VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta strojního inženýrství

Ing. Petr Palubják

VÍCEPRODUKTOVÉ TOKY V SÍTÍCH

MULTICOMMODITY NETWORK FLOWS

Zkrácená verze Ph.D. Thesis

Obor: Technická kybernetika

Školitel: doc. RNDr. Ing. Miloš Šeda, Ph.D.

Oponenti: prof. Ing. Vladimír Strakoš, DrSc.

doc. Ing. Radim Farana, CSc.

doc. Ing. Čestmír Ondrůšek, CSc.

Datum obhajoby: 12. 12. 2003

KLÍČOVÁ SLOVA

víceproduktové toky, stochastické heuristiky, toky s minimálními náklady, maximální toky, nejkratší cesty

KEYWORDS

Multicommodity Flows, Stochastic heuristics, Minimum Cost Flows, Maximum Flows, Shortest Paths

MÍSTO ULOŽENÍ PRÁCE

Oddělení pro vědu a výzkum Fakulty strojního inženýrství Vysokého učení technického v Brně

ISBN 80-214- 2551-2
ISSN 1213-4198

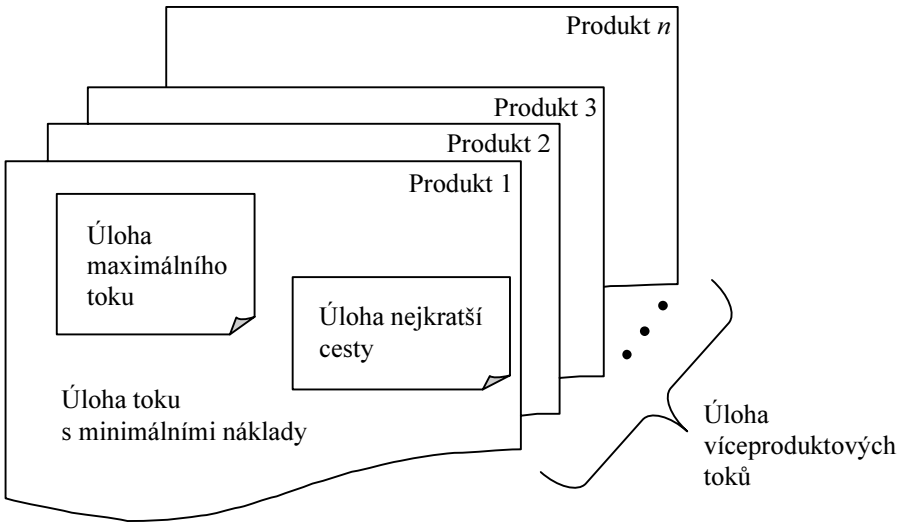
Obsah

A journey of a thousand miles starts with a single step and if that step is the right step, it becomes the last step
–Lao Tzu

1 Úvod

Kamkoliv se v dnešním světě podíváme, najdeme nějaké sítě. Elektrické a energetické sítě nám přináší světlo, teplo a zábavu do našich domovů. Telefonní sítě nám umožňují vzájemně komunikovat, dálniční, železniční a letecká síť nám dovolují překonávat velké zeměpisné vzdálenosti, abychom se dostali do práce, abychom mohli navštívit své blízké či abychom mohli navštívit nová místa a získat nové zkušenosti. Výrobní a distribuční sítě nám poskytují prostředky k životu – potraviny a spotřební výrobky. A počítačové sítě dokonce změnily způsob, jakým sdílíme informace nebo realizujeme obchody. Ve všech těchto oblastech a v mnoha dalších chceme zpravidla přemístit nějaké entity (elektřinu, spotřební výrobky, osoby, vozidla nebo zprávy) z jednoho místa na jiné místo v dané síti a udělat to co možná nejefektivněji. Naším cílem je poskytovat nejen dobré služby uživatelům sítě, ale i použít základní (zpravidla nákladné) přenosové zařízení efektivně. Tento typ úloh je dobře popsán matematickým modelem známým jako úloha víceproduktových toků v sítích.

Víceproduktové toky (VT) představují velmi komplexní typ úlohy (NP-těžký problém [Khul94]), kterým jsme schopni řešit řadu problémů z praxe. Jak vidíme z obrázku 1.1, VT jsou nejobecnější formou několika jednodušších typů úloh. Pokud budeme pracovat pouze s jedním produktem (tzv. jednoproduktové úlohy), úloha se velmi zjednoduší a pro řešení tohoto typu úloh již existují speciální pseudopolynomiální a polynomiální algoritmy. Vidíme, že na úlohu toku s minimálními náklady opět můžeme pohlížet jako na obecnější případ dvou jiných úloh – nejkratší cesty a maximálního toku. Jak pomocí VT, tak i pomocí jejich jednodušších modelů jsme schopni řešit mnoho praktických úloh. Přehled některých aplikací najdete v tabulce na obrázku 1.2. Poznamenejme ještě, že jednoproduktové úlohy můžeme řešit i pomocí (univerzálního) síťového simplexového algoritmu ([Ahuj93], [Hoch98]).



Obrázek 1.1 Model víceproduktových toků.

Tato práce se zaměřuje nejen na úlohu VT, ale i na její tři speciální případy. I když s pomocí existujících algoritmů pro VT ([Dant61, Geof74, Kenn80, Arro86]) bychom byli schopni řešit i jednoproduktové úlohy, nebylo by to příliš smysluplné. Existující přesné VT algoritmy jsou pouze exponenciální a pro řešení jednoproduktových úloh již existuje řada efektivnějších polynomiálních algoritmů. Výpočetní náročnost exponenciálních algoritmů je enormní a umožňuje nám nalézt optimum pouze u úloh malého rozsahu. Jejich použitelnost pro praxi je tak velmi omezená. Z tohoto důvodu tady existuje snaha vývojářů o vytváření různých heuristických algoritmů, které nám sice nezaručí nalezení optima, ale na druhou stranu nám umožní v „rozumném“ čase řešit mnohem rozsáhlejší problémy. Většina heuristik je určena pro řešení konkrétního problému (tzv. problémově specifické heuristiky) a nelze zpravidla zobecnit (např. [Agga95]).

Heuristické algoritmy navržené v této práci byly inspirovány stochastickými heuristickými metodami ([Mich96], [Reev93]) a jejich použití je univerzální.

Pojmenování úlohy (autor)	Druh algoritmu
rozvržení kontrolních stanovišť ve výrobní lince ([White69])	nejkratší cesty
aproximace po částech lineární funkce ([Imai86])	nejkratší cesty
úloha o batohu ([Fulk66])	nejkratší cesty
model městské dopravy ([Zawa87])	nejkratší cesty
návrh přepojovaných počítačových sítí ([Jane96])	maximální toky
rozvrhování výroby na více shodných strojích ([Fede86])	maximální toky
maticové zaokrouhlování ([Bach66])	maximální toky
distribuovaný výpočet na dvouprocesorovém počítači ([Ston77])	maximální toky
distribuční úloha ([Glov77])	toky s minimálními náklady
rekonstrukce levé srdeční komory z rentgenu ([Slum82])	toky s minimálními náklady
optimální vytížení letadla na trase s mezipřistáními ([Gupt85])	toky s minimálními náklady
evakuační plán budovy ([Chal82])	toky s minimálními náklady
skladování sezónní produkce ([Jewe57])	víceproduktové toky
návrh VLSI čipů ([Kort88])	víceproduktové toky
úloha obměny výrobního zařízení ([Agga95])	víceproduktové toky

Obrázek 1.2 Příklady praktického využití VT.

1.1 Cíle disertační práce

Cílem disertační práce bylo podrobné studium VT a navržení stochastického heuristického algoritmu pro jejich řešení. V rámci studia VT jsme se zaměřili i na jejich speciální jednoproduktové verze, protože navržený heuristický algoritmus rozkládá komplexnější úlohu VT na posloupnost jednoproduktových úloh. Jedním z hlavních kritérií při návrhu heuristického algoritmu byla jeho výpočetní rychlost. Ta byla značně ovlivněna především rychlostí použitých (přesných) algoritmů pro řešení jednoproduktových úloh. Proto byl podrobně zkoumán i každý typ jednoproduktové úlohy. Toto zkoumání zahrnovalo vždy:

- definici matematického modelu dané jednoproduktové úlohy
- zkoumání oblastí použití dané jednoproduktové úlohy
- popis a programovou implementaci vybraných algoritmů
- porovnání algoritmů a určení nejvhodnějšího algoritmu pro použití v heuristickém VT algoritmu

Poté, co jsme nalezli vhodné jednoproduktové algoritmy, jsme se mohli zaměřit na analýzu a řešení samotné víceproduktové úlohy, které zahrnovalo:

- definici matematických modelů
- zkoumání oblastí použití VT
- možnosti řešení VT pomocí exaktních a aproximativních algoritmů
- návrh několika stochastických heuristických algoritmů pro řešení VT
- implementaci vybraných stochastických heuristických algoritmů
- porovnání implementovaných heuristických algoritmů

2 Nejkratší cesty

Úloha nejkratší cesty je jednou ze základních úloh toků v sítích. Je stavebním kamenem mnoha dalších algoritmů (viz obrázek 1.1). Velmi často se objevuje v praxi – při mnoha činnostech bychom chtěli poslat nějaký materiál mezi dvěma určenými body v síti co možná nejrychleji, co možná nejlevněji, co možná nejspolehlivěji. Ačkoliv úloha nejkratší cesty je relativně snadno řešitelná (základními algoritmy), návrh a analýza nejefektivnějších algoritmů již vyžaduje značné znalosti a přehled v problematice.

Úloha nejkratší cesty se často objevuje v telekomunikacích či v dopravě, kdykoliv chceme poslat zprávu či automobil mezi dvěma geografickými místy co možná nejrychleji či co nejlevněji. Významným příkladem využití je plánování městské dopravy ([Zawa87]). Ve většině algoritmů pro hledání modelu městské dopravy se řeší jako podproblém v hlavním algoritmu velké množství úloh nejkratší cesty (pro každou dvojici počátečních a koncových uzlů v síti zvlášť). Jmenujme ještě několik dalších příkladů známých z literatury: aproximace po částech lineární funkce ([Imai86]), řešení tzv. úlohy o batohu ([Fulk66]), plánování počtu telefonních operátorů ([Bart80]), optimální rozložení kontrolních stanovišť v rámci výrobní linky ([Whit69]) atd.

