j} \le \left(X \max\left([B_s]_{off}\right)\right)_i \quad \forall i \ne j \tag{31}$$

$$\|[Y_1]_{i:}\|_2 \le X \,\|[B_1]_{i:}\|_2 \quad \forall i \tag{32}$$

$$\|[Y_2]_{i:}\|_2 \le X \,\|[B_2]_{i:}\|_2 \quad \forall i \tag{33}$$

$$\|[Y]_{i:}\|_2 \le X \,\|[B]_{i:}\|_2 \quad \forall i \tag{34}$$

$$\|[Y_s]_{i:}\|_2 \le X \,\|[B_s]_{i:}\|_2 \quad \forall i \tag{35}$$

$$X \ge 0 \;\; Xe = X^T e = e \tag{36}$$

B_{off} denotes the matrix consisting of all the off-diagonal elements of B, i.e., $B_{off} = B - \mathrm{diag}(b_{11}, b_{22}, \dots, b_{nn})$, $\max(B)$ (or $\min(B)$)denotes the vector whose i-th component is the maximal element (or minimal element) in the i-th row (denoted by $B_{i,:}$) of B.

In order to obtain a starting point (in our case, a starting permutation) for the upcoming heuristic, we use a projection of the matrix obtained from the lower bounding schemes to the space of permutation matrices: Let $\hat{X}$ be a matrix obtained from the computation of the lower bounds. The "closest" permutation matrix X to $\hat{X}$ can be computed by solving the following problem:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(X_{i,j} - \hat{X}_{i,j}\right)^2 \tag{37}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} X_{i,\,j} = 1 \;\; \sum_{i=1}^{n} X_{i,\,j} = 1 \quad \forall i \;\forall j \tag{38}$$

$$X_{i\,j} \in \{0\ ,1\} \quad \forall i \;\forall j, \tag{39}$$

The problem above can be reformulated into an equivalent problem using the fact that $X_{i,j}$ is binary:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\left(1 - 2\hat{X}_{i,\,j}\right)X_{i,\,j} \tag{40}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} X_{i,\,j} = 1 \;\; \sum_{i=1}^{n} X_{i,j} = 1 \quad \forall i \;\forall j \tag{41}$$

$$X_{i\,j} \in \{0,1\} \quad \forall i \;\forall j, \tag{42}$$

which is a simple LAP.

2.4 HC12 algorithm

The binary HC12 algorithm, described in detail in (Matousek & Zampachova, 2011), is a stochastic heuristic searching algorithm which belongs to the class of pseudo global search methods. The basic step of the algorithm is a generation of a neighborhood of the current solution, which serves as a base for the construction of a new (improved) population. The method of generating the neighborhood is the pivotal characteristic of HC12. The paradigm of the algorithm is the search of the optimal solution in the binary (Hamming) space, that encodes the solution. In this context, it is a parallel approach to genetic algorithms, where the solution is encoded as a binary vector. The best individual of the ith generation (or iteration) is chosen as the base for the following $(i + 1)$ generation. The approach is depicted in Fig. 1. The binary vector of the current solution is called a kernel a is denoted with an index "ker" (e.g., a_{ker}). The newly generated neighborhood creates a set of c new binary vectors a_i with the same length as the vector a_{ker}. These new vectors can be viewed as a population and represented by a matrix $A_0 = (a_1, \dots, a_c)^T$. The degree of locality/globality of the optimization depends on the particular way the new population $A_0$is generated. The goal of the search is to find optimal parameters $x_{opt}(43)$ with respect to the define objective function $f(x)$ on a parametric space $D \in N$. Because of the binary representation, the parametric space is defined by a mapping Γ: $\{0,1\} \to D$. An important implementation detail of the mapping Γ is the translation of the binary vector from the Gray code into direct binary; afterwards, there is a problem-based decoding of the binary vector (0-1

problem, integer problem, or mixed integer problem). The following relationship is used $x = \Gamma(a)$ to denote the optimal solution as follows:

$$x_{opt} = \text{argmin}_{x \in D} \text{f}(x) \tag{43}$$

$$a_{opt} = \text{argmin}_{a \in \{0\ 1\}^n} f\big(\Gamma(a)\big) \tag{44}$$

Over this binary representation is defined the neighborhood relation, that describes the neighborhood *s* for each feasible a_{ker} as points $a \in s(a_{ker})$. The choice of the neighborhood function s determines the behavior and character of the HC algorithm (Fig. 1).

