VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

CELOČÍSELNÁ OPTIMALIZACE PRO ŘEŠENÍ DOPRAVNÍCH ÚLOH
INTEGER OPTIMIZATION FOR TRANSPORTATION PROBLEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matouš Cabalka**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. Pavel Popela, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Student:	**Matouš Cabalka**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**RNDr. Pavel Popela, Ph.D.**
Akademický rok:	2015/16

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Celočíselná optimalizace pro řešení dopravních úloh

Stručná charakteristika problematiky úkolu:

Student se seznámí s problematikou modelů celočíselné optimalizace pro řešení dopravních úloh, s důrazem na úlohy typu obchodního cestujícího. Student využije osvojené poznatky diskrétní matematiky, lineární algebry a teorie grafů. Prostuduje vlastnosti a modely vybraných úloh, zaměří se na vybrané algoritmy celočíselné optimalizace a jejich softwarové implementace. Pro testovací výpočty budou také využita data z reálných problémů.

Cíle bakalářské práce:

Předpokládá se studium, modifikace a implementace existujících modelů a algoritmů a jejich testovací aplikace v návaznosti na svozové problémy řešené v rámci projektu NETME+ a ve spolupráci ústavu matematiky FSI VUT v Brně se specialisty v oblasti logistiky z norské Molde University College. Důraz bude kladen na přístupy celočíselného programování a jejich matematický popis. Pro výpočty bude využit vhodný modelovací optimalizační programový systém.

Seznam literatury:

Klapka, J. a kol. (2000): Metody operačního výzkumu, Brno.

Christofides, N. (1975): Graph Theory - an Algorithmic Approach. Academic Press.

Wolsey, L. A. (1998): Integer programming. John Wiley and Sons.

Ghiani, G., Laporte, G., Musmanno, R. (2004): Introduction to Logistics Systems Planning and Control. John Wiley and Sons, New York.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Práce se zabývá optimalizačními modely v dopravních úlohách s důrazem na úlohu obchodního cestujícího. Po stručném úvodu do historie následuje část popisující základy lineárního a celočíselného programování. Následuje uvedení formulace úlohy obchodního cestujícího. Dále je zahrnuta část věnovaná přípravě dat, na kterou přímo navazuje výpočtová část. Dosažené výsledky jsou opatřeny komentářem a závěry.

KLÍČOVÁ SLOVA

Lineární programování, celočíselné programování, přiřazovací problém, úloha obchodního cestujícího, GAMS

ABSTRACT

The thesis deals with optimization models in transportation problems with emphasis on traveling salesman problem. Brief introduction to the history is followed by theoretical part describing linear programming, integer programming and formulation of traveling salesman problem. Description of data preprocessing is included. Finally computational results are discussed and evaluated.

KEYWORDS

Linear programming, integer programming, assignment problem, traveling salesman problem, GAMS

CABALKA, Matouš. *Celočíselná optimalizace pro řešení dopravních úloh*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky, 2016. 46 s. Vedoucí práce RNDr. Pavel Popela, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Celočíselná optimalizace pro řešení dopravních úloh“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu RNDr. Pavlu Popelovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Velký dík patří taktéž Ing. Jakubu Kůdelovi za čas strávený konzultacemi a dobře mířené rady. Zvláštní poděkování bych na závěr rád věnoval rodině a přátelům, kteří mne během studia vytrvale podporovali.

Brno ....

podpis autora

OBSAH

ÚVOD

Ve své práci, Celočíselná optimalizace pro řešení dopravních úloh, se zabývám problémem obchodního cestujícího. Cílem bakalářské práce je detailní popis jeho modelu, otestování dat a patřičné okomentování výsledků. Na závěr jsou uvedeny možné modifikace v práci použité formulace.

První kapitola zahrnuje stručný úvod do historie problému. Následující kapitola je věnována lineárnímu programování a simplexové metodě, na jejímž principu pracuje řešič použitého softwaru GAMS. Navazuje kapitola shrnující důležitá fakta celočíselného programování.

Samotné úloze obchodního cestujícího předchází formulace přiřazovacího problému, na který úloha obchodního cestujícího navazuje. V další části je uvedena a vysvětlována vybraná formulace úlohy obchodního cestujícího.

Pátá kapitola vysvětluje důvod výběru softwaru GAMS, přípravu dat v MS Excel a jejich zpracování. Dále je uveden testovací příklad a výsledky dosažené během výpočtů, které jsou komentovány i s pomocí grafů. Na závěr jsou uvedeny další vybrané formulace úlohy.