2.1 Matematický model

Uvažujme orientovanou síť $G = (N, A)$, kde N je množina uzlů ($n = |N|$) a A je množina hran ($m = |A|$), s délkou (náklady) c_{ij}, které jsou asociovány s každou hranou $(i, j) \in A$. Síť obsahuje jeden význačný uzel s, který budeme označovat jako uzel *zdrojový* (*source*). Nechť $A(i)$ reprezentuje sousední hrany uzlu i a nechť $C = \max \{c_{ij}: (i, j) \in A\}$. Budeme definovat *délku orientované cesty* jako součet délek jednotlivých hran na cestě. Úloha nejkratší cesty stanoví pro každý nezdrojový uzel $i \in N$ orientovanou cestu s nejkratší délkou z uzlu s do uzlu i. Alternativně bychom mohli úlohu vidět jako poslání jedné jednotky toku (nejlevněji, jak je to jen možné) z uzlu s do každého z uzlů $N - \{s\}$ v kapacitně neomezené síti. Z tohoto hlediska můžeme formulovat úlohu nejkratší cesty jako úlohu lineárního programování následujícím způsobem:

$$\text{Minimalizuj} \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2.1a}$$

za podmínek

$$\sum_{(j:(i,j)\in A)} x_{ij} - \sum_{(j:(j,i)\in A)} x_{ji} = \begin{cases} n-1 & \text{pro } i = s \\ -1 & \text{pro všechna } i \in N - \{s\} \end{cases} \tag{2.1b}$$

$$x_{ij} \geq 0 \qquad \text{pro všechny } (i, j) \in A \tag{2.1c}$$

Aby byla úloha nejkratší cesty řešitelná pomocí dále uvedených algoritmů, musí síť definující problém splňovat několik předpokladů:

- Všechny délky hran jsou celočíselné.
- Síť obsahuje orientovanou cestu z uzlu s do všech zbylých uzlů v síti.
- Síť neobsahuje záporný cyklus (úloha nejkratší cesty v síti se záporným cyklem je NP-úplný problém).
- Síť je orientovaná.

Ovšem, jak se můžeme přesvědčit např. v [Ahuj93], tato „omezení“ nás při řešení problémů ve skutečnosti nijak neomezují. Pomocí různých transformací sítí (viz [Palu03c]) jsme vždy schopni splnit požadavky modelu. Algoritmy řešící úlohu nejkratší cesty můžeme rozdělit do několika kategorií. Na algoritmy, které:

- Naleznou nejkratší cestu z jednoho uzlu do všech ostatních uzlů, přičemž délky hran jsou nezáporné (např. Dijkstrův algoritmus [Dijk59]).
- Naleznou nejkratší cestu z jednoho uzlu do všech ostatních uzlů sítě, jejíž délky hran jsou libovolné (tedy i záporné – např. Bellmanův algoritmus [Bell58]).
- Naleznou nejkratší cestu z každého uzlu do všech ostatních uzlů (např. Floydův algoritmus [Ahuj93]).

Nás budou zajímat pouze algoritmy z první skupiny, protože by nám dokonce stačilo nalezení nejkratší cesty mezi určenou dvojicí uzlů. Takový algoritmus ale neexistuje. Můžeme však stávající algoritmus upravit tak, že po nalezení cesty do cílového uzlu ukončí svůj běh.

V literatuře [Ahuj93], [Cook98], [Core91] se algoritmy první skupiny označují též jako *značky nastavující* (*label setting*). Základním algoritmem tohoto typu je *Dijkstrův algoritmus.* Dijkstra byl jedním z lidí, kteří nezávisle na sobě tento algoritmus publikovali [Dijk59]. Původní Dijkstrův algoritmus pracuje v čase $O(n^2)$. Časové omezení $O(n^2)$ pro Dijkstrův algoritmus je nejlepší možné v případě úplných hustých sítí (tj. $m = \Omega(n^2)$), ale může být lepší pro řídké sítě, které se v praxi vyskytují mnohem častěji. Úzkým místem Dijkstrova algoritmu je volba uzlu s minimální distanční značkou. Byly proto vyvinuty sofistikované datové struktury a postupy ([Dial69], [John77], [Fred84]), které nám umožňují efektivnější volbu uzlu. Podařilo se nám tak buď dramaticky snížit dobu běhu algoritmu, nebo zlepšit časovou složitost ([Dvor02]). My se zaměříme konkrétně na Dialův algoritmus [Dial69], který by podle [Ahuj93] měl dosahovat výborných výsledků v praxi a na haldové modifikace původního Dijkstrova algoritmu – jmenovitě s binární haldou ([John77]), *d*-haldou ([John77]) a Fibonacciho haldou ([Fred84]).

2.2 Výsledky empirických testů a vyhodnocení

Mezi výše jmenovanými algoritmy jsme hledali nejvýkonnější implementaci v Delphi ([Teix02]). Rozhodnutí bylo provedeno na základě empirické analýzy. V tabulce na obrázku 2.1 najdete srovnání algoritmů z hlediska časové složitosti. Nejlepší omezení počtu operací pro nejhorší případ sítě má Dijkstrův algoritmus s Fibonacciho haldou. Časová složitost však může být ovlivněna nějakou „patologickou" instancí problému a skutečný výkon algoritmu pro většinu instancí může být mnohem lepší. Výsledky empirické analýzy jsou shrnuty v tabulce na obrázku 2.2. Kompletní testy a popis algoritmů a datových struktur najdete v [Palu03c], zde nám postačí dosažené pořadí jednotlivých algoritmů na testovaných sítích. Sítě byly vygenerovány pomocí programu *GVS* (příloha [Palu03c]). Vidíme, že přesvědčivě nejlepších výsledků dosahoval právě Dialův algoritmus, jehož časová složitost je pouze pseudopolynomiální.

Název algoritmu	Publikován	Časová složitost
Dijkstr	1959	$O(n^2)$
Dial	1969	$O(m + nC)$
Dijkstr s binární haldou	1977	$O(m \log n)$
Dijkstr s *d*-haldou	1977	$O(m \log_d n)$, $d = m/n$
Disjkstr s Fibonacciho haldou	1984	$O(m + n \log n)$

Obrázek 2.1 Algoritmy nejkratší cesty.

Označení sítě	Relace uzlů a hran	Dijkstr	Dial	Binární halda	*d*-halda	Fibonacciho halda
Mesh	$m \sim 3n$	5	1	3	2	4
DEL	$m \sim 6n$	5	1	3	2	4
RandomMesh	$m \sim 20n$	5	1	3	2	4

Obrázek 2.2 Přehled dosažených umístění algoritmů nejkratší cesty v jednotlivých testech.

Bohužel rychlost Dialova algoritmu je velmi závislá od hodnoty C. Proto byla provedena i další série testů (viz [Palu03c]) – tentokrát jen na jednom typu sítě o stejném rozsahu, ale s rostoucí hodnotou C. Porovnávány byly již jen dva nejlepší algoritmy z minulého testu – Dial a Dijkstr s *d*-haldou. Měřením bylo zjištěno, že přibližně do hodnoty $C = 25000$ je Dialův algoritmus stále nejrychlejší. Pro vyšší hodnoty C ale rychle ztrácí a stává se brzo nejpomalejším algoritmem v testu. Proto pro použití v praxi nebo v rámci jiných komplexnějších algoritmů je mnohem vhodnější použít variantu Dijkstrova algoritmu s *d*-haldou, jehož výkonnost není ovlivněna hodnotou C.

3 Maximální toky

Úloha maximálního toku (tj. přepravy maximálního množství „hmoty" při omezené propustnosti přepravních linek) je speciálním případem obecnější úlohy toku s minimálními náklady. Základní teorie a algoritmy maximálních toků jsou dobře popsané v literatuře ([Ples83], [Ahuj93], [Goem94]). Pro řešení úlohy maximálního toku existují výkonné polynomiální algoritmy (Goldbergův [Gold85]). Vybrané

algoritmy byly opět implementovány v Delphi ([Teix02]) a empirickou analýzou jsme hledali nejvýkonnější implementaci, kterou pak použijeme v komplexnějších algoritmech. Algoritmy pro řešení maximálního toku můžeme využít i pro řešení úloh minimálního řezu ([Ples83], [Ahuj93], [Goem94]). Mezi oběma úlohami totiž existuje zajímavý vztah, který je pojí dohromady. Oba typy úloh jsou použitelné pro řešení řady praktických problémů. Přímo s úlohou maximálního toku se můžeme setkat při rozvrhování výroby pro více strojů stejného typu ([Fede86]), při distribuovaných výpočtech na dvouprocesorovém počítači ([Ston77]), při zaokrouhlování racionálních prvků matic a součtů sloupců a řad ([Bach66]), při návrhu počítačových přepojovaných sítí ([Jane96]) atd.

3.1 Matematický model

Nechť $G = (N, A)$ je síť s nezápornou kapacitou u_{ij} asociovanou s každou hranou $(i, j) \in A$. Nechť $U = \max \{u_{ij}: (i, j) \in A\}$. Podobně jako jsme uvedli dříve, seznam sousedních hran $A(i) = \{(i, k): (i, k) \in A\}$ obsahuje všechny hrany vycházející z uzlu i. Abychom mohli definovat úlohu maximálního toku, rozlišujeme v síti G dva speciální uzly: *zdrojový* (*source*) uzel s a *spotřební* (*destination*) uzel t. Jak už napovídá název úlohy, snažíme se najít maximální tok ze zdroje do spotřebiče, který splňuje kapacitní omezení hran a rovnováhu toku u všech uzlů (co do uzlu přiteče, musí z uzlu i odtéct). Úlohu můžeme obecně formulovat následovně:

$$\text{Maximalizuj } v \tag{3.1a}$$

za podmínek:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} v & pro\ i = s, \\ 0 & pro\ všechny\ i \in N - \{s,t\}, \\ -v & pro\ i = t. \end{cases} \tag{3.1b}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{pro každou } (i,j) \in A. \tag{3.1c}$$

Vektor $x = \{x_{ij}\}$ splňující (3.1b) a (3.1c) budeme označovat jako *tok* (přesněji jako *s – t tok)* a odpovídající hodnotu skalární proměnné v (*value*) jako *velikost toku.*

Aby byla úloha maximálního toku řešitelná pomocí prezentovaných algoritmů, musí síť splňovat několik předpokladů:

- Síť je orientovaná.
- Všechny kapacity jsou nezáporné a celočíselné.
- Síť neobsahuje orientovanou cestu z uzlu s do uzlu t složenou pouze z hran s nekonečnou kapacitou.
- Kdykoliv hrana (i, j) náleží do A, tak i hrana (j, i) náleží do A.
- Síť neobsahuje paralelní hrany (tzn. dvě nebo více hran se shodným počátečním a koncovým uzlem).

Podobně jako u úlohy nejkratší cesty neznamenají ani tentokrát tato omezení žádnou ztrátu na obecnosti modelu. Pomocí vhodných úprav a transformací ([Palu03c]) můžeme tato omezení odstranit. Spojení mezi úlohou maximálního toku a úlohou minimálního řezu vyjadřuje následující věta zformulovaná a dokázána Fordem ([Ford56]).