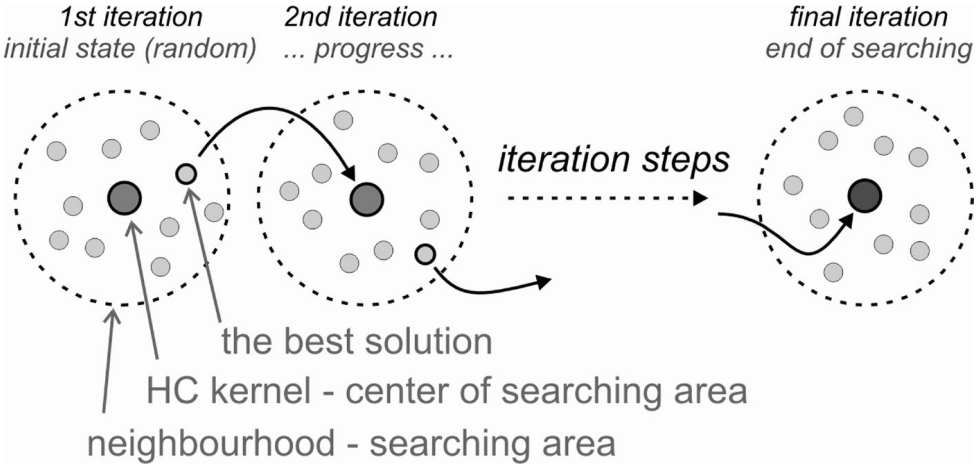


Fig. 1. An schematic example of the progress of the binary HC algorithm

The HC12 algorithm is very effectively parallelizable. Using the neighborhood function s (45) on a binary vector a_{ker}, the population A_0 is generated. The set of possible neighborhood functions is denoted by H (46).

$$s\colon a_{ker} \to A_0 \quad \text{i, e, } s\colon \{0\ 1\}^n \to (\{0\ 1\}^n)^c \tag{45}$$

$$H = \{s_0\ s_1\ \dots\ s_n\} \tag{46}$$

The number c of newly generated vectors in the population A_0 depends on the chosen neighborhood function s_k and on the length n of the binary vector a_{ker} – it is computed as $c = \binom{n}{k}$. For the realization of the transformations from the set H a system of matrices M is defined. The matrix M corresponding to the function s_k will be called a matrix of the k-th order and denoted by M_k. Matrix of the k-th order (M_k) is a matrix whose rows represent all points of the Hamming metric space that are distance k from the origin (zero vector of length n):

$$M_0 = (0_{1,1} \quad 0_{1,2} \quad \cdots \quad 0_{1,n})$$

$$M_1 = \begin{pmatrix} 1_{1,1} & 0_{1,2} & 0_{1,3} & \cdots & 0_{1,n} \\ 0_{2,1} & 1_{2,2} & 0_{2,3} & \cdots & 0_{2,n} \\ \vdots & & & \ddots & \\ 0_{c_1,1} & 0_{c_1,2} & 0_{c_1,3} & \cdots & 1_{c_1,n} \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 1_{1,1} & 1_{1,2} & 0_{1,3} & \cdots & 0_{1,n} \\ 1_{2,1} & 0_{2,2} & 1_{2,3} & \cdots & 0_{2,n} \\ \vdots & & & \ddots & \\ 0_{c_2,1} & 0_{c_2,2} & \cdots & 1_{c_2,n-1} & 1_{c_2,n} \end{pmatrix}$$

$$\vdots$$

$$M_n = (1_{1,1} \quad 1_{1,2} \quad \cdots \quad 1_{1,n})$$

Using the k-th order matrices, the function s_k can be effectively computed

as:

$$s_k: A_0 = a_{ker} \oplus M_k$$

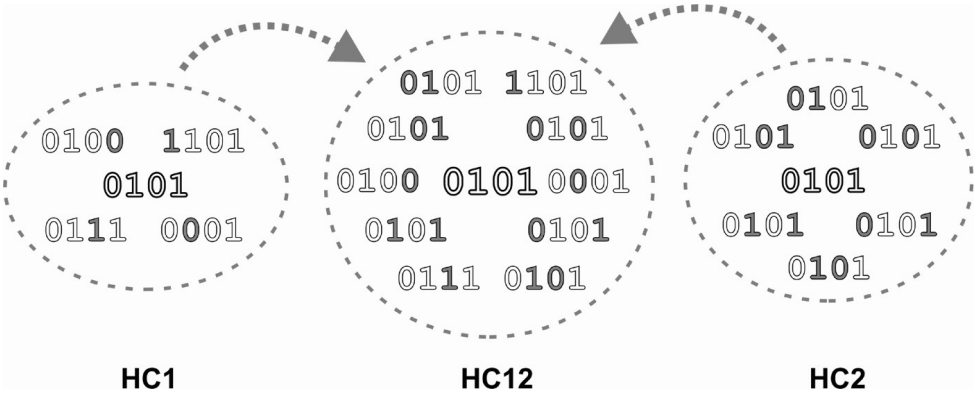


Fig. 2. An example of neighbourhood generation for 4-bit binary string using transformations $H = \{s_0,s_1,s_2\}$ and matrixes M_0,M_1,M_2, i.e. utilization in algorithms HC1, HC2 and their union HC12