1 HISTORIE

Historie dopravních úloh se začíná psát až v 19. století při rozvoji průmyslu ve větších městech a rozvoji sériové výroby, jak uvádí [5]. Logicky však úvahy o této problematice sahají až do počátků lidstva, kdy různí kupci cestovali mezi osadami a jinými centry tehdejšího dění.

Zprvu bych rád uvedl několik historických poznámek dle [1]. Jako první příklad lze uvést tzv. „Circuit riders“. Jedná se o výraz z Anglie 15. století popisující soudce cestující po větších městech v Anglii. Vzhledem k dané době byl na daném území poměrně malý počet osob, který mohl vykonávat soudní praxi. Z tohoto důvodu měli tito lidé jasně naplánované pravidelné trasy a zastávky. Protože náklady na cestu soudce byly poměrně vysoké, hledala se vždy nejkratší trasa mezi určitým počtem měst. Pojem „Circuit riders“ se později objevuje i ve Spojených státech amerických, kde byl jedním z nejslavnějších vykonavatelů soudní praxe i mladý Abraham Lincoln vykonávající praxi na čtrnácti místech ve státě Illinois.

Jako další lze například uvést „Cestu šachového jezdce“. Cílem úlohy, kterou řešil Leonhard Euler, je najít cestu šachového jezdce přes všechna pole šachovnice tak, aby se na každé pole postavil právě jednou, a přitom se vrátil na původní startovací pole. Dále za zmínku stojí hra Ikosián od irského matematika Sira Williama Rowana Hamiltona. Ikosián je dvanáctistěn s dvaceti vrcholy. I zde je cílem navštívit každý z vrcholů jen jednou a přitom se vrátit do výchozího bodu. Cesta, která tomuto cyklu vyhovuje, se nazývá Hamiltonův cyklus.

Úloha začala poutat rozsáhlou pozornost až ve třicátých letech 20. století. Prvním významným počinem v této oblasti byl problém 48 států. Jednalo se o problém formulovaný Whitney Hasslerem a Merrillem Floodem, jehož řešení mělo obsahovat cestu přes 48 vybraných měst ve Spojených státech se startem a cílem v tomtéž městě. Jako zajímavost lze zmínit i obrazy pánů Bosche a Kaplana, kteří namalovali slavný obraz Mony Lisy nebo římského Kolosea pouze s pomocí optimální cesty skrz značné množství vhodně rozmístěných bodů, viz [1].

Úloha je aplikována vědci napříč vědním spektrem. Mimo nejzřejmějšího užití v logistice nachází taktéž zásadní využití v genetice, telekomunikacích, neurovědách a mnoha dalších oborech.

Úlohu lze řešit mnoha různými způsoby, jako příklad lze uvést metodu sečných rovin, metodu větví a mezí nebo velmi populární heuristické metody, jimiž se zabývala autorka práce [5]. Jako zajímavé se mi jevilo uchopení problému formou celočíselné lineární optimalizace, pro níž jsem se rozhodl a kterou ve své práci rozebírám.

2 LINEÁRNÍ PROGRAMOVÁNÍ

Lineární programování se využívá k hledání vázaných extrémů lineárních funkcí více proměnných, jejichž omezující podmínky mají tvar lineárních rovnic a nerovnic. Původní myšlenky formulované J. B. J. Fourierem (1768-1830) v polovině 20. století rozvinul v moderní efektivní metody G. B. Dantzig, který dále využil myšlenek předchozích autorů a sestavil algoritmus, který je dnes nazýván simplexová metoda. Cílem této kapitoly je popis a vysvětlení simplexové metody pro řešení úloh lineárního programování.

2.1 Obecná formulace úlohy lineárního programování

Vzhledem k tomu, že v mé práci je kladen důraz na maximalizaci zisku, respektive minimalizaci nákladů, dopravní úlohy, dovolím si formulovat úlohu lineárního programování jako maximalizační. Při formulaci úlohy vycházím z [2].

Nechť existují vektory $\mathbf{x} = (x_j)$, $\mathbf{c} = (c_j)$, $\mathbf{b} = (b_i)$ a matice $\mathbf{A} = (a_{ij})$ typu $\mathsf{m} \times \mathsf{n}$, $i = 1, ..., m$, $j = 1, ..., n$. Nechť dále a_{ij}, b_i, c_j jsou daná reálná čísla a x_j jsou reálné proměnné. Poté lze úlohu

$$\begin{aligned} \max \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x}, \\ \text{za podmínky} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

nazvat maximalizační úlohou lineárního programování. Množina bodů splňující soustavu lineárních nerovnic se nazývá množina přípustných řešení. Koeficienty a_{ij} obvykle nazýváme strukturálními koeficienty, koeficienty b_i kapacitními limity a koeficienty c_j cenovými koeficienty.