Věta 3.1 *Velikost maximálního s – t toku se rovná kapacitě minimálního s – t řezu.*

3.2 Výsledky empirických testů a vyhodnocení

Algoritmy řešící úlohu maximálního toku můžeme rozdělit do dvou základních skupin. První skupinu tvoří algoritmy založené na myšlence *zlepšující cesty*. Patří sem většina zde testovaných algoritmů

(Ford-Fulkerson ([Ford56]), Edmonds-Karp ([Ples83]), Dinic ([Dini99])). Druhá skupina algoritmů, známa jako *předtok šířící* algoritmy, je zastoupena Algoritmem tří Indů ([Palu01a]) a Goldbergovým algoritmem ([Gold85], [Palu01b]). Algoritmy patřící do druhé skupiny jsou mnohem obecnější, výkonnější a flexibilnější než algoritmy zlepšujících cest. Nejlepší předtok šířící algoritmy překonávají nejlepší algoritmy zlepšujících cest jak teoreticky, tak prakticky. Již z časové složitosti jednotlivých algoritmů (obrázek 3.1) vidíme, že omezení počtu iterací je příznivější pro předtok šířící algoritmy. Tato skutečnost se potvrdila i při empirických testech (obrázek 3.2). Původní verze Goldbergova algoritmu byla autorem upravena ([Palu01b]) do výkonnější podoby. Při programové realizaci původního Goldbergova algoritmu se ukázalo, že přeznačkovávat postupně jednotlivé uzly není příliš efektivní, že je mnohem lepší po nějakém vhodně zvoleném počtu iterací provést „hromadné“ přeznačkování všech uzlů. Následně provedenými testy byla tato skutečnost potvrzena. Touto úpravou se významně zlepšilo empirického chování algoritmu, přičemž ale časová složitost algoritmu zůstala stále $O(n^3)$. Jako nejvhodnější se nakonec ukázalo provádět „hromadné“ přeznačkování po provedení $(0.5 – 0.7)n$ iterací, kde n je počet uzlů.

Modifikovaná verze Goldbergova algoritmu dosáhla nejlepších výsledků v testech a byla proto zvolena pro použití v komplexnějších algoritmech. Kompletní testy a popis algoritmů najdete v [Palu00] a [Palu03c].

Název algoritmu	Publikováno	Časová složitost
Ford-Fulkerson (F-F)	1956	$qO(m)$, $q \in (0, \infty)$
Edmonds-Karp (nejkratší cesta) (E-K)	1972	$O(nm^2)$
Edmonds-Karp (cesta s maximální kapacitou) (E-K2)	1973	$O(n^2m \ln v)$
Dinic (D)	1972	$O(n^2m)$
Algorimus tří Indů (A3I)	1978	$O(n^3)$
Goldbergův algoritmus (G)	1985	$O(n^3)$
Modifikovaný Goldbergův algoritmus (MG)	2000	$O(n^3)$

Obrázek 3.1 Algoritmy maximálních toků.

Všechny současné nejlepší algoritmy pracují na podobném principu jako Goldbergův algoritmus, ale modifikují jeho „slabé“ místo, což je práce s frontou FIFO při výběru aktivního uzlu. Jedna z dalších modifikací Goldbergova algoritmu nejdříve prozkoumává aktivní uzly s největší hodnotou distanční značky (highest-label preflow algorithm [Gold93]), v jiné modifikaci se naopak upřednostňují aktivní uzly s dostatečně velkým přebytkem a v rámci této skupiny uzlů se volí uzel s nejmenší hodnotou distanční značky (excess scaling algorithm [Ahuj89]) a v poslední existující modifikaci se pro ukládání aktivních uzlů využívá speciální datové struktury – dynamických stromů ([Slea83]). Tyto nové algoritmy zpravidla přináší zlepšení ve formě lepší časové složitosti, ale jejich praktický přínos ve formě skutečného zvýšení rychlosti výpočtu je sporný.

Označení sítě	Relace uzlů a hran	Název algoritmu						
		F-F	E-K	E-K2	D	A3I	G	MG
Cheryian	$m \sim n$	6	2	7	3	4	5	1
Gold	$m \sim n$	6	5	7	4	2	1	3
Řídký	$m \sim 2n$	7	2	4	5	6	3	1
Síť	$m \sim 3n$	6	7	5	3	2	4	1
Random	$m \sim 3n$	5	7	4	2	3	6	1
Čtverec	$m \sim 6n$	6	7	5	4	3	2	1
DoubleExp	$m \sim 8n$	7	6	5	4	3	2	1
Bipart	$m \sim n^2/4$	5	7	6	3	4	1-2	1-2
Hustý	$m \sim n^2/2$	7	6	5	1	3	4	2

Obrázek 3.2 Přehled dosažených umístění jednotlivých algoritmů maximálních toků.

V rámci této práce realizovaná implementace modifikovaného Goldbergova algoritmu patří k nejrychlejším ve své třídě a umožňuje řešit rozsáhlé úlohy (tisíce uzlů a hran) v krátkém čase (v řádu sekund). V [Ahuj97] se taktéž věnovali podrobnému testování doposud vyvinutých algoritmů pro úlohu maximálního toku a relaci mezi skutečným výkonem v praxi a časovou složitostí. Nejlepších výsledků dosáhla *highest label*

verze Goldbergova algoritmu ([Gold93]) těsně následována FIFO verzí, která byla použita i v této práci. Ostatní algoritmy následovaly už s větším odstupem.

4 Toky s minimálními náklady

Jedná se o variantu úlohy víceproduktových toků pouze s jedním produktem. Je to základní úloha toků v sítích ([Ples83, Ahuj93]). Sama je opět obecnějším případem již popsaných úloh (maximální tok a nejkratší cesta). Problém, který tato úloha řeší, je následující: Chceme najít nejlevnější přenos produktu skrz síť tak, abychom naplnili poptávku po produktu u některých uzlů při využití nabídky produktu u jiných uzlů. Tento model má řadu použití v praxi. Úloha toku s minimálními náklady se objevuje téměř ve všech oblastech průmyslu. Rozhodně úlohy toků s minimálními náklady pronikají do praxe. S problémy řešitelnými pomocí algoritmů hledajících tok s minimálními náklady se setkáme při projektování distribučních systémů, v lékařské diagnostice, při přepravě zboží, při výrobě zboží, při kapacitním plánování, řízení lidských zdrojů atd. Konkrétně jmenujme například rekonstrukci levé srdeční komory z rentgenových snímků ([Slum82]), návrh evakuačního plánu budovy ([Chal82]), optimální vytížení letadla na trase s mezipřistáními ([Gupt85]) a nejčastěji se vyskytující distribuční úlohu ([Glov77]).

4.1 Matematický model

Nechť $G = (N, A)$ je orientovaná síť s náklady c_{ij} a kapacitami u_{ij} asociovanými s každou hranou $(i, j) \in A$. Dále s každým uzlem i asociujeme číslo $b(i)$, které určuje jeho *zásobu* (*supply*) nebo *požadavek* (*demand*) po produktu podle toho, zda $b(i) > 0$ nebo $b(i) < 0$. Úlohu toku s minimálními náklady můžeme stanovit následovně:

$$\text{Minimalizuj} \qquad z(x) = \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{4.1a}$$

za podmínek

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{pro všechny } i \in N, \tag{4.1b}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{pro všechny } (i, j) \in A \tag{4.1c}$$

Nechť C je označení pro největší náklady na kterékoliv hraně. Dále nechť U je označení pro největší hodnotu zásoby/požadavku nebo kapacity hrany (mimo ∞). Předpokládáme, že dolní hranice l_{ij} pro tok hranou jsou nulové. Dále zavedeme ještě několik předpokladů:

- Všechna data (náklady, zásoba/požadavek, kapacita) jsou celočíselná.
- Síť je orientovaná.
- Když zásoby/požadavky u uzlů splňují podmínku $\sum_{i\in N} b(i) = 0$, pak úloha toku s minimálními náklady má přípustné řešení.
- Předpokládáme, že každá síť G obsahuje orientovanou cestu s neomezenou kapacitou (tj. každá hrana na cestě má nekonečnou kapacitu) mezi každou dvojicí uzlů.
- Všechny náklady na hranách jsou nezáporné.

Podobně jako u předchozích úloh neznamenají uvedené předpoklady žádné omezení na obecnosti modelu. Lze je „obejít“ vhodnými transformacemi sítě([Palu03c]).

4.2 Podmínky optimality

Definování podmínek, za kterých je řešení úlohy toku s minimálními náklady optimální, je pro nás užitečné z několika důvodů. Poskytuje nám kontrolu, zda získané řešení je skutečně optimální. Často jsme schopni na základě podmínek optimality navrhnout algoritmus pro řešení dané úlohy. Ukažme si nyní několik různých (ale ekvivalentních) podmínek optimality pro úlohu toku s minimálními náklady. Důkazy následujících vět najdete např. v [Ahuj93].

Věta 4.1 *(Podmínky optimality záporného cyklu) Přípustné řešení x^* je optimálním řešením úlohy toku s minimálními náklady tehdy a pouze tehdy, když splňuje podmínky optimality záporného cyklu, tzn. že residuální síť $G(x^*)$ neobsahuje (orientovaný) cyklus se zápornou cenou.*

Věta 4.2 *(Podmínky optimality redukovaných nákladů) Přípustné řešení x^* je optimálním řešením úlohy toku s minimálními náklady tehdy a pouze tehdy, když nějaká množina potenciálů uzlů π splňuje následující podmínky optimality redukovaných nákladů:*

$$c_{ij}^{\pi} \geq 0 \qquad \text{pro každou hranu } (i, j) \text{ v } G(x^*). \tag{4.2}$$

Věta 4.3 *(Komplementární podmínky optimality) Přípustné řešení x^* je optimální řešení úlohy toku s minimálními náklady tehdy, když pro nějakou množinu potenciálů uzlů, redukovaných nákladů a hodnot toku splňuje následující komplementární podmínky optimality pro každou hranu $(i, j) \in A$:*

$$\textit{Je-li } c_{ij}^{\pi} > 0, \text{ potom } x_{ij}^* = 0. \tag{4.3a}$$

$$\textit{Je-li } 0 < x_{ij}^* < u_{ij}, \text{ potom } c_{ij}^{\pi} = 0. \tag{4.3b}$$

$$\textit{Je-li } c_{ij}^{\pi} < 0, \text{ potom } x_{ij}^* = u_{ij}. \tag{4.3c}$$

Na podmínkách optimality záporného cyklu je založen algoritmus stornovaných cyklů ([Klei67]), na komplementárních podmínkách optimality je založen cenově-zjemňující algoritmus [Röck80], ostatní algoritmy jsou založeny na podmínkách optimality redukovaných nákladů (algoritmus postupných nejkratších cest [Busa61], primárně-duální algoritmus [Ford62], kapacitně-zjemňující [Orli93], relaxační algoritmus [Bert88]).