From the practical point of view (because of the combinatorial expansion), only the transformations $M_0, M_1,$ and M_2 are used. The algorithm HC12 encodes in the last digit of its name the utilized transformations (upto order 2). The general paradigm of the HC12 algorithm is implemented using several input parameters: *fun* (indicator of the objective function f), *nRun* (the number of runs/restarts of the algorithm). The value *nRun* depends on the difficulty of the problem. The section of rows 6 to 10 are the HC12 algorithm itself. This part is very well (row 8) and well (row 9) parallelizable. The computations in row 8 contain the conversion from the Gray code into direct binary, implicitly. The main focus of this paper is on row 5 of the algorithm. How does one select a good starting solution? One possibility is to start at a random solution with the hope that after a sufficiently large number of tries, one does get a "good enough" solution. The other possibility is to start from a solution that is obtained by some heuristic. In this case, the heuristic in question entails the computation of the lower bound for the QAP (by one of the methods described earlier) and the projection of the obtained lower bound solution on the space of permutation matrices, by solving (40)-(42). The implementation of the HC12 algorithm for the QAP was described in (Matousek et al., 2019). As noted earlier, the optimization problem in (1)-(4) can be interpreted as a search over the space of permutation matrices $X \in \Pi_n$. From the problem structure of the QAP it is clear that swapping arbitrary columns of a (feasible) matrix X always results in a different feasible matrix (and swapping the rows of the matrix has the same effect).

Algorithm 1 The HC12 algorithm (Pseudo code of the general paradigm).

1: *fun,nRun* ← **inputs**
2: $M \leftarrow (M_0, M_1, M_2)^T$
3: $f_{best} \leftarrow \infty$
4: **for** *i* = [1 : *nRun*] **do**
5: $a_{opt} \leftarrow$ **random / heuristic**
6: **repeat**
7: $a_{ker} \leftarrow a_{opt}$
8: $A \leftarrow a_{ker} \oplus M$
9: $a_{opt} \leftarrow \text{argmin}_{a\in\{0,1\}^n} f\left(\Gamma(a)\right)$
10: **until** $a_{opt} = a_{ker}$
11: $f_{best}\,(i) \leftarrow f\left(\Gamma\left(a_{opt}\right)\right)$
12: $A_{best}\,(i,:) \leftarrow a_{opt}$
13: **endfor**
14: $[i, f_{min}] \leftarrow \min_i f_{best}\ (i)$
15: $a_{min} \leftarrow A_{best}\,(i,:)$
16: $x_{min} \leftarrow \Gamma(a_{min})$
17: **return** $\{f_{min}, a_{min}, x_{min}\}$

4. Results and discussion

The computational experiments were carried out on 53 symmetrical QAP instances from the QAPLIB. For these instances the GLB (12)-(15), HRW (16), convex quadratic (17)-(18), and semidefinite (19)-(36) bounds, and their projections (40)-(42) were computed. For the computation of the convex quadratic (AB) and semidefinite (PE) bounds, and for the computation of the LAP for the projection, the corresponding optimization problems were implemented in JuMP environment in JULIA language and the MOSEK solver was used to obtain the solutions. The results from these computations are summarized in Table 1. For one of the instances (tho150), the AB and PE formulations were too big to handle. These computations were carried out on Intel Xeon E5530 2.40GHz CPU with 16GB of RAM. The HC12 algorithm was implemented for HPC computations on GPU CUDA 7.x (i.e., NVIDIA RTX 2080, 8GB), where not more than 6GB were used for any of the QAP instances. From Table 1 we can see that, at least in general, the more complicated formulations (convex quadratic and semidefinite) produce better (higher) lower bounds, but not necessarily better (lower) starting point values, when judged solely on the resulting projection. The trade-off is that these more complicated formulations need quite a lot more computational resources (judged by the computational time) and are only feasible for instances up to $n = 100$. Also, every method produced the best lower bound and best projected value for at least one problem instance. It should be noted that the lower bounds are not only useful for constructing possible starting solutions for heuristics, but also help to judge the closeness of the solution obtained by the heuristic to the true optimum. This is especially important in situation, where the is otherwise no information about what the optimal value of the QAP instance might be. Next, we used the projected values from the lower bounding techniques as the starting points for the HC12 metaheuristic and run it 1,000 times for each problem instance. The best results of these simulations (the solution with the lowest objective value out of the 1,000) are reported in Table 2. We also include the results from simulations that used random permutations as the starting point (again 1,000). Similarly, to the results of the lower bounds, there is not a clear winner, as for each of the methods (even for the random start) there are instances where it produced solutions that were better than the ones from the other methods. However, we can compare each of the bounding methods with the random start to see if there is significant difference. This comparison is summarized in Table 3 – we can see that even the "worst" performing lower bounding technique (HWB) was significantly better that random start, beating it in 30 of the 53 instances. The "best" performing lower bounding technique was the most complicated semidefinite formulation (PE), which was better than random start in 41 of the 52 instances. We can also see that the GLB method performed a bit better than the much more complicated convex quadratic (AB) one. Similar pattern can be observed for the median results reported in Table 4. The main difference is that the random start was never the best scoring method, while each of the lower bounding methods were the best in at least 6 instances. The comparison of the lower bounding method with random start for median results summarized in Table 5 shows even bigger difference than the one for best (minimum) results – the "worst" lower bounding technique (HWB, again) was better than random start in 41 of the 53 instances, and the "best" one (PE, again) was better in every one of the 52 instances. The GLB and AB methods perform similarly well. From these results, it is clear that starting a heuristic from a carefully chosen points leads to an increase in quality of the resulting solutions. The choice of the technique for constructing these starting points mainly depends on the computational resources at our disposal. While for the QAP the semidefinite (PE) formulation produced the best behaving starting points, it was also the most computationally demanding method, requiring the use of advanced convex optimization algorithms or the use of powerful solvers. In contrast to this, the GLB method produce starting points that are almost as good as the PE one, but the computational requirements for GLB are negligible.