Alternativou k maticovému zápisu je tzv. sumačně indexový zápis vhodný pro GAMS, který lze zapsat takto:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{za podmínek} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m, \\ & x_j \geq 0. \end{aligned}$$

Existují úlohy, které mohou vyjít celočíselně, přestože požadavky na celočíselnost úlohy nebyly kladeny, viz 2.3. Zároveň je důležité zmínit, že existují speciální úlohy, při jejichž výpočtu je celočíselnost řešení zaručena automaticky. Takový typ

úlohy představuje tzv. přiřazovací problém, jehož prvky jsou užity při řešení úloh obchodního cestujícího.

2.2 Simplexová metoda

Jak uvádí [3], simplexovou metodu odvodil s využitím myšlenek Gaussovy eliminační metody pro řešení soustav lineárních rovnic George Dantzig. Cílem simplexové metody je najít optimální řešení příslušné úlohy. Množina všech optimálních řešení představuje polyedrickou množinu popsatelnou krajními body a směry. Nejsnazší představu umožňuje problém se dvěma bázickými proměnnými, neboť se řešení nachází v rovině. Předpokládejme, že známe krajní bod $\mathbf{x_0}$ množiny přípustných řešení M. Z $\mathbf{x_0}$ vychází konečné množství hran, z nichž každá může obsahovat další krajní bod, nebo být neomezená. Pokud některá neomezená hrana obsahuje bod, kde je hodnota účelové funkce větší než $\mathbf{c}^{\mathrm{T}}\mathbf{x_0}$, optimální řešení úlohy neexistuje a výpočet končí.

V opačném případě se hledá sousední bod $\mathbf{x}$, v němž je hodnota účelové funkce $\mathbf{c}^{\mathrm{T}}\mathbf{x} > \mathbf{c}^{\mathrm{T}}\mathbf{x_0}$. Pokud takový sousední bod neexistuje, výpočet končí a $\mathbf{x_0}$ je optimálním řešením. Pokud takový bod existuje, pokračuje se v hledání, dokud se nenarazí na optimum. Podrobnosti algoritmu lze nalézt v [2].

V případě úlohy, která má optimální řešení, se výsledek získá konečným počtem kroků, nebo počtem kroků nutně vedoucím ke zjištění, že optimální řešení neexistuje. Konečnost úlohy zajišťuje konečné množství krajních bodů. Maximalizační úlohu lineárního programování lze převést na maximalizační úlohu lineárního programování ve standardním tvaru, viz. následující podkapitola.

2.3 Příklad

Tento příklad je smyšlený a slouží především k vysvětlení principu algoritmu simplexové metody, zároveň však bude využito i možnosti předvedení sestavení modelu, metod grafického řešení a řešení pomocí simplexové tabulky.

Nechť existuje malá firma Cabalka s.r.o. vyrábějící hračky pro děti v předškolním věku. Vyrábí postavičky ze dvou dílů $D1$ a $D2$. Data úlohy jsou uvedena v tabulce.

	potřebné díly		denní zásoba dílů
	panáčci	panenky	(tis. ks)
D1	2	1	60
D2	1	2	60
zisk z prodeje (80 Kč)	2	3	

Na základě průzkumu trhu se prokázalo, že o kvalitní hračky je velký zájem, a tedy co se vyrobí, se také ihned prodá.

2.3.1 Sestavení modelu

Při sestavování matematického modelu je nutné dle [3] jasně definovat tyto tři body:

1. Hledané proměnné
2. Účelová funkce, jejíž optimum je předmětem výpočtu
3. Omezující podmínky

Při tvorbě modelu jsem se inspiroval [3]. Cílem mistra výroby ve společnosti je určit, kolik postaviček kterého druhu je třeba vyrobit, aby zisk byl maximální. Definice proměnných se provede následujícím způsobem:

x_1 = množství vyrobených panenek,

x_2 = množství vyrobených panáčků.