4.3 Pseudopolynomiální a polynomiální algoritmy

Úloha toku s minimálními náklady je již natolik složitá, že pro její řešení uvažujeme již i pseudopolynomiální algoritmy, jejichž doba řešení může s velikostí úlohy růst exponenciálně, ale na druhou stranu jejich struktura bývá velmi jednoduchá ([Palu03a]). Podle [Ahuj93] by měl k nejlepším algoritmům patřit právě pseudopolynomiální relaxační algoritmus založený na myšlence Lagrangeovy relaxace ([Fish81]). V duchu teorie o výpočetní složitosti není použití pseudopolynomiálního algoritmu úplně uspokojivé. Nemáme dobré teoretické záruky, že algoritmus bude řešit dostatečně rychle všechny typy problémů, se kterými bychom se mohli setkat. V této situaci se nabízí otázka, zda můžeme vymyslet algoritmy, které jsou polynomiální v běžných parametrech problému, tj. v počtu uzlů n, počtu hran m, $\log U$ a $\log C$. Jak získáme polynomiální algoritmus? K získání algoritmu pracujícího v polynomiálním čase použijeme myšlenku známou jako *zjemňování* (*scaling*, [Edmo72]), konkrétně použijeme zjemňování dat s kapacitou a zjemňování dat s cenou. Vytvoříme tak dva algoritmy pracující v polynomiálním čase: (1) kapacitně zjemňující algoritmus, který je zjemňující verzí algoritmu postupných nejkratších cest a (2) cenově zjemňující algoritmus, který je zobecněnou verzí Goldbergova algoritmu pro hledání maximálního toku v síti. Zjemňování je významná idea, pomocí které byla vylepšena řada algoritmů v kombinatorické optimalizaci včetně algoritmů řešících úlohu toku s minimálními náklady ([Röck80], [Gabo85], [Ahuj93], [Palu03b]). Na zjemňovací algoritmy se můžeme dívat následujícím způsobem. Začneme s podmínkami optimality pro úlohu sítových toků, kterou zkoumáme, ale místo přesného vyjádření těchto podmínek, generujeme aproximativní řešení, které může porušit jednu (nebo více) podmínek o maximální množství Δ. Na začátku tím, že zvolíme Δ dostatečně velké, například C nebo U, snadno můžeme najít počáteční řešení, které splňuje uvolněné podmínky optimality. Potom změníme parametr Δ na $\Delta/2$ a provedeme reoptimalizaci, aby aproximativní řešení nyní porušovalo podmínky optimality nanejvýše o $\Delta/2$. Opakujeme tento postup – opět reoptimalizujeme, dokud aproximativní řešení neporušuje podmínky optimality o maximálně $\Delta/4$ atd. Tato strategie řešení je docela flexibilní a vede k různým algoritmům v závislosti na tom, které podmínky optimality jsme uvolnili a jak jsme provedli reoptimalizaci.

Z pseudopolynomiálních algoritmů byly zkoumány a implementovány následující:

- algoritmus stornovaných cyklů ASC ([Klei67])
- algoritmus postupných nejkratších cest APNC ([Busa61])
- primárně-duální algoritmus P-DA ([Ford62])
- relaxační algoritmus RA ([Bert88])

Z polynomiálních algoritmů jsme se zaměřili na:

- kapacitně zjemňující algoritmus KZA (polynomiální verze APNC [Orli88])
- cenově zjemňující algoritmus CZA (zobecnění Goldbergova algoritmu [Röck80])

4.4 Výsledky empirických testů a vyhodnocení

Oproti dřívějším testům jsou již algoritmy hledající tok s minimálními náklady komplexnější a jejich součástí mohou být i jiné jednodušší algoritmy, jako je algoritmus pro nalezení nejkratší cesty nebo algoritmus pro nalezení maximálního toku. Těmito algoritmy jsme se zabývali v předchozích kapitolách. Jejich volbou můžeme značně ovlivnit výslednou výkonnost hlavního algoritmu. Protože se můžeme opřít o výsledky testů skupiny algoritmů pro maximální tok (kapitola 3.2) i pro nejkratší cestu (kapitola 2.2), byl pro nalezení maximálního toku použit modifikovaný Goldbergův algoritmus ($O(n^3)$) a pro nalezení nejkratší cesty Dijkstrův algoritmu pracující s d-haldou ($O(m\log_d n)$, kde $d = m/n$).

Název algoritmu	Publikováno	Časová složitost
Algoritmus stornovaných cyklů (ASC)	1967	$O(nm^2CU)$
Algoritmus postupných nejkratších cest (APNC)	1961	$O(nmU\log_d n)$, $d = m/n$
Primárně-duální algoritmus (P-DA)	1962	$O(n^4\min\{C, U\})$
Relaxační algoritmus (RA)	1988	$O(m^2nCU^2)$
Kapacitně zjemňující algoritmus (KZA)	1988	$O(m^2\log_d n \log U)$, $d = m/n$
Cenově zjemňující algoritmus (CZA)	1980	$O(n^2m\log(nC))$

Obrázek 4.1 Přehled testovaných algoritmů.

Z hlediska využití jiných algoritmů můžeme algoritmy řešící úlohu toku s minimálními náklady rozdělit do tří skupin. První skupinu tvoří algoritmy, které nepotřebují žádný jiný algoritmus ke své činnosti (RA, CZA), do druhé skupiny řadíme algoritmy obsahující řešení úlohy nejkratší cesty (APNC, KZA) a do třetí skupinu patří algoritmy požadující jak nalezení nejkratší cesty, tak i maximálního toku (ASC, P-DA).

Přibližný odhad chování algoritmu můžeme získat z jeho časové složitosti. Přehled časových složitostí testovaných algoritmů najdete v tabulce na obrázku 4.1. Bude zajímavé konfrontovat tyty údaje s reálnými výsledky v testech. Jako vodítko nám může posloužit tabulka na obrázku 4.2, která shrnuje dosažené pořadí jednotlivých algoritmů v testech. Kompletní výsledky testů naleznete v [Palu03c]. Cílem testů bylo najít vhodného kandidáta pro řešení víceproduktového toku s minimálními náklady. Z naměřených dat je zřejmé, že při řešení VT s minimálními náklady použijeme APNC.

Označení sítě	Relace uzlů a hran	Název algoritmu					
		ASC	APNC	P-DA	RA	KZA	CZA
Cheryian	$m \sim n$	1-2	1-2	3	5	4	6
Mesh	$m \sim 3n$	6	1	3	4	2	5
SquareMesh	$m \sim 6n$	6	1	2	4	3	5
DoubleExpLine	$m \sim 10n$	6	1	3	5	2	4
RandomMesh	$m \sim 20n$	6	1	2	4	3	5
Dense	$m \sim n^2/2$	6	2	3	4	1	5

Obrázek 4.2 Přehled dosažených umístění algoritmů řešících úlohu toků s minimálními náklady.

5 Víceproduktové toky

Úloha víceproduktových toků je speciální třídou lineárního programování a může být použita jako model pro řešení mnoha praktických problémů, z nichž některé vyžadují, aby tok byl celočíselný. Jak jsme viděli v předchozích kapitolách, pro řešení jednoproduktových tokových úloh byly vyvinuty velmi efektivní algoritmy. Navíc je u nich i garantována celočíselnost optimálního toku, pokud jsou celočíselné i zásoby/požadavky u uzlů a kapacity u hran. Avšak v důsledku složitější struktury omezení u víceproduktové úlohy nemůžeme použít klasické metody lineárního programování pro nalezení celočíselného optima. Pro obrovskou výpočetní náročnost se nehodí ani použití existujících celočíselných programovacích technik. Jako schůdné řešení se jeví v současné době použití heuristických metod. V této práci se zaměříme hlavně na stochastické heuristické metody ([Mich96], [Reev93]), mezi které řadíme mimo jiné genetické algoritmy, simulované žíhání a tabu search (zakázané prohledávání).

V předchozích kapitolách jsme vždy řešili úlohy s jediným produktem, který jsme chtěli poslat ze zdroje do spotřebiče nějakým optimálním způsobem, např. podél nejkratší cesty nebo s minimálními náklady. V praxi se ale většinou vyskytuje několik fyzických produktů, dopravních prostředků nebo zpráv, každý z nich má své vlastní omezení pro tok v síti a společně sdílejí pouze síť. Například v telekomunikačních aplikacích telefonní hovor mezi danou dvojicí uzlů definuje jeden produkt v základní síti. Kdyby se produkty vzájemně nijak neovlivňovaly, řešili bychom každou úlohu samostatně s využitím již popsaných algoritmů. V opačném případě ale nebudou jednotlivé jednoproduktové úlohy nezávislé. Pokud tedy chceme najít optimální tok, musíme úlohy řešit ve vzájemném souladu.

Úloha víceproduktových toků může být použita pro řešení mnoha praktických problémů jako jsou např. distribuční problémy ([Kenn80]), multiperiodický přiřazovací problém ([Arro86]), problém obměny zařízení ([Agga95]), úlohy v telekomunikačních sítích ([Riou92]), při návrhu VLSI čipů ([Kort88]), při skladování sezónních výrobků ([Jewe57]). Zatímco v některých aplikacích není celočíselnost toku nutná [Kenn80], v jiných je naopak požadována [Arro86].

5.1 Matematické modely víceproduktových toků

Úloha víceproduktových toků má dvě základní formy:

- maximální víceproduktový tok
- víceproduktový tok s minimálními náklady

Nejdříve si uveďme model pro úlohu maximálního víceproduktového toku. Předpokládejme, že s^k a t^k pro $k = 1, 2, \ldots, K$ představují zdroj a spotřebič pro produkt k. Potom úlohu víceproduktového maximálního toku (ve formě s uzly a hranami) můžeme definovat následujícím způsobem:

$$\text{Maximalizuj} \sum_{1 \le k \le K} v^k \tag{5.1a}$$

za podmínek

$$\sum_{\{j:(i,j)\in A\}} x_{ij}^k - \sum_{\{j:(j,i)\in A\}} x_{ji}^k = \begin{cases} v^k & i = s^k \\ 0 & i \ne (s^k, t^k) \\ -v^k & i = t^k \end{cases} \qquad i \in N, \quad k = 1, 2, \ldots, K \tag{5.1b}$$

$$\sum_{1 \le k \le K} x_{ij}^k \le u_{ij} \qquad (i,j) \in A \tag{5.1c}$$

$$0 \le x_{ij}^k \le u_{ij}^k \qquad (i,j) \in A \quad \text{a} \quad k = 1, 2, \ldots, K \tag{5.1d}$$

Ve výše uvedené formulaci x_{ij}^k je tok produktu k přes hranu (i, j) a v^k je celkový tok produktu k, který opustí zdrojový uzel s^k a dosáhne spotřebiče t^k. Pro každou hranu (i, j) je také zavedeno společné omezení kapacity u_{ij}. Kapacita u_{ij} představuje množství nějakého společného zdroje (např. kapitálu, práce, zařízení,

místa atd.), který je společně využíván všemi produkty. Kromě toho je zavedena i kapacita u_{ij}^k, která reprezentuje maximální povolenou hodnotu toku *k*-tého produktu přes hranu (*i*, *j*) danou specifickými omezeními daného produktu (omezené zdroje, technologické limity). Dále musí být pro každý produkt a pro každý uzel rozdíl přítoku a odtoku uzlu s výjimkou zdrojů a spotřebičů nulový.