5. Conclusion

In this paper we have studied the effects of using the lower bounding techniques for the QAP as for the generation of starting points for the HC12 heuristic, that subsequently tried to find the optimal solution for the QAP. We have shown through extensive numerical computations that this utilization of the lower bounding techniques significantly improves the values of the resulting solutions. Out of the four compared lower bounding techniques, the best overall results were obtained by using the semidefinite relaxation method, which was also the most computationally demanding one. When the computational resources, or the access to high quality semidefinite optimization solvers are limited, the GLB bound can serve as an excellent surrogate – although the resulting solutions are not as good, the computational requirements are negligible.

Future research will focus on extending the multicriteria and stochastic QAP instances. Also, the evaluation of various other heuristics that can use the starting points could be interesting, as different methods could benefit more (or less) from starting from an already decent point. Lastly, we expect to work on the evaluation of the starting solutions for other NP-hard optimization problems.

Table 1
Lower bounds (LB), values of the projections (PV), and computational time (T) of the four considered QAP lower bounding techniques for selected QAPLIB problems (BKW – best known value). Best results (highest for LB and lowest for PV) are emphasized in bold.

		GLB (Gilmore, 1962)			HRW (Hadley et al., 1992)			AB (Anstreicher & Brixius, 2001)			PE (Peng et al., 2015)		
Instance	BKW	LB	PV	T [s]	LB	PV	T [s]	LB	PV	T [s]	LB	PV	T [s]
chr12a	9552	724	44232	0.001	0	25638	0.001	0	**23246**	0.095	**8499**	25752	1.136
chr12b	9742	7146	**25580**	0.001	0	26712	0.001	0	27088	0.107	**7340**	41170	1.317
chr12c	11156	**7976**	**18784**	0.001	0	31608	0.001	0	30876	0.089	9832	40438	1.052
chr15a	9896	5625	50174	0.001	0	25958	0.001	0	**25880**	0.253	**7441**	46976	2.515
chr15b	7990	4653	54254	0.001	0	38470	0.001	0	**19008**	0.256	**5166**	30798	2.870
chr15c	9504	6165	44602	0.001	0	**26144**	0.001	0	35936	0.260	**8808**	43370	2.207
chr18a	11098	6779	89486	0.001	0	**38778**	0.001	0	51800	0.331	**9077**	78282	4.383
chr18b	1534	**1534**	4640	0.001	0	**2372**	0.001	0	3922	0.338	**1534**	4156	2.580
chr20a	2192	2150	9778	0.001	0	7742	0.001	0	**6904**	0.489	**2156**	10302	5.814
chr20b	2298	2196	10430	0.001	0	**6418**	0.001	0	7386	0.497	**2237**	10320	7.654
chr20c	14142	8601	79430	0.013	0	**63268**	0.001	0	63350	0.511	**8825**	69124	7.588
chr22a	6156	5924	17622	0.003	0	**10178**	0.001	0	10588	0.815	**5964**	10316	8.907
chr22b	6194	5936	13486	0.002	0	**9530**	0.001	0	11744	0.995	**6015**	10264	7.664
chr25a	3796	2765	18964	0.002	0	14186	0.001	0	13062	1.444	**3244**	**11708**	17.67
had20	6922	6166	7550	0.001	6625	7460	0.001	6671	**7184**	0.519	**6778**	7486	4.486
kra30a	88900	68360	120000	0.004	63717	**117490**	0.001	68467	111970	10.52	**73983**	118920	34.27
kra30b	91420	69065	**118720**	0.005	120990	140040	0.001	68876	122440	3.782	**68737**	127900	20.88
kra32	88700	67390	121620	0.007	59735	119450	0.002	64591	**117010**	4.780	**72297**	119540	29.96
nug18	1930	1554	2402	0.002	1663	**2250**	0.001	1703	2320	0.303	**1753**	2278	2.937
nug20	2570	2057	3080	0.002	2196	2940	0.001	2253	**2884**	0.481	**2338**	2972	4.191