Jestliže z reprezentuje zisk, pak se vzhledem k cenám daných výrobků na trhu definuje účelová funkce jako:

$$z = 2x_1 + 3x_2.$$

Vzhledem k poptávce stačí pro splnění posledního bodu definovat pouze dvě výrobní omezení týkající se omezených zdrojů zásob.

$$\begin{aligned} 2x_1 + x_2 &\leq 60, && \textit{(Dovezené díly 1. druhu)} \\ x_1 + 2x_2 &\leq 60, && \textit{(Dovezené díly 2. druhu)} \\ x_1, x_2 &\geq 0. && \textit{(Výrobky jsou vyráběny)} \end{aligned}$$

Celou úlohu lze tedy zapsat jako:

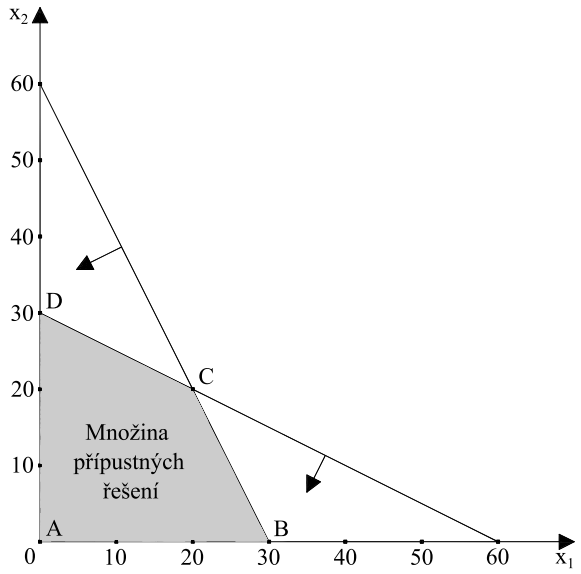
$$\begin{aligned} \max z, \\ z = 2x_1 + 3x_2, \end{aligned}$$

za podmínek:

$$\begin{aligned} 2x_1 + x_2 &\leq 60, \\ x_1 + 2x_2 &\leq 60, \\ x_1, x_2 &\geq 0. \end{aligned}$$

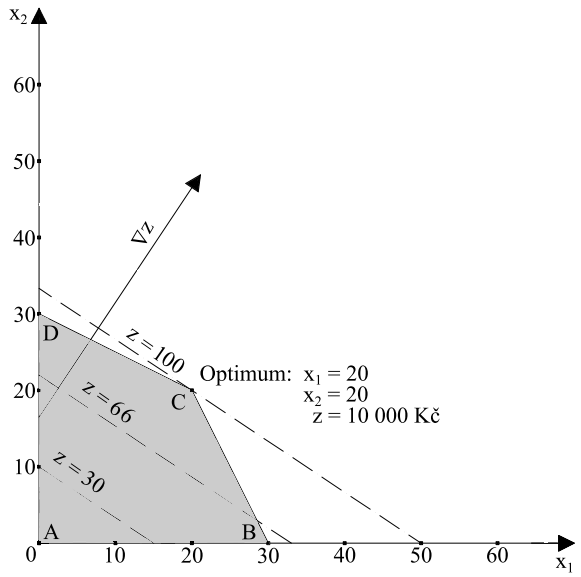
2.3.2 Grafické řešení

Krok 1: Určit množinu přípustných řešení.
Pomocí nerovnic vyjadřujících omezení úlohy se určí množina všech $\mathbf{x} = (x_1, x_2)$ splňujících tyto podmínky. Vzniklá množina se nazývá množina přípustných řešení.



Obr. 2.1: Vymezení množiny přípustných řešení

Krok 2: Určit optimální řešení.
Nezpochybnitelnou výhodou úlohy lineárního programování je fakt, že směr největšího růstu reprezentovaný gradientem účelové funkce je právě díky linearitě účelové funkce ve všech bodech $\mathbf{x}$ stejný. Směr největšího růstu lze vyjádřit vektorem $\nabla z = (2, 3)$. Je zřejmé, že na vrstevnici je hodnota účelové funkce konstantní. Nalezením optimálního řešení se tedy rozumí nalezení bodu $\mathbf{x}$, jenž leží na vrstevnici a ze všech bodů množiny M mu náleží maximální hodnota.



Obr. 2.2: Nalezení optimálního řešení

2.3.3 Řešení pomocí simplexové metody

Jestliže je požadováno řešení úlohy pomocí simplexové metody, je nejprve nutné převést úlohu na standardní tvar, v němž:

1. Všechna omezení jsou ve tvaru rovnic s nezápornou pravou stranou vyjma těch, které zaručují nezápornost proměnných.
2. Všechny proměnné jsou nezáporné.
3. Úloha je maximalizačního typu.