Podobně je definována i úloha víceproduktového toku s minimálními náklady:

$$\text{Minimalizuj} \sum_{1\le k\le K} c^k x^k \tag{5.2a}$$

za podmínek

$$\sum_{(j:(i,j)\in A)} x_{ij}^k - \sum_{(j:(j,i)\in A)} x_{ji}^k = b_i^k \qquad i \in N \text{ a } k = 1, 2, ..., K, \tag{5.2b}$$

$$\sum_{1\le k\le K} x_{ij}^k \le u_{ij} \qquad (i,j) \in A, \tag{5.2c}$$

$$0 \le x_{ij}^k \le u_{ij}^k \qquad (i,j) \in A \text{ a } k = 1, 2, ..., K. \tag{5.2d}$$

V této formulaci c^k je vektor jednotkových nákladů produktu *k* a $b_i^k > 0$ ($b_i^k < 0$) je zásoba (požadavek) produktu *k* u uzlu *i*. Poznamenejme, že musí platit $\sum_{i\in N} b_i^k = 0$ pro $k = 1, 2, \ldots, K$.

Na rozdíl od formulace víceproduktového maximálního toku (5.1) se může v této formulaci vyskytnout více zdrojů a spotřebičů u každého produktu. Není ovšem problém zkonstruovat novou síť s doplňkovými uzly a hranami, která bude představovat ekvivalentní úlohu víceproduktového toku s minimálními náklady s jedním zdrojem a spotřebičem pro každý produkt. Stačí přidat do sítě $2m$ uzlů (S^k, T^k) pro $k = 1, 2, \ldots, K$ a hrany (S^k, i) s kapacitou b_i^k pro všechna i s $b_i^k > 0$ a hrany (i, T^k) s kapacitou $-b_i^k$ pro všechna i s $b_i^k < 0$. Uzel S^k budeme označovat jako *super zdroj* a uzel T^k jako *super spotřebič*. Jednotkové náklady na těchto hranách budou pro produkt *k* nulové a nekonečné pro ostatní produkty. Nechť *N'* je množina nových uzlů a *A'* je množina nových přidaných hran. Nechť V^k je celkový tok, který by měl opustit uzel S^k pro $k = 1, 2, \ldots, K$, určený takto:

$$V^k = \sum_{i\in D} b_i^k = \sum_{i\in D'} (-b_i^k) \ ,$$

kde *D* je množina uzlů *i* s $b_i^k > 0$ a *D'* je množina uzlů s $b_i^k < 0$.
Odpovídající síťový model pro síť $G' = (N \cup N', A \cup A')$ je:

$$\text{Minimalizuj} \sum_{1\le k\le K} c^k x^k \tag{5.3a}$$

za podmínek

$$\sum_{\{j:(i,j)\in A\}} x_{ij}^k - \sum_{\{j:(j,i)\in A\}} x_{ji}^k = \begin{cases} V^k & i = S^k \\ 0 & i \ne (S^k, T^k) \\ -V^k & i = T^k \end{cases} \qquad i \in (N \cup N') \quad k = 1, 2, ..., K \tag{5.3b}$$

$$\sum_{1\le k\le K} x_{ij}^k \le u_{ij} \qquad (i,j) \in A, \tag{5.3c}$$

$$0 \le x_{ij}^k \le u_{ij}^k \qquad (i,j) \in (A \cup A') \text{ a } k = 1, 2, ..., K. \tag{5.3d}$$

5.2 Metody řešení

Pro jednoproduktové úlohy byly již vyvinuty velmi efektivní metody (jak jsme poznali v předchozích kapitolách) pro nalezení optimálního toku. Díky jejich speciální struktuře (unimodulární matici omezení) je garantováno, že optimální tok bude celočíselný, pokud i kapacity na hranách budou celočíselné. Úloha

víceproduktových toků má ale složitější strukturu omezení a optimální tok nemusí být nutně celočíselný. Pokud toky nejsou celočíselně omezeny, jedná se o tzv. úlohy „nepřetržitého“ toku. Tento typ úloh můžeme řešit pomocí tří základních typů algoritmů, které byly vyvinuty:

- Rozdělující algoritmy (*partitioning algorithms*)
- Algoritmy s dekompozicí podle ceny (*price-directive decomposition*)
- Algoritmy s dekompozicí podle kapacity (*capacity-directive decomposition*)

Rozdělující algoritmy jsou založeny na speciálně upravené simplexově metodě a neprovádí dekompozici na jednoproduktové úlohy. Byla vyvinuta celá řada algoritmů tohoto typu. Zájemcům o tento způsob řešení úlohy víceproduktových toků doporučuji [Kenn80].

Algoritmy provádějící dekompozici podle ceny jsou založeny na nejznámějším algoritmu tohoto typu – Dantzing-Wolfeho dekompoziční metodě [Dant61], která rozdělí úlohu na hlavní (řídící) část, do které jsou zahrnuta společná (komplikující) omezení na hranách a na podprogram, který zahrnuje jednotlivé jednoproduktové úlohy. Nejdříve řídící část algoritmu stanoví „doplňkové“ ceny pro hrany se společným omezením kapacity a potom se použije podprogram pro jednotlivé produkty. Získané výsledky se předají zpět řídící části algoritmu, která provede vhodnou úpravu cen. Podstatné je, že algoritmus udržuje stále přípustnost řešení. Do této skupiny algoritmů můžeme zařadit i aproximační Lagrangeův algoritmus ([Ahuj93], [Fish81]), ve kterém k celkovým omezením toku přiřadíme tzv. Lagrangeovy operátory a přeneseme tyto omezení do účelové funkce. Dostaneme tak nový problém, jehož optimální řešení bude odpovídat původnímu problému.

Algoritmy provádějící dekompozici podle kapacity ([Geof74]) eliminují omezení společné hranové kapacity rozdělením této kapacity mezi jednotlivé produkty. Výsledkem je množina vzájemně nezávislých jednoproduktových úloh. Bohužel nalezení optimálního rozdělení kapacity mezi jednotlivé produkty je obtížnou úlohou a doposud nebyla nalezena žádná efektivní metoda. K řešení tohoto problému byly vyvinuty iterační algoritmy, které se liší v tom, jak testují optimalitu řešení a jak stanovují rozdělení společné kapacity mezi produkty.

♦

Existují i algoritmy pro řešení celočíselného víceproduktového toku. Arronson navrhl speciální víceproduktový model pro řešení multiperiodického přiřazovacího problému pomocí algoritmu větví a mezí ([Arro86]). Pro řešení speciální třídy víceproduktových toků definované planárními sítěmi navrhli Hassin a Lomonosov algoritmy a stanovili jisté vlastnosti týkající se celočíselnosti ([Hass84], [Lomo83]).

Protože úloha víceproduktových toků patří mezi tzv. NP-těžké úlohy (s lineárním růstem rozsahu problému roste její výpočetní náročnost exponenciálně), můžeme výše uvedenými metodami řešit pouze úlohy malého rozsahu a s nízkým počtem produktů. Chceme-li řešit i rozsáhlejší úlohy, musíme použít některý z heuristických algoritmů. Zaplatíme za to ale ztrátou jistoty nalezení optima. Deterministickou heuristiku inspirovanou metodou rozkladu podle kapacit navrhl pro řešení celočíselných víceproduktových toků A. K. Aggarwal et al. ([Agga95]). My se v této práci zaměříme na použití stochastických heuristik pro nalezení celočíselného víceproduktového toku. Je zřejmé, že víceproduktový tok závisí na pořadí, v kterém je přidělována volná kapacita sítě jednotlivým produktům. Nalezení optimálního pořadí produktů je však složitý kombinatorický problém a nebyl doposud ještě efektivně vyřešen. My jsme použili pro nalezení optimálního pořadí algoritmy založené na vybraných stochastických heuristických metodách. Pokud bychom použili „silový“ přístup k řešení a pokusili se prohledat prostor všech možných řešení, který je v našem případě dán permutací indexu K, rostl by počet nutných výpočtů s počtem produktů exponenciálně. Např. uvažujme síť typu Mesh ($m \sim 3n$) s 3000 uzlů a s 1, 6 a 10 produkty. Když vezmeme jako základ dobu výpočtu maximálního toku pro jeden produkt při použití nejvýkonnějšího modifikovaného Goldbergova algoritmu (výsledky převzaty z [Palub03], AMD Athlon 1900+), tak pro 1 produkt je doba výpočtu přibližně 187 ms, pro 6 produktů je to již 13.5 minuty a pro 10 produktů 78 dní.

5.3 Heuristické metody

V operačním výzkumu můžeme heuristické metody charakterizovat jako metody pro hledání „dobrých“ řešení (tj. řešení blízkých optimálnímu) pomocí intuitivního postupu a při rozumných výpočetních nákladech. Heuristiky proto používáme pro řešení složitých optimalizačních úloh (jako je i úloha víceproduktových toků), u kterých nejsme schopni při dnešní úrovni výpočetní techniky rozumně uplatnit existující exaktní algoritmy (pokud existují). Heuristické metody nám nezaručují nalezení optimálního řešení a obvykle ani možnost stanovení, jak daleko je určité přípustné řešení od optimálního řešení. Výhodou heuristických metod je, že jsou velmi obecně formulované, a tudíž snadno aplikovatelné na širokou škálu optimalizačních problémů. Samotné heuristické metody jsou poměrně snadno naprogramovatelné (na rozdíl od samotného problému, který řeší).

Heuristické optimalizační metody se používají pro optimalizaci mnohoparametrových funkcí s divokým průběhem, tj. s mnoha extrémy nebo neznámým gradientem. Standardní gradientové metody (např. nejprudšího spádu, sdružených gradientů, proměnné metriky) nebo negradientové metody (např. simplexová) nejsou vhodnými přístupy, protože požadujeme nalezení globálního extrému funkce s mnoha lokálními extrémy. Tyto metody obvykle konvergují jen k extrému blízko startovního bodu a tento extrém už nejsou schopné opustit. Tento nedostatek lze odstranit jejich randomizací tak, že se opakovaně spustí z různého počátečního řešení a za výsledné optimum se vezme nejlepší výsledek. Stochastičnost tohoto postupu spočívá pouze v náhodném výběru počátečního řešení, ale následovně použitý optimalizační algoritmus je striktně deterministický.

Stochastické heuristické optimalizační metody si obvykle zachovávají svoji stochastičnost v celém průběhu optimalizačního procesu a ne jen ve výběru počátečního řešení.
Evoluční proces prohledávání prostoru potenciálních řešení vyžaduje rovnováhu dvou cílů:

- co nejrychleji najít nejbližší (většinou lokální) optimum v okolí výchozího bodu,
- co nejlépe prohledat prostor všech řešení.