nug21	2438	1833	3198	0.002	1979	**2914**	0.001	2051	3030	0.627	**2215**	3266	5.777
nug22	3596	2483	4598	0.002	2966	4448	0.001	3074	4346	0.816	**3284**	**4312**	9.638
nug24	3488	2676	4222	0.035	2960	4170	0.001	3024	4272	1.378	**3178**	**4066**	7.517
nug25	3744	2869	4728	0.003	3190	4634	0.001	3267	4534	1.521	**3404**	**4496**	8.902
nug27	5234	3701	**6246**	0.003	4493	6586	0.001	4604	6378	2.495	**4820**	6548	18.64
nug28	5166	3786	6470	0.003	4433	6310	0.001	4538	**6208**	2.705	**4732**	6274	15.07
nug30	6124	4539	7706	0.003	5266	7342	0.001	5360	**7270**	3.702	**5608**	7664	25.51
scr15	51140	44737	**70154**	0.002	10355	75950	0.001	12478	77786	0.299	**46015**	83280	1.339
scr20	110030	86766	203736	0.001	16113	**172306**	0.001	22714	193250	0.474	**92426**	193396	5.518
sko42	15812	11311	19522	0.008	13830	19088	0.002	14029	18748	18.23	**14612**	**18492**	121.7
sko64	48498	32522	57316	0.030	43890	56376	0.003	44513	**56214**	126.2	**45467**	56766	623.7
sko72	66256	44280	75860	0.048	60402	75772	0.004	61069	**75632**	222.4	**61497**	76082	1209
sko81	90998	60283	105932	0.061	82277	104844	0.005	83433	**103642**	466.3	**85795**	103954	2757
sko90	115534	75531	132996	0.083	105983	**131398**	0.006	107171	131794	1448	**109260**	131646	4585
sko100a	152002	98953	171886	0.091	139365	**170880**	0.009	140946	172554	2847	**144091**	171294	9946
sko100b	153890	99028	174226	0.100	141251	**173878**	0.007	143138	174290	3055	**145783**	174950	10745
sko100c	147862	95979	169888	0.101	135011	**167142**	0.007	136773	170340	3306	**140146**	169792	8920
sko100d	149576	95921	172024	0.091	136979	**167642**	0.007	138736	169774	2928	**140072**	171152	10162
sko100e	149150	95551	171360	0.093	136996	168732	0.007	138711	**168640**	3090	**139277**	169914	9688
sko100f	149036	96016	169768	0.093	136860	170008	0.007	138661	**169090**	3634	**140885**	169564	10627
ste36a	9526	7124	**14866**	0.006	0	16112	0.001	0	17454	9.257	**7731**	20690	60.95
ste36b	15852	8653	47768	0.007	0	42034	0.001	0	**37398**	8.375	**12930**	51936	60.06
ste36c	8.239e6	6.393e6	2.191e7	0.006	0	2.119e7	0.001	0	1.952e7	8.193	**6.546e6**	**1.775e7**	101.4
tai25a	1.167e6	962417	1.457e6	0.002	956657	**1.310e6**	0.001	**967207**	1.366e6	1.758	958027	1.411e6	12.89
tai50a	4.9386e	3.854e6	5.838e6	0.017	3.840e6	5.680e6	0.002	**3.870e6**	**5.613e6**	67.49	3.842e6	5.667e6	253.0
tai60a	7.205e6	5.55e6	8.481e6	0.27	5.537e6	8.398e6	0.004	**5.575e6**	8.464e6	82.67	5.544e6	**8.216e6**	841.2
tai80a	1.349e7	1.032e7	1.570e7	0.065	1.030e7	1.568e7	0.005	**1.035e7**	**1.573e7**	413.5	1.031e7	1.556e7	3213
tai100a	2.105e7	1.582e7	2.403e7	0.101	1.579e7	2.348e7	0.007	**1.585e7**	**2.347e7**	1667	1.552e7	2.347e7	6460
tho30	149936	90578	195698	0.003	119255	**193756**	0.001	124217	200194	4.067	**131588**	198154	27.20
tho40	240516	143804	**298906**	0.007	191042	303704	0.002	197661	319004	13.45	**210210**	313114	109.9
tho150	8.133e6	4.123e6	**9.703e6**	0.226	**7.350e6**	9.756e6	0.013	–	–	–	–	–	–
wil50	48816	38069	53942	0.014	45731	53420	0.003	46194	**52938**	40.64	**46901**	53754	281.4
wil100	273038	210949	293908	0.095	260827	293206	0.007	262584	**291630**	2591	**264724**	294948	11078

Table 2
Best (minimum) results from the simulations. If the BKW is confirmed optimal, it is highlighted in bold. Also, in bold is the method that produced the best solution for the given instance.