Cesta k získání rovnic vede přes doplnění nerovnic o nezáporné proměnné. Počet nových přidaných proměnných je zpravidla roven původnímu počtu nerovnic. Místo standardní, běžně vyučované minimalizace lze uvažovat také maximalizaci, která bude předvedena na příkladu uvedeném v této kapitole. Původní úloha je:

$$\max z = 2x_1 + 3x_2$$

za podmínek

$$\begin{aligned} 2x_1 + x_2 &\leq 60, \\ x_1 + 2x_2 &\leq 60, \\ x_1, x_2 &\geq 0. \end{aligned}$$

Standardní tvar původní úlohy je:

$$\max z = 2x_1 + 3x_2$$

za podmínek

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 60, \\ x_1 + 2x_2 + x_4 &= 60, \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

Ač se tento případ v úloze nevyskytuje, může se stát, že se vyskytne omezení typu:

$$x \geq 2.$$

S tím si lze ale snadno poradit jednoduchým odečtením přidané nezáporné proměnné:

$$\begin{aligned} x - x_n &= 2 \\ x, x_n &\geq 0. \end{aligned}$$

Objeví-li se v modelu i proměnná, která není ostatními proměnnými nijak omezena, řešení se skrývá v rozkladu na rozdíl její kladné a záporné části:

$$x_i = {x_i}^+ - {x_i}^-,$$
$${x_i}^+, {x_i}^- \geq 0.$$

Standardní tvar je definován soustavou m lineárních rovnic o n neznámých ($m < n$). Proměnné n lze rozdělit na dva typy:

1. m proměnných, jejichž hodnoty jsou řešením soustavy rovnic,
2. pomocných $n - m$ proměnných, jimž bude na závěr výpočtu přiřazena nulová hodnota.

Jestliže existuje pouze jedno řešení soustavy lineárních rovnic, nazveme m vybraných proměnných popisující řešení bázické proměnné, naopak $n - m$ proměnných nazveme nebázické proměnné. Jestliže jsou všechny proměnné dle formulace úlohy nezáporné, hovoří se o přípustném řešení, v opačném případě o nepřípustném. Standardní tvar úlohy bude v tabulce zapsán dle [3] následujícím způsobem:

	z	x_1	x_2	x_3	x_4	řešení
z	1	-2	-3	0	0	0
x_3	0	2	1	1	0	60
x_4	0	1	2	0	1	60

Vzhledem k tomu, že cílem je najít bázické řešení, hledá se jednotková matice uvnitř této tabulky. Lze snadno vidět, že toto kritérium splňují sloupce proměnných x_3 a x_4. Jestliže se proměnným x_1, x_2 přiřadí hodnota 0, řešením bude $x_3 = 60$, $x_4 = 60$. Hodnota účelové funkce je nyní 0.

Optimalitu řešení lze v každém kroku otestovat dosazením do účelové funkce $z = 2x_1 + 3x_2$, jejíž hodnotu lze během výpočtu pozorovat v pravém horním rohu simplexové tabulky. Jestliže se hodnota účelové funkce oproti předchozímu kroku zvýšila, dostali jsme optimálnější řešení. Aktuální řešení se nachází v množině přípustných řešení, optimálním není, protože řešení určují pouze pomocné proměnné x_3, x_4. Je tedy třeba hledat jinou bázi.

Efektivní postup při výpočtu vede přes vybrání a následnou eliminaci podle nejzápornějšího koeficientu proměnné v účelové funkci, která tak zajistí největší růst z. V tomto případě je jednoznačnou volbou x_2:

bázové proměnné	x_2	řešení	poměr	
x_3	1	60	$\frac{60}{1} = 60$	
x_4	2	60	$\frac{60}{2} = 30$	(minimum)

Dalším krokem je vybrání proměnné s nejmenším poměrem koeficientů ve sloupci $\frac{\text{řešení}}{x_2}$, protože jen tak dojde k maximálnímu růstu účelové funkce. V tomto případě se tedy jedná o proměnnou x_4. Volba je zvýrazněna v tabulce, daný řádek a sloupec budeme nazývat pivotní sloupec a pivotní řádek.

	z	x_1	x_2	x_3	x_4	řešení
z	1	-2	-3	0	0	0
x_3	0	2	1	1	0	60
x_4	0	1	2	0	1	60

Nové bázické řešení bude získáno pomocí Gauss-Jordanovy eliminace následujícím způsobem:

- Jelikož je cílem obdržet nové bázické řešení, je důležité na dané pozici v pivotním řádku a sloupci obdržet číslo 1. Toho lze docílit dělením daného řádku pivotním elementem.
- Aby byla eliminace kompletní, je nutné v tomtéž sloupci ve všech ostatních řádcích eliminovat prvky, tj. aby měly hodnotu 0, a tedy upravit celé řádky eliminací.