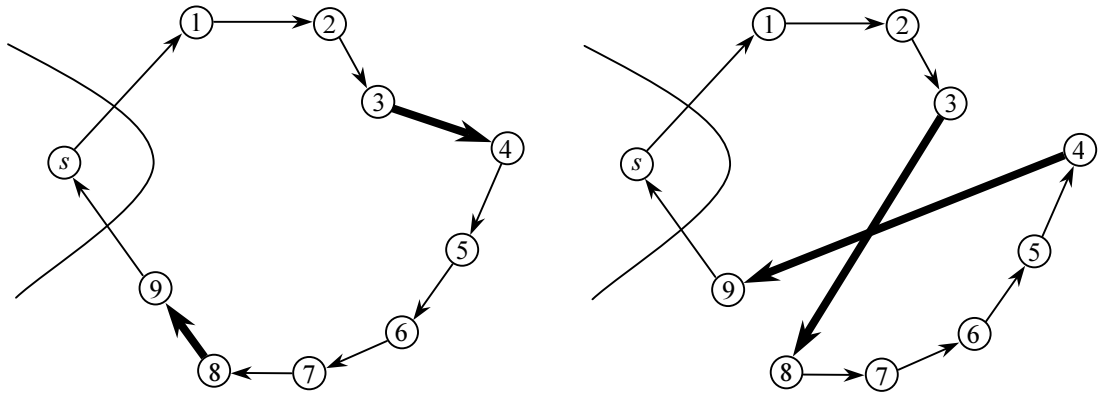
Pro řešení úlohy víceproduktových toků byly navrženy tři heuristické algoritmy založené na třech základních stochastických heuristických metodách – tabu search, simulované žíhání a genetické algoritmy (popsány v [Reev93], [Mich96], [Klap01], [Šeda98], [Glov97], [Otte89], [Gold89], [Holl75]). Význačným rysem těchto metod je, že jsou založeny na myšlence simulace některých v přírodě se vyskytujících procesů. Tato myšlenka se při řešení složitých problémů ukázala jako velmi cenná. Simulované žíhání bylo původně zavedeno jako analogie termodynamických procesů. Tabu search nachází některé ze svých motivací v pokusech imitovat „inteligentní“ procesy při využití jistého druhu „paměti“. Genetické algoritmy koncipují proces řešení problémů jako analogii vývoje genetických struktur. Pro úplnost ještě dodejme, že existuje ještě jedna metoda z této oblasti, která ale nebyla uplatněna v této práci, a to neuronové sítě. Objevují se také úspěšné pokusy o konstrukci hybridních metod, které nějakým způsobem kombinují myšlenky několika heuristických přístupů.

Přípustná řešení, účelová funkce a relace sousedství

Pro problém víceproduktových toků je přípustným řešením jistá posloupnost jednoproduktových problémů, ve které se každý produkt může vyskytnout pouze jednou. Tato forma vyjádření přípustného řešení odpovídá permutačnímu kódování. Posloupnost je zde tvořena permutací indexu K, kde K je počet produktů.

Pro každé přípustné řešení lze vypočítat hodnotu účelové funkce. Účelová funkce stanovuje pro danou posloupnost ohodnocení nalezeného toku. Pořadí produktů v přípustném řešení (chromozomu) bude určovat sled, ve kterém budeme řešit jednoproduktové úlohy odpovídající danému produktu. Hodnota účelové funkce potom bude odpovídat součtu nalezených jednoproduktových řešení. Účelová funkce je tedy kritériem kvality každého přípustného řešení. Cílem optimalizace pomocí heuristických metod je najít takové přípustné řešení, které má nejlepší hodnotu účelové funkce.

Pro použití heuristických metod potřebujeme definovat určitou relaci sousedství, která pro každé přípustné řešení *x* umožňuje pomocí množiny transformací *S(x)* stanovit jeho jisté okolí *U(x)* jako množinu přípustných řešení sousedících s *x*. Například přípustným řešením je sekvence jednoproduktových úloh 123456. Relaci sousedství můžeme jednoduše definovat například jako prohození pořadí dvou sousedních úloh. Potom by naše řešení 123456 mělo pět sousedních řešení: 213456, 132456, 124356, 123546, 123465. Způsobů, jak definovat relaci sousedství, lze vymyslet mnoho (výměna libovolných dvou, prohození tří, otočení pořadí části řetězce, atd.). Při stanovení množiny sousedních řešení je důležité splnění podmínky dosažitelnosti, která požaduje, aby každé řešení bylo dosažitelné z libovolného jiného řešení postupnou aplikací vztahu sousednosti.



Obrázek 5.1 Definice sousedství pomocí transformace Lin-2-Opt.

V práci bylo použito celkem 11 různých způsobů definování relace sousedství. Jeden z nich – *Lin-2-Opt change* – je naznačen na obrázku 5.1. Popis zbylých relací a implementační detaily jednotlivých heuristických algoritmů najdete v [Palu03c].

Na základě provedených testů s různým nastavením parametrů heuristických algoritmů bylo jako nejlepší shledáno následující nastavení jednotlivých heuristických algoritmů [Palu03]:

- **Tabu search**

  Délka tabu listu: polovina počtu produktů ($K/2$)
  Počet iterací: smluvně stanoven
  Relace sousedství: Swap
  Aspirační kritérium: zapnuto
  Řešení jednoproduktové úlohy: modifikovaný Goldbergův algoritmus

- **Simulované žíhání**

  Výchozí teplota: 1700 – 1800
  Konečná teplota: 12 – 290
  Redukční funkce teploty: $f(t) = t / (1 + at)$, kde $a = 0.000008$;
  Relace sousedství: FullSwap
  Řešení jednoproduktové úlohy: modifikovaný Goldbergův algoritmus

- **Genetický algoritmus**

  Velikost populace: 2× počet produktů
  Selekce: ruleta
  Křížení: jednobodové
  Pravděpodobnost mutace: 0.01

Operátor mutace: 1stSwap
Generování nové populace: iterační
Vyřazený jedinec z populace: „jiný než nejlepší“
Počet generací: smluvně stanoven
Zamezení duplicitě chromozomů v populaci: zapnuto
Řešení jednoproduktové úlohy: modifikovaný Goldbergův algoritmus

Abychom mohli jednotlivé algoritmy srovnávat, byl u všech metod počet iterací stejný (přesné nastavení viz [Palu03c]). Jinak při běžném výpočtu bychom použili spíše vyšší počty iterací (dle našich časových možností).

5.4 Testy heuristických algoritmů

Chceme-li zjistit, který z navržených stochastických algoritmů je nejvýkonnější, musíme jednotlivé algoritmy podrobit sérii testů. Celkem bylo testováno 21 různých zadání. Testy probíhaly na třech různých typech sítí se 100 uzly: (1) *Mesh* s počtem hran 300; (2) *DoubleExponencialLine* (DEL) s počtem hran 600 a (3) *RandomMesh* (Random) s počtem hran 900. Každá síť byla ještě vygenerována pro 5 až 11 produktů. Sítě byly vygenerovány pomocí programu *GVS* (příloha [Palu03c]).

Výpočty byly provedeny na následující sestavě:

AMD Athlon XP 1900+
512 MB RAM
Windows XP

Aby bylo možné jednotlivé algoritmy porovnat, byly všechny výpočty ukončeny po prozkoumání stanoveného počtu prvků. Prozkoumáním prvku rozumíme výpočet hodnoty účelové funkce pro vygenerované řešení. Pro každou vstupní síť proběhl výpočet 30-krát. Kompletní záznam naměřených hodnot najdete v [Palu03c].

Důležitým hlediskem pro vyhodnocení testů je úspěšnost algoritmu při hledání optima a nalezená průměrná hodnota účelové funkce. Pro lepší názornost jsou průměrné dosažené hodnoty účelové funkce shrnuty v tabulce na obrázku 5.2. Pokud jako hlavní hodnotící kritérium budeme brát úspěšnost při nalezení optima, pak nejlepších výsledků dosahoval algoritmus SA. Když jako hlavní hledisko vezmeme kvalitu průměrného nalezeného řešení, vychází jako nejlepší opět algoritmus SA. Všimněme si ale, že rozdíly mezi jednotlivými algoritmy jsou opravdu minimální.

Počet produktů	Mesh			DEL			Random		
	TS	SA	GA	TS	SA	GA	TS	SA	GA
5	89836	89852	89868	15767	15767	15767	35071	35143	35143
6	95500	95495	95315	17191	17181	17159	44195	44301	44241
7	97639	97644	97627	17577	17578	16509	47367	47368	47356
8	96964	96965	96998	19747	19757	19735	53098	53083	53054
9	87700	87700	87700	18849	18818	18791	54554	54555	54547
10	90558	90625	90618	15868	15875	15816	40779	40847	40705
11	105111	104923	105215	17313	17324	17236	55399	55430	55357

Obrázek 5.2 Průměrné hodnoty nalezených řešení heuristickými algoritmy.

Neméně důležitým hlediskem je rychlost heuristického algoritmu. Vidíme, že i když se testy prováděly s malým počtem produktů a na sítích s malým počtem uzlů a hran, přesáhla i tak doba jednoho výpočtu mnohdy 20 minut. V praxi budeme řešit mnohem rozsáhlejší problémy (jak ve velikosti sítě, tak i v počtu produktů) a doba výpočtu se úměrně velikosti úlohy protáhne. Rozdíly v rychlosti jednotlivých algoritmů, které se jeví nyní jako nevýznamné, tak mohou nabýt na důležitosti. Přehled časových nároků jednotlivých

algoritmů je shrnut v tabulce na obrázku 5.3. Vidíme, že doba výpočtu v podstatě odpovídá „složitosti“ algoritmů. Nejrychleji počítá algoritmus TS, následuje SA a poslední skončil GA. Že GA především u sítí s menším počtem produktů zaostával jde především na vrub sledování duplicit jedinců v populaci. Pokud tuto kontrolu vypneme, zvýší se rychlost GA téměř na úroveň ostatních algoritmů. Důvodem zavedení této kontroly byla postupná degenerace populace (konvergence populace k jedinému řešení). Náchylnost k tomuto jevu klesá s velikostí populace, takže pro řešení úloh s více než 9 produkty nemá zavedení této kontroly již příliš opodstatnění.

Počet produktů	Mesh			DEL			Random		
	TS	SA	GA	TS	SA	GA	TS	SA	GA
5	4	4	26	6	6	21	7	6	19
6	11	13	44	14	15	36	23	17	41
7	80	86	89	111	210	243	150	168	350
8	197	209	279	246	275	486	154	406	659
9	310	338	599	442	478	781	631	581	689
10	384	455	453	620	640	801	781	820	1091
11	631	695	718	565	587	764	1087	1190	1397

Obrázek 5.3 Průměrné časy řešení heuristických algoritmů (v sekundách).

Z naměřených výsledků vidíme, že rozdíly mezi algoritmy nejsou významné. Máme-li se pokusit o vyhodnocení algoritmů, potom jako nejlepší algoritmus musíme označit SA, jako druhý TS a třetí GA. Z pohledu dalšího možného vývoje a zdokonalení heuristických algoritmů vidíme, že nejmenší prostor pro zlepšení máme v případě TS algoritmu. O něco větší možnosti se skrývají v případě algoritmu SA. Největší potenciál pro další zlepšení (modifikace) je skrytý v GA. Směrem na GA by měl být zaměřen i další výzkum.

6 Závěr

Účelem této práce bylo podrobné studium úlohy víceproduktových toků včetně speciálních případů s jedním produktem a navržení stochastického heuristického algoritmu pro její řešení. Úloha víceproduktových toků představuje nejkomplexnější typ úlohy mezi úlohami síťových toků. Navzdory častému výskytu této úlohy v praxi není úloha víceproduktových toků příliš předmětem zájmu výzkumníků. Přes velký význam je úloha víceproduktových toků v české literatuře dost opomíjena. Zpravidla se autoři věnují jen jednoproduktovým úlohám anebo pouze dvouproduktovému toku, který ale nelze zobecnit na víceproduktový. Ostatně i ze seznamu odkazů je zřejmé, že se bylo nutné opřít především o zahraniční literaturu.