Instance	BKW	Rand	GLB	HRW	AB	PE
chr12a	**9552**	**9552**	**9552**	**9552**	**9552**	**9552**
chr12b	**9742**	**9742**	**9742**	**9742**	**9742**	**9742**
chr12c	**11156**	11186	**11156**	**11156**	**11156**	**11156**
chr15a	**9896**	10094	10010	9980	10106	**9978**
chr15b	**7990**	8626	**8210**	9096	8458	8452
chr15c	**9504**	10118	**9504**	10426	9940	10002
chr18a	**11098**	**11682**	**11682**	12396	12004	12424
chr18b	**1534**	1538	**1534**	**1534**	1538	**1534**
chr20a	**2192**	2480	2532	2592	**2398**	2402
chr20b	**2298**	2612	2608	**2598**	2674	2618
chr20c	**14142**	**14610**	14988	15636	17274	14876
chr22a	**6156**	6408	6342	6354	6456	**6336**
chr22b	**6194**	6522	6526	6534	6410	**6352**
chr25a	**3796**	5062	4678	5056	4970	**4230**
had20	**6922**	6924	6928	**6922**	6956	**6922**
kra30a	**88900**	93460	92480	93850	93460	**92300**
kra30b	**91420**	95020	94570	94690	93620	**92380**
kra32	**88700**	**91660**	92420	92270	92650	92320
nug18	**1930**	1958	**1936**	1950	1938	1938
nug20	**2570**	2598	2614	**2590**	2602	2598
nug21	**2438**	2458	2452	2472	**2450**	2452
nug22	**3596**	3628	**3610**	3628	3628	**3610**
nug24	**3488**	3552	3554	3582	3546	**3528**
nug25	**3744**	3806	3788	3800	**3760**	3762
nug27	**5234**	5298	5298	5328	**5294**	5304
nug28	**5166**	5314	5284	5288	5272	**5260**
nug30	**6124**	6272	6260	6316	6254	**6220**
scr15	**51140**	**51140**	52340	**51140**	**51140**	**51140**
scr20	**110030**	111078	111938	110802	112660	**110772**
sko42	15812	16304	16282	16290	**16106**	16172
sko64	48498	50090	**49904**	49932	49970	49942
sko72	66256	68298	68182	68264	**67902**	68140
sko81	90998	93684	93492	93968	93840	**93148**
sko90	115534	119630	119064	119078	119092	**118446**
sko100a	152002	157426	157034	156934	**156116**	156820
sko100b	153890	159060	**158002**	158456	158220	158184
sko100c	147862	152742	152592	153186	152476	**152374**
sko100d	149576	154708	154340	**153896**	153978	154144
sko100e	149150	**153880**	154522	153930	154010	154536
sko100f	149036	154284	153994	153788	153766	**153558**
ste36a	**9526**	10234	10126	10052	**9790**	10266
ste36b	**15852**	17786	17140	17112	**16724**	17770
ste36c	**8239110**	8701576	8678652	8738822	8695554	**8578694**
tai25a	**1167256**	1194194	**1177180**	1188890	1188248	1187984
tai50a	4938796	5148702	**5119448**	5131652	5130768	5132594
tai60a	7205962	7486562	7506384	7499212	7434242	**7427410**
tai80a	13499184	14034018	14027470	**13966388**	14050896	13993668
tai100a	21052466	21951138	21932812	21957694	**21893240**	21932070
tho30	**149936**	154134	156170	153558	154874	**152020**
tho40	240516	251428	249370	248116	**247260**	251034
tho150	8133398	8511942	8461186	**8454432**	–	–
wil50	48816	49504	49472	49652	**49434**	49470
wil100	273038	278428	**277210**	277730	277230	277458

Table 3
Comparison of the lower bounding methods with Rand – best (minimum) results.

	GLB	HWB	AB	PE
Rand better	12	19	13	7
Rand worst	37	30	33	41
Rand equal	4	4	6	4

Table 4
Median results from the simulations. If the BKW is confirmed optimal, it is highlighted in bold. Also, in bold is the method that produced the lowest median value for the given instance.

Instance	BKW	Rand	GLB	HRW	AB	PE
chr12a	**9552**	12531	11866	**11550**	12798	12130
chr12b	**9742**	13170	**10570**	11628	11548	11886
chr12c	**11156**	14221	**13060**	14139	13518	13355
chr15a	**9896**	14691	13886	13850	**13625**	14264
chr15b	**7990**	13505	12334	13637	12490	**12142**
chr15c	**9504**	15445	**14303**	15812	14518	14675
chr18a	**11098**	19074	18159	18664	17565	**17237**
chr18b	**1534**	1779	**1730**	1760	1776	1734
chr20a	**2192**	3480	3374	3406	**3280**	3359
chr20b	**2298**	3455	3384	3468	3453	**3272**
chr20c	**14142**	25957	**23068**	24678	27073	23338
chr22a	**6156**	7168	7007	7016	7144	**6962**
chr22b	**6194**	7218	7086	7227	7078	**7059**
chr25a	**3796**	6819	6068	6677	6520	**5687**
had20	**6922**	7008	7002	7018	**6982**	7000
kra30a	**88900**	99285	98410	100070	97930	**97340**
kra30b	**91420**	100360	100945	100275	100695	**100145**
kra32	**88700**	98365	98360	98435	98340	**98260**
nug18	**1930**	2038	2022	**2010**	2018	2012
nug20	**2570**	2728	2708	2708	**2704**	2714
nug21	**2438**	2596	2562	2570	**2548**	2560
nug22	**3596**	3789	3738	3878	3740	**3734**
nug24	**3488**	3754	3731	3744	3689	**3670**
nug25	**3744**	4002	3950	3930	**3920**	3940
nug27	**5234**	5588	5544	5532	5497	**5486**
nug28	**5166**	5546	5492	5492	5462	**5460**
nug30	**6124**	6562	6490	6524	6518	**6474**
scr15	**51140**	57428	56850	**56240**	56604	56613
scr20	**110030**	125945	124260	122999	**121845**	122096
sko42	15812	16854	16728	16830	**16661**	16663
sko64	48498	51228	50996	51092	50894	**50853**
sko72	66256	69782	69362	69625	69278	**69267**
sko81	90998	95591	95186	95379	95093	**94751**
sko90	115534	121745	121200	**120730**	121122	120980
sko100a	152002	159834	159447	159059	**158770**	159425
sko100b	153890	161739	160704	160686	**160100**	160450
sko100c	147862	156070	155364	155557	**154906**	155008
sko100d	149576	157353	156497	156886	**156012**	156377
sko100e	149150	157467	156798	156325	156646	**156178**
sko100f	149036	156510	155907	**155665**	155841	155841
ste36a	**9526**	11375	11134	11126	**11062**	11192
ste36b	**15852**	21794	20978	**20601**	20783	20899
ste36c	**8239110**	9523806	9564108	9602558	9566412	**9411506**
tai25a	**1167256**	1227125	**1223279**	1226160	1223803	1226389
tai50a	4938796	5266951	5249145	5255063	5235616	**5234108**
tai60a	7205962	7629342	7632746	7622290	7617218	**7592056**
tai80a	13499184	14235086	14239312	14251040	14249678	**14214487**
tai100a	21052466	22265769	22209784	22277072	**22202771**	22230955
tho30	**149936**	162522	163144	162024	161934	**159821**
tho40	240516	262386	259343	259822	**258655**	262188
tho150	8133398	8647360	8609647	**8594449**	–	–
wil50	48816	50434	50269	50544	**50092**	50126
wil100	273038	280670	280037	279836	**279480**	279705