Přepočtená tabulka po uskutečnění výše zmíněných kroků má následující podobu:

	z	x_1	x_2	x_3	x_4	řešení
z	1	$\frac{-1}{2}$	0	0	$\frac{3}{2}$	90
x_3	0	$\frac{3}{2}$	0	1	$\frac{-1}{2}$	30
x_2	0	$\frac{1}{2}$	1	0	$\frac{3}{2}$	30

Opět aplikujeme postup zmiňovaný výše. Nebázickým proměnným se přiřadí automaticky hodnota 0. Z tabulky se tak obdrží bázické řešení $x_2 = 30$, $x_3 = 30$. Přestože hodnota účelové funkce z vzrostla, je třeba provést kontrolu, zda je řešení optimální. Transformovaná funkce z má následující tvar:

$$z = \tfrac{1}{2}x_1 + \tfrac{-3}{2}x_4 + 90$$

Jelikož se nachází kladné číslo u koeficientu x_1, ani toto řešení není zcela optimálním, a proto je třeba provést další krok Gauss-Jordanovy eliminace. Dle předchozího postupu se opět vybere nejzápornější proměnná v řádku účelové funkce, kterou je nyní x_1.

bázové proměnné	x_1	řešení	poměr	
x_3	$\frac{3}{2}$	30	$\frac{30}{\frac{3}{2}} = 20$	(minimum)
x_2	$\frac{1}{2}$	30	$\frac{30}{\frac{1}{2}} = 60$	

Další eliminovanou proměnnou je podle algoritmu simplexové metody x_3. Pivotní element má hodnotu $\frac{3}{2}$, zbývá tedy provést eliminaci, jejímž výsledkem je následující tabulka:

	z	x_1	x_2	x_3	x_4	řešení
z	1	0	0	$\frac{1}{3}$	$\frac{4}{3}$	100
x_1	0	1	0	$\frac{2}{3}$	$\frac{-1}{3}$	20
x_2	0	0	1	$\frac{-1}{3}$	$\frac{1}{6}$	20

Vzhledem k tomu, že koeficienty u nebázických proměnných v účelové funkci jsou záporné, je hledané řešení optimální a hodnota účelové funkce maximální.

$$x_1 = 20,\ x_2 = 20,\ z = 100.$$

3 CELOČÍSELNÉ PROGRAMOVÁNÍ

V celočíselném programování se dle [2] zabýváme úlohami, při jejichž řešení potřebujeme, aby některé proměnné byly celočíselné. Požadavek je zcela logický, nemůžeme prodat 3,7 brýlí nebo vyrobit 48,6 klavírů. Situace je sice možné řešit buď zaokrouhlením, nebo zanedbáním čísel za desetinnou čárkou, nicméně takové kroky vedou často na značně chybná, nepřesná, ba dokonce nepřípustná řešení.

Z výše uvedených příkladů je zřejmé, že hovoříme o rozhodovacích proměnných, jejichž základní jednotky nelze dále fyzicky dělit. Úlohy s nedělitelnostmi jsou neopomenutelnou částí úloh celočíselné optimalizace. Kromě úloh s nedělitelnostmi lze však s pomocí celočíselného programování řešit další komplikované úlohy, které nelze řešit jiným způsobem vůbec, nebo alespoň dostatečně efektivně, jak uvádí [2]. Typem takových úloh jsou kombinatorické problémy. Při řešení úloh tohoto typu hledáme řešení z konečné množiny přípustných řešení, které optimalizujeme podle dané účelové funkce. Kromě problému obchodního cestujícího můžeme za příklady považovat například přiřazovací úlohy a úlohy rozvrhování.