Úloha víceproduktových toků je nejčastěji se vyskytující úlohou síťových toků v praxi. Protože se jedná o NP-těžkou úlohu, jsme při použití exaktních (přesných) algoritmů v dnešní době (při úrovni počítačů používaných dnes a v blízké budoucnosti ve firmách) schopni řešit pouze úlohy definované malými sítěmi a s nízkým počtem produktů. Pokud chceme optimalizovat problémy definované rozsáhlejšími sítěmi s více produkty, musíme se vzdát jistoty nalezení optima a použít některou ze stochastických heuristických metod anebo problémově specifickou heuristiku, pokud ale pro náš problém nějaká existuje. Tato práce byla zaměřena na řešení jak obecných víceproduktových úloh pomocí stochastických heuristických metod, tak i na řešení jednoproduktových úloh (úloha nejkratší cesty, úloha maximálního toku, úloha toku s minimálními náklady) pomocí (pseudo-) polynomiálních algoritmů.

V rámci této práce bylo vypracováno pět programů, které mimo jiné umožnily testování jednotlivých algoritmů (celkem 20) a mohou být taktéž použity přímo pro řešení konkrétních síťových úloh. Vlajkovou lodí mezi realizovanými programy je program nazvaný *VíceproduktovéToky*, kde byly uplatněny veškeré znalosti nabyté studiem víceproduktových a jednoproduktových úloh. Program umožňuje řešit úlohu celočíselných víceproduktových toků některým ze tří stochastických heuristických algoritmů. Při jejich návrhu se vycházelo z následujících heuristických strategií: tabu search, simulovaného žíhání a genetických algoritmů. Samotné heuristické strategie jsou definovány velmi obecně a jejich běh je závislý na množství

volitelných parametrů, které mají velký vliv na celkovou rychlost algoritmu a úspěšnost při nalezení optima. Po provedení série testů na různých typech a velikostech sítí s různým počtem produktů byla stanovena vhodná nastavení pro jednotlivé algoritmy. V další fázi se již přistoupilo k samotnému porovnání algoritmů a jako nejvýkonnější se nakonec ukázal algoritmus simulovaného žíhání. Z hlediska úspěšnosti nalezení optima byl nejlepší, rychlost výpočtu měl druhou nejvyšší. Experimentální výsledky potvrdily schopnost navržených stochastických heuristických algoritmů řešit úlohu víceproduktových toků.

Protože jednotlivé heuristické algoritmy rozkládají úlohu víceproduktových toků na posloupnost jednoproduktových úloh, pro které již existují výkonné (pseudo-) polynomiální algoritmy, měla volba algoritmu řešícího jednoproduktovou úlohu velký vliv na celkovou rychlost heuristiky. Každému typu jednoproduktové úlohy byla věnována jedna kapitola. Byl popsán matematický model dané úlohy, naznačeny možné příklady využití a především byly vytipovány vhodné algoritmy k implementaci. Definice algoritmů uváděné v literatuře se většinou zaměřují pouze na klíčové vlastnosti algoritmů a nebývají příliš podrobně popsány, čímž nechávají velký prostor pro samotnou implementaci algoritmu. Velmi potom záleží na zkušenostech a znalostech programátora, jak kvalitní (výkonná) bude implementace. V případě jednoho z algoritmů – Goldbergova algoritmu – se autorovi podařilo navrhnout modifikaci algoritmu, jejíž časová složitost sice zůstala stále $O(n^3)$, ale empirické chování algoritmu se výrazně zlepšilo, jak vyplynulo z provedených testů.

Abychom byli schopni určit nejvýkonnější algoritmus, byl pro každý typ jednoproduktové úlohy vytvořen speciální program zahrnující zvolené algoritmy a umožňující jejich testování na vybraném vzorku sítí. V případě úlohy nejkratší cesty byl jako nejlepší vyhodnocen Dijkstrův algoritmus pracující s *d*-haldou, a proto byl následně použit i v komplexnějších algoritmech. Vynikajících výsledků dosahoval taktéž Dialův algoritmus, ale pouze u sítí s malým ohodnocením hran (do 25000). Tato výkonová závislost na velikosti ohodnocení hrany ho stála pozici nejlepšího algoritmu. V případě úlohy maximálního toku přesvědčivě nejlepších výsledků dosahoval modifikovaný Goldbergův algoritmus. Poslední jednoproduktovou úlohou byl tok s minimálními náklady a v tomto případě v testech nejlepších výsledků dosahoval algoritmus postupných nejkratších cest. Součástí projektu byl i programem *GVS* (Generátor Víceproduktových Sítí), který dokáže generovat sítě s předepsaným počtem uzlů, hran a produktů. Ohodnocení hran je generováno náhodně ve zvoleném rozsahu. K dispozici máme několik různých topologií sítí. Veškeré testovací sítě byly vygenerovány pomocí tohoto programu.

I když původní motivací k testovaní algoritmů jednoproduktových úloh bylo nalezení rychlých exaktních algoritmů pro následné použití v heuristickém algoritmu, měly tyto testy význam i z pohledu samotných jednoproduktových úloh – nalezli jsme vhodné algoritmy pro řešení daného typu úloh. Problémy řešitelné pomocí jednoproduktových úloh se vyskytují v praxi stejně často jako u obecnější víceproduktové úlohy. Použití některých jednoproduktových úloh a datových struktur popsaných v této práci je ještě mnohem širší. Např. haldy (*d*-halda, Fibonacciho halda) můžeme využít i pro efektivní setřídění množiny čísel, pro zefektivnění algoritmů hledajících minimální kostru grafu ([Šeda01]); při redukci grafu ve Steinerově problému využijeme algoritmus pro řešení nejkratší cesty atd.

Práce nemohla vyčerpat všechny možnosti stochastických heuristických algoritmů. Velký potenciál na zlepšení je skryt především v genetickém algoritmu. Stále otevřená zůstává možnost vytvoření nějakého hybridního algoritmu.

Literatura

[Ahuj89] Ahuja, R. K., Orlin, J. B.: A fast and simple algorithm for the maximum flow problem. *Operation Research* 37, 748–759, 1989.

[Ahuj93] Ahuja, R. K., Magnanti T. L., Orlin B. J.: *Network flows: Theory, Algorithms, and Applications*. PRENTICE HALL, Englewood Cliffs, New Jersey, 1993.

[Ahuj97] Ahuja, R. K., Kodialam, M., Mishra, A. K., Orlin, J. K.: Computational investigations of maximum flows algorithms. *European Journal of Operational Research 97*, 509-542, 1997.

[Agga95] Aggarwal, A. K., Oblak, M., Vemuganti, R. R.: A heuristic solution procedure for multicommodity integer flows. *Computers Operations Research* 10, 1075–1087, 1995.

[Arro86] Arronson, J.E.: The multiperiod assignment problem: A multicommodity network flow model and specialized branch and bound algorithm. *European Operations Research* 23, 367–381, 1986.

[Bach66] Bacharach, M.: Matrix rounding problems. *Management Science* 9, 732–742, 1966.

[Bart80] Bartholdi, J. J., Orlin, J. B., Ratliff, H. D.: Cyclic scheduling via integer programs with circular ones. *Operations Research* 28, 1074–1085, 1980.

[Bell58] Bellman, R,: On a routing problem. *Quartely of Applied Mathematics* 16, 87–90, 1958.

[Bert88] Bertsekas, D. P., Tseng, P.: Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research* 36, 93–114, 1988.

[Busa61] Busaker, R. G., Gowen, P. J.: A procedure for determining minimal-cost network flow patterns. ORO Technical Report 15, Operational Research Office, John Hopkins University, Baltimore, MD, 1961.

[Chal82] Chalmet, L. G., Francis, R. L., Saunders, P. B.: Network models for building evacuation. *Management Science* 28, 86–105, 1982.

[Cook98] Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., Schrijver, A.: *Combinatorial Optimization.* John Wiley and Sons, New York, 1998. ISBN 0-471-55894-X

[Core91] Coremen, T. H., Leiserson C. E., Rivest, R. L.: *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, London, England, 1991.

[Dant61] Dantzig, G. B., Wolfe, P.: The decomposition method for linear programming. *Econometrica* 29, 767–778, 1961.

[Dial69] Dial, R.: Algorithm 360: Shortest path forest with topological ordering. *Communications of ACM* 12, 632–633, 1969.

[Dijk59] Dijkstra, E., A note on two problems in connexion with graphs. *Numerische Matematik* 1, 269–271, 1959.

[Dini99] Dinitz, Y., Garg, N., Goemans, M.: On the Single – Source Unsplittable Flow Problem. In *Combinatorica* 19, 1–25, 1999.

[Dvor02] Dvořák, T.: *Metody návrhu efektivních algoritmů*. KSVI MFF UK, Praha, 2002.

[Edmo72] Edmonds, J., Karp, R. M.: Theoretical improvements in alghoritmic efficiency for network flow problems. Journal of ACM 19, 248–264, 1972.

[Fede86] Federgruen, A., Groenevelt, H.: Preemptive scheduling of uniform machines by ordinary network flow techniques. *Management Science* 32, 341–349, 1986.

[Fish81] Fisher, M. L.: The Lagrangian relaxation methods for solving integer programming problems. *Management Science* 27, 1–18, 1981.

[Ford56] Ford, L. R., Fulkerson, D. R.: Maximal flow through a network. *Canadian Journal of Mathematics* 8, 399–404, 1956.

[Ford62] Ford, L. R., Fulkerson, D. R.: *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[Fred84] Fredman, M. L., Tarjan, R. E.: Fibonacci heaps and their uses in improved network optimization algorithms. In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338–346, 1984, Full paper in *Journal of ACM* 34, 596–615, 1987.

[Fulk66] Fulkerson, D. R.: Flow Networks and combinatorial operations research. *American Mathematical Monthly* 73, 115–138, 1966.

[Gabo85] Gabow, H. L.: Scaling algorithms for network problems. *Journal of Computer and System Sciences* 31, 148–168, 1985.

[Geof74] Geoffrion, A. M., Graves, G. W.: Multicommodity distribution system design by Benders decomposition. *Management Science* 20, 822–844, 1974.

[Glov77] Glover, F., Klingman D.: Network applications in industry and government. *AIIE Transactions* 9, 363–376, 1977.

[Glov97] Glover, F., Laguna, M.: *Tabu Search*. Kluwer Academic Publisher, Boston, 1977.

[Goem94] Goemans, M.: *Networks Flow*. Massachusetts Institute of Technology, Laboratory for Computer Science, Lecture Notes, 1994.

[Gold85] Goldberg, A. V.: A new max-flow algorithm. Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, MIT, Cambridge, MA, 1985.

[Gold89] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Gold93] Goldberg, A. V., Tarjan, R. E: A New Approach to the Maximum-Flow Problem. In *Journal of the Association for Computing Machinery* 35, 921–940, 1993.