Table 5
Comparison of the lower bounding methods with Rand – median results

	GLB	HWB	AB	PE
Rand better	5	12	5	0
Rand worst	47	41	47	52

Acknowledgement

This work was supported by IGA BUT: FSI-S-20-6538.

References

Abdel-Basset, M., Manogaran, G., El-Shahat, D., & Mirjalili, S. (2018). Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments. *Applied soft computing*, 73, 530-546.

Adams, W. P., & Johnson, T. A. (1994). Improved linear programming-based lower bounds for the quadratic assignment problem. *DIMACS series in discrete mathematics and theoretical computer science*, 16, 43-77.

Adams, W. P., Guignard, M., Hahn, P. M., & Hightower, W. L. (2007). A level-2 reformulation–linearization technique bound for the quadratic assignment problem. European Journal of Operational Research, 180(3), 983-996.

Ahmed, Z. H. (2015). A multi-parent genetic algorithm for the quadratic assignment problem. *Opsearch*, 52(4), 714-732.

Aksan, Y., Dokeroglu, T., & Cosar, A. (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, 103, 105-115.

Anstreicher, K. M., & Brixius, N. W. (2001). A new bound for the quadratic assignment problem based on convex quadratic programming. *Mathematical Programming*, 89(3), 341-357.

Anstreicher, K. M. (2003). Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97(1), 27-42.

Benlic, U., & Hao, J. K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1), 584-595.

Burkard, R. E., & Rendl, F. (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17(2), 169-174.

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB–a quadratic assignment problem library. *Journal of Global optimization*, 10(4), 391-403.

Burkard, R. E., Dell'Amico, M., & Martello, S. (2012). *Assignment problems: revised reprint*. Society for Industrial and Applied Mathematics.

Cela, E. (2013). The quadratic assignment problem: theory and algorithms (Vol. 1). Springer Science & Business Media.

Cela, E., Deineko, V., & Woeginger, G. J. (2018). New special cases of the Quadratic Assignment Problem with diagonally structured coefficient matrices. *European journal of operational research*, 267(3), 818-834.

Chmiel, W., & Szwed, P. (2016). Bees algorithm for the quadratic assignment problem on CUDA platform. In *Man–Machine Interactions 4* (pp. 615-625). Springer, Cham.

Czapiński, M. (2013). An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. *Journal of Parallel and Distributed Computing*, 73(11), 1461-1468.

de Klerk, E., & Sotirov, R. (2012). Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical programming*, 133(1), 75-91.

Dell'Amico, M., Diaz, J. C. D., Iori, M., & Montanari, R. (2009). The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11), 3002-3012.

Dickey, J. W., & Hopkins, J. W. (1972). Campus building arrangement using TOPAZ. *Transportation Research*, 6(1), 59-68.

Dokeroglu, T. (2015). Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem. *Computers & Industrial Engineering*, 85, 86-101.

Dokeroglu, T., & Cosar, A. (2016). A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, 52, 10-25.

Eschermann, B., & Wunderlich, H. J. (1990). Optimized synthesis of self-testable finite state machines. In *20th international symposium on fault-tolerant computing (FTCS 20).*

Fischetti, M., Monaci, M., & Salvagnin, D. (2012). Three ideas for the quadratic assignment problem. *Operations research*, 60(4), 954-964.

Fleurent, C., & Ferland, J. A. (1994). Genetic hybrids for the quadratic assignment problem. *Quadratic assignment and related problems*, 16, 173-187.

Gilmore, P. C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the society for industrial and applied mathematics*, 10(2), 305-313.