3.1 Obecná formulace úlohy celočíselného lineárního programování

Aplikace celočíselného programování je specifická tím, že alespoň jedna proměnná má celočíselnou povahu, ve speciálním případě binární povahu, jak lze nalézt v [5]. Nechť existují vektory $\mathbf{x} = (x_j)$, $\mathbf{c} = (c_j)$, $\mathbf{b} = (b_i)$ a matice $\mathbf{A} = (a_{ij})$ typu $\mathsf{m} \times \mathsf{n}$, $i = 1, ..., m$, $j = 1, ..., n$. Nechť dále a_{ij}, b_i, c_j jsou daná reálná čísla a $x_j \in I$, $j \in J_I$. Poté lze úlohu

$$\begin{aligned} \max \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x}, \\ \text{za podmínky} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

nazvat maximalizační úlohou celočíselného lineárního programování. Dále lze soubor úloh dle [5] rozdělit do tří podkategorií takto:

- **Celočíselné lineární programování** (CLP) - druh lineárního programování, kdy všechny proměnné mají celočíselnou povahu: $I = \mathbb{Z}, J_I = \{1, ..., n\}$,
- **Smíšené celočíselné lineární programování** (SCLP) - druh lineárního programování, kdy některé, zpravidla hlavní rozhodovací proměnné, mají celočíselnou povahu: $I = \mathbb{Z}, J_I \subset \{1, ..., n\}$,
- **Binární celočíselné lineární programování** - (BCLP) speciální případ, kde proměnné mají binární (někdy také 0-1) povahu: $I = \{0, 1\}, J_I = \{1, ..., n\}$.

4 ÚLOHA OBCHODNÍHO CESTUJÍCÍHO

Před uvedením samotné formulace úlohy obchodního cestujícího považuji za vhodné uvést přiřazovací problém. Uvedení této úlohy má své opodstatnění. Úlohu obchodního cestujícího lze totiž na úlohu o přiřazovacím problému relaxovat, nebo lze opačně využít úlohy o přiřazovacím problému jako součást formulace úlohy obchodního cestujícího.

4.1 Přiřazovací problém

Tuto formulaci lze nalézt taktéž v [9]. Máme n lidí schopných vykonat n pracovních úkolů. Každá osoba je přiřazena k vykonání právě jednoho pracovního úkolu. Tento fakt lze zapsat do matice $\mathbf{X} = (x_{ij})$, jejímiž prvky jsou 0 a 1. Někteří lidé jsou pro určité úkoly způsobilejší než ostatní, proto uvažujeme cenu c_{ij}, reprezentující zisk, jestliže je osoba i přiřazena k vykonání práce j. Úkolem je najít maximální cenu přiřazení. Formulace je následující:

$$\text{maximalizovat} \quad f(\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{4.1}$$

za podmínek

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad i = 1, ..., n, \tag{4.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad j = 1, ..., n, \tag{4.3}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., n, \tag{4.4}$$

kde

$$x_{ij} = \begin{cases} 1 & \text{pokud osoba } i \text{ vykonává práci } j, \\ 0 & \text{v opačném případě.} \end{cases}$$

Cílem úlohy je samozřejmě minimalizace nákladů na vykonání jistého množství pracovních úkonů. Význam omezení je následující. Omezení 4.2 vyjadřuje, že každá osoba i vykonává pouze jeden pracovní úkon, omezení 4.3 vyjadřuje, že každý pracovní úkon j je vykonáván pouze jednou osobou.

Ve své podstatě se jedná o speciální případ dopravní úlohy, viz [9]. Samozřejmě nelze opomenout fakt, že bude-li řešena úloha zaměstnání 100 lidí na 100 pozicích, vznikne úloha s 10.000 proměnnými. Vzhledem k tomuto faktu je vhodnější užít nějaký specializovanější algoritmus pro transportní úlohy. Na efektivní algoritmy řešení této úlohy odkazuje zdroj [9].

4.2 Úloha obchodního cestujícího

Do první poloviny 19. století se objevují knihy pojednávající o úlohách podobného typu, nikoli však s matematickým popisem. Postupem času si jako školský příklad získala mnoho příznivců, neboť je snadné ji formulovat a zároveň je velmi lákavé ji vyřešit. Pro snazší představitelnost lze najít vhodnou analogii s prací poštovního doručovatele nebo s řidičem nákladního vozidla majícího seznam klientů, které musí navštívit.

Máme dáno $n+1$ měst a známe matici $\mathbf{C} = (c_{ij})$ vzdáleností mezi těmito městy. Matice nemusí být nutně symetrická, jestliže pojem vzdálenosti nahradíme náročností trasy, je například podstatně náročnější jít z města A do města B do kopce nežli naopak. Pro jednoduchost se však omezíme pouze na symetrickou matici vzdáleností.

Při sestavování tohoto modelu jsem využil zdroj [8]. Řekněme například, že trasa obchodního cestujícího začíná v městě 0. Cílem je navštívit všechna města, ideálně právě jednou, a zároveň překonat co nejmenší vzdálenost a cestu zakončit opět ve městě 0. Matematicky lze tuto úlohu popsat tímto způsobem:

$$\text{minimalizovat} \quad f(\mathbf{X}) = \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{4.5}$$

za podmínek

$$\sum_{\substack{j=0 \\ i \neq j}}^{n} x_{ij} = 1, \qquad j = 0, 1, ..., n, \tag{4.6}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{n} x_{ij} = 1, \qquad i = 0, 1, ..., n, \tag{4.7}$$

$$u_i - u_j + n x_{ij} \leq n - 1, \qquad i, j = 1, 2, ..., n, i \neq j, \tag{4.8}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 0, 1, ..., n, \tag{4.9}$$

kde

$$x_{ij} = \begin{cases} 1 & \text{pokud obchodní cestující cestuje z města } i \text{ do města } j, \\ 0 & \text{v opačném případě.} \end{cases} \tag{4.10}$$

$$u_i, u_j \in \mathbf{N}$$

Způsobů jak zvolit hodnoty u_i a u_j je mnoho, jeden bude uveden a vysvětlen závěrem kapitoly. Podmínka (4.6) vyjadřuje, že obchodní cestující opouští každé město právě jednou a zároveň podmínka (4.7) vyjadřuje, že obchodní cestující přichází do každého města právě jednou. Pokud by byla podmínka (4.8) opomenuta, bylo by možné obdržet jako řešení několik subtras (viz obr. 4.1a), které nebudou navzájem

propojeny. Taková řešení samozřejmě považujeme za nevhodná a je naší snahou je eliminovat.

Z tohoto důvodu se zavádí proměnné u_i. Omezení navíc se přidá ve tvaru (4.8). Cílem bude nyní na příkladu ukázat, že omezení (4.8) neumožňuje vznik subtras, které neobsahují startovací město 0 (logicky řešením bude trasa přes všech n měst), proto budeme nyní uvažovat pouze dané omezení, nikoli celé zadání. Mějme 6 měst, která je třeba navštívit, a řešení na obr. 4.1a. Pro subtrasu nezahrnující město 0 lze formulovat tyto nerovnice:

$$u_2 - u_5 + 6x_{25} \leq 5, \tag{4.11}$$

$$u_6 - u_2 + 6x_{62} \leq 5, \tag{4.12}$$

$$u_5 - u_6 + 6x_{56} \leq 5, \tag{4.13}$$

Jak lze vidět na obr. 4.1a, proměnné x_{25}, x_{62}, x_{56} nabývají hodnoty 1. Pokud dané nerovnice sečteme, získáme

$$18 \leq 15.$$

Evidentně nerovnost neplatí a řešení z obr. 4.1a nevyhovuje novým omezením. Podobně lze pracovat se všemi subtrasami jiných příkladů, které neobsahují startovací město 0.

Zároveň je vhodné ukázat, že nová omezení nevylučují všechna přípustná řešení. Existuje mnoho způsobů, jak volit hodnoty proměnných u_i, avšak my se zaměříme na tu nejpřirozenější:

$$u_i = k,$$

jestliže navštívíme i-té město na k-tém kroku trasy ($k = 1, 2, ...n$). Pokud bychom se například rozhodli aplikovat tuto volbu na celou trasu z obr. 4.1b, proměnné u_i by měly tyto hodnoty:

$$u_3 = 1 \qquad u_4 = 2 \qquad u_2 = 3$$
$$u_6 = 4 \qquad u_5 = 5 \qquad u_1 = 6$$

Zřejmě výše zmíněná volba vyhovuje omezením (4.8). Podobně se můžeme zachovat při řešení všech příkladů tohoto typu.

Problémem této formulace je však fakt, že neexistuje nic jako proměnná u_0, a proto žádné omezení nemůže zahrnout $i = 0$ nebo $j = 0$. Není tedy možné sestavit podobné nerovnice jako (4.11) - (4.13) pro subtrasy obsahující město 0. Pokud neexistují žádné subtrasy, musí celková trasa město 0 obsahovat.

(a) Nepřípustné řešení (b) Přípustné řešení

Obr. 4.1: Ukázka možných řešení

Velikost celočíselného modelu generovaného touto formulací určíme dle [8] následujícím způsobem: Mějme problém s $n+1$ městy. Tento model má pak:

$$\begin{aligned} n(n+1) \quad & x_{ij} \quad \text{proměnných,} \\ n \quad & u_i \quad \text{proměnných,} \\ n+1 \quad & \text{omezení