[Gupt85] Gupta, S. K.: *Linear Programming and Network Models*. Affiliated East-West Press, New Delhi, India.

[Hass84] Hassin, R.: On multicommodity flows in planar graphs. *Networks* 14, 225–235, 1984.

[Hoch98] Hochbaum, D.: The Pseudoflow Algorithm and the Pseudoflow-Based Simplex for the Maximum Flow Problem. In *Proceedings of the 6th International Conference on Integer Programming and Combinatorial Optimization (IPCO '98)*, Houston (Texas), 325–337, 1998.

[Holl75] Holland, J., H.: *Adaptation in Natural and Artificial System*. Ann Arbor, MI: The University of Michigan Press, 1975.

[Imai86] Imai, H., Iri, M.: Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics and Image Processing* 36, 31–41, 1986.

[Jane96] Janeček, J.: *Distribuované systémy*. ČVUT Praha, 1996.

[Jewe57] Jewell, W.S.: Warehousing and distribution of a seasonal product. Naval Research Logistics Quarterly 4, 29–34, 1957.

[John77] Johnson, D. B.: Efficient special-purpose priority queues. In *Proceedings of the 15th Annual Allerton Conference on Communications, Control, and Computing* , 1–7, 1977.

[Kenn80] Kennington, J. L., Helgason, R. V.: *Algorithms for Network Programming*. Wiley-Interscience, New York, 1980.

[Khul94] Khuller, S.: Design and Analysis of Algorithms. Lecture Notes, University of Maryland, Department of Computer Science, 1994.

[Klap01] Klapka, J., Dvořák, J., Popela, P.: *Metody operačního výzkumu*. VUT Brno, 2001.

[Klei67] Klein, M.: A primal method for minimal cost flows with application to the assignment and transportation problem. *Management Science* 24, 205–220, 1967.

[Kort88] Korte, B.: Applications of combinatorial optimization. Technical Report 88541-OR, Institute für Okonometrie und Operations Research, Bonn, Germany, 1988.

[Lomo83] Lomonosov, M.V.: On the planar integer two-flow problem. *Combinatorica* 3, 207–218, 1983.

[Mich96] Michalewicz, Z.: *Genetic Algorithms + Data Structure = Evolution Programs*. Springer Verlag, Berlin, 1996.

[Orli93] Orlin, J. B.: A faster strongly polynomial minimum cost flow algorithm. *Operations Research* 41, 338–350, 1993.

[Otte89] Otten, R. H. J. M., van Giniken, L. P. P. P.: *The Annealing Algorithm.* Kluwer, Boston, 1989.

[Ples83] Plesník, J.: *Grafové algoritmy.* Alfa, Bratislava, 1983.

[Reev93] Reeves, C. R.: *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publication, Oxford, 1993.

[Riou92] Rioux, P., Smith, B. T., Thulasiraman, K.: Minimum Cost Sizing of Rearrangeable Networks with Multiperiod Demands. Submitted to *INFOR,* 1992.

[Röck80] Röck, H.: Scaling Techniques for Minimal Cost Network Flows. In *Discrete Structures and Algorithms,* 181–191, 1980.

[Slea83] Sleator, D. D., Tarjan R. E.: A data structure for dynamic trees. *Journal of Coumputer and System Sciences* 24, 362–391, 1983.

[Slum82] Slump, C. H., Gerbrands, J. J.: A network flow approach to reconstruction of the left ventricle from two projections. *Computer Graphics and Image Processing* 18, 18–36, 1982.

[Ston77] Stone, H. S.: Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering* 3, 85–93, 1977.

[Šeda98] Šeda, M.: *Využití moderních heuristických metod v rozvrhování.* Disertační práce, Vysoké učení technické v Brně, Ústav automatizace a informatiky, 1998.

[Šeda01] Šeda, M.: Steinerovy problémy a přibližné metody jejich řešení. Habilitační práce, Vysoké učení technické v Brně, fakulta strojního inženýrství, 153 stran, 2001.

[Teix02] Teixeira, S., Pacheco, X.: *Mistrovství v Delphi 6*. Computer Press, Praha, 2002. ISBN 80-7226-627-6

[Whit69] White, L. S.: Shortest route models for the allocation of inspection effort on a production line. *Management Science* 15, 249–259, 1969.

[Zawa87] Zawack, D. J., Thompson, G. L.: A dynamic space-time network flow model for city traffic congestion. *Transportation Science* 21, 153–162, 1987.

Publikace autora

[Palu00a] Palubják, P.: Toky v sítích. In *Sborník příspěvků doktorandů pedagogicko-vědecké konference u příležitosti 100. výročí založení fakulty strojního inženýrství*. VUT FSI, Brno, str. 237–242, 2000.

[Palu00b] Palubják, P.: *Toky v sítích.* Diplomová práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, 64 stran, 2000.

[Palu01a] Palubják, P.: Moderní algoritmy pro hledání maximálního toku v síti. In *Proceedings of the XXVIth Seminar ASR 2001 Instruments & Control*. Ostrava, str. 55/1–55/10, 2001. ISBN 80-7078-890-9.

[Palu01b] Palubják, P.: Goldberg's Algorithms. In *Proceedings of the 13th International Conference PROCESS CONTROL '01.* Štrbské pleso (Slovakia), 2001, 4 pp. on CD-ROM `\Papers\Session4\p062.pdf`. ISBN 80-227-1542-5; (Summaries Volume, pp. 122).

[Palu03a] Palubják, P.: Minimum Cost Flows. In *Proceedings of the First International Conference on Soft Computing Applied in Computer and Economic Environments ICSC 2003.* Evropský polytechnický institut, Kunovice, pp. 103–108, 2003. ISBN 80-7314-017-9.

[Palu03b] Palubják, P.: Zjemňující algoritmy. In *Proceedings of the XXVIIIth Seminar ASR 2003 Instruments & Control.* Ostrava, 10 stran, 2003.

[Palu03c] Palubják, P.: *Víceproduktové toky v sítích.* Pojednání o disertační práci, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 31 stran, 2003.

Summary

The aim of this paper was to study the multicommodity flow problem in detail and to design a stochastic heuristic algorithm. This study also covered special cases with one product. The multicommodity flow problem is the most complex type of the problem among network flow problems. In spite of a frequent occurrence in practice, multicommodity flow problem is not the object of research's attention. Despite great importance the multicommodity flow problem is not studied in the Czech literature. Authors only deal with single commodity flows or two commodities flows that cannot be generalized to multicommodity. It is evident from the list of the references, it was necessary to use the foreign literature.

The multicommodity flow problem is most often occurring problem from network flows in practice. Because this problem is NP-hard problem, we are able to solve by exact algorithms in present (by the levels of the computers used in the firms today and in the near future) only the problems that are defined by small networks and with low number of products. If we want to optimize problems defined by large networks and many products, we have to give up guarantees of optimality and to use a stochastic heuristic method or deterministic heuristic, if any has been developed for our problem. This work was aimed at solving multicommodity flow problems by stochastic heuristic methods and at solving single commodity problems (shortest path problem, maximum flow problem, minimum cost flow problem) by (pseudo-) polynomial algorithms.

Within this work has been designed several computer programs that enabled to test algorithms and may be used directly for solving concrete network problems. Leading product among realized programs was program named *ViceproduktovéToky*, where all knowledge obtained by study multicommodity and single commodity problems was used. Program allows solving integer multicommodity flow problems by some of the three stochastic heuristic algorithms. At their design it was gone out from following heuristic strategies: taboo search, simulated annealing and genetic algorithms. Single heuristic strategies are defined very generally and their running time is depending on many optional parameters that have a great influence on the total speed of the algorithm and successfulness at the finding optimum. In the first phase of the tests suitable settings for individual algorithms was determined. In the second phase heuristic algorithms have already been compared and as the best algorithm has resulted simulated annealing algorithm.

Because individual heuristic algorithms decompose multicommodity flows problem on the sequence single commodity problems (for which efficient polynomial algorithms have already been developed), choice of the algorithm for solving single commodity problem had the great influence on the total speed of the heuristics. One chapter has been devoted to each type of single commodity problem. Mathematical model of the given problem was described, possible examples of the applications were showed and above all suitable algorithms to the implementation were determined. In the literature introduced definitions of the algorithms are aimed only on the key properties these algorithms and therefore there is very wide scope for their implementation. It depends very much on the experience and knowledge of the programmer, how efficient the implementation will be. For Goldberg's algorithm author suggested a modification that markedly improves its empirical behavior. Correctness of this modification was confirmed by the computer results.

So that we were able to determine the most efficient algorithm, a special program for each type of single commodity problem was created. This program allowed testing the algorithms for the selected sample networks. For shortest paths problem Dijkstra's algorithm working with *d*-heap was measured as the best. Subsequently, this algorithm was used in more complex algorithms. Excellent results were reached by Dial's algorithm, but only for networks with the low weights of the arcs (up to the 25000). For maximum flow problem the best results were reached by the modified Goldberg's algorithm. Last single commodity problem solved in this dissertation were minimum cost flows. For this problem the best results were reached by the successive shortest paths algorithm. Part of the project was the program *GVS* that can generate networks with determined numbers of nodes, arcs and products. Numerical values associated with arcs are generated randomly in a selected range. At disposal we have several different topologies of networks. This program generated all the test networks.

Although the primal motivation was finding the fast exact algorithms for latter using in the heuristic algorithm, these tests had also another function – to find suitable algorithms for solving given type problem. Problems solvable by single commodity algorithms occur frequently in practice. This work could not exhaust all possibilities of the stochastic heuristic algorithms. Great potential of the improvement is hidden in the genetic algorithm. Still opened possibility stays a creation of hybrid algorithm.

Curriculum vitae

Ing. Petr Palubják
Malý Skalník 803, Vsetín 755 01, Česká republika
Tel: 604 638 892
e-mail: palubjak@kn.vutbr.cz

Osobní data

Datum narození:	18.6. 1976
Národnost:	česká
Stav:	svobodný

Vzdělání

2000 – 2003 **Postgraduální doktorské studium na Vysokém učení technickém v Brně**
Fakulta strojního inženýrství
Obor: Technická kybernetika
Téma disertační práce: Víceproduktové toky v sítích

1995 – 2000 **Vysoké učení technické v Brně**
Fakulta strojního inženýrství
Obor: Inženýrská informatika
Téma diplomové práce: Toky v sítích

1991 – 1995 **Střední průmyslová škola strojnická ve Vsetíně**

Pedagogická činnost

V rámci doktorského studia výuka předmětů Informatika I, Zpracování informací, Počítačové sítě a Teorie grafů.

Jazykové znalosti

Němčina:	výborná (slovem i písmem)
Angličtina:	dobrá

Dovednosti

Programování v Delphi, C++, PHP, CSS, HTML, SQL.
Databáze, Linux.
AutoCAD, Corel Draw, MS Office.

Zájmy

Počítače, programování.
Triatlon, cyklistika, stolní tenis.

Další informace

Řidičský průkaz skupiny A, B.