Hadley, S. W., Rendl, F., & Wolkowicz, H. (1992). A new lower bound via projection for the quadratic assignment problem. *Mathematics of Operations Research*, 17(3), 727-739.

Hafiz, F., & Abdennour, A. (2016). Particle Swarm Algorithm variants for the Quadratic Assignment Problems-A probabilistic learning approach. *Expert Systems with Applications*, 44, 413-431.

Haghani, A., & Chen, M. C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6), 437-454.

Hahn, P. M., Zhu, Y. R., Guignard, M., Hightower, W. L., & Saltzman, M. J. (2012). A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS Journal on Computing*, 24(2), 202-209.

Hameed, A., Aboobaider, B., Mutar, M., & Choon, N. (2020). A new hybrid approach based on discrete differential evolution algorithm to enhancement solutions of quadratic assignment problem. *International Journal of Industrial Engineering Computations*, 11(1), 51-72.

Harris, M., Berretta, R., Inostroza-Ponta, M., & Moscato, P. (2015, May). A memetic algorithm for the quadratic assignment problem with parallel local search. In *2015 IEEE congress on evolutionary computation (CEC)* (pp. 838-845). IEEE.

Helber, S., Böhme, D., Oucherif, F., Lagershausen, S., & Kasper, S. (2016). A hierarchical facility layout planning approach for large and complex hospitals. *Flexible services and manufacturing journal*, 28(1), 5-29.

Hoffman, A. J., & Wielandt, H. W. (2003). The variation of the spectrum of a normal matrix. In *Selected Papers Of Alan J Hoffman: With Commentary* (pp. 118-120).

Hubert, L., & Schultz, J. (1976). Quadratic assignment as a general data analysis strategy. *British journal of mathematical and statistical psychology*, 29(2), 190-241.

Hussin, M. S., & Stützle, T. (2014). Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. *Computers & operations research*, 43, 286-291.

Kaufman, L., & Broeckx, F. (1978). An algorithm for the quadratic assignment problem using Bender's decomposition. *European Journal of Operational Research*, 2(3), 207-211.

Karisch, S. E., Cela, E., Clausen, J., & Espersen, T. (1999). A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing*, 63(4), 351-403.

Koopmans, T. C., & Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, 53-76.

Matousek, R., & Zampachova, E. (2011). Promising GAHC and HC12 algorithms in global optimization tasks. *Optimization methods and software*, 26(3), 405-419.

Matousek, R., Popela, P., & Kudela, J. (2017). Heuristic approaches to stochastic quadratic assignment problem: Var and cvar cases. *Mendel,* 23(1), 73-78.

Matousek, R., Dobrovsky, L., & Kudela, J. (2019, July). The quadratic assignment problem: metaheuristic optimization using HC12 algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 153-154).

Mohammadi, J., Mirzaie, K., & Derhami, V. (2015, November). Parallel genetic algorithm based on GPU for solving quadratic assignment problem. In *2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI)* (pp. 569-572). IEEE.

Laporte, G., & Mercure, H. (1988). Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3), 378-381.

Lawler, E. L. (1963). The quadratic assignment problem. *Management science*, 9(4), 586-599.

Li, Y., & Pardalos, P. M. (1992). Generating quadratic assignment test problems with known optimal permutations. *Computational Optimization and Applications*, 1(2), 163-184.

Peng, J., Mittelmann, H., & Li, X. (2010). A new relaxation framework for quadratic assignment problems based on matrix splitting. *Mathematical Programming Computation*, 2(1), 59-77.

Peng, J., Zhu, T., Luo, H., & Toh, K. C. (2015). Semi-definite programming relaxation of quadratic assignment problems based on nonredundant matrix splitting. *Computational Optimization and Applications*, 60(1), 171-198.

Popela, P., Matousek, R., & Kudela, J. (2016). Heuristic approaches to stochastic quadratic assignment problem: VO and MM cases. *Mendel* 22(1), 117-122.

Samanta, S., Philip, D., & Chakraborty, S. (2018). Bi-objective dependent location quadratic assignment problem: Formulation and solution using a modified artificial bee colony algorithm. *Computers & Industrial Engineering*, 121, 8-26.

Sanhueza, C., Jiménez, F., Berretta, R., & Moscato, P. (2017, June). *PasMoQAP: a parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem. In 2017 IEEE congress on evolutionary computation (CEC)* (pp. 1103-1110). IEEE.

Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *Siam Review*, 3(1), 37-50.

Taillard, É. (1991). Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5), 443-455.

Tosun, U. (2015). On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering applications of artificial intelligence*, 39, 267-278.

Tsutsui, S., & Fujimoto, N. (2009, July). Solving quadratic assignment problems by genetic algorithms with GPU computation: a case study. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers* (pp. 2523-2530).

Xia, Y., & Yuan, Y. X. (2006). A new linearization method for quadratic assignment problems. *Optimisation Methods and Software*, 21(5), 805-818.

Zhang, H., Beltran-Royo, C., & Ma, L. (2013). Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. *Annals of Operations Research*, 207(1), 261-278.

Zhao, Q., Karisch, S. E., Rendl, F., & Wolkowicz, H. (1998). Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1), 71-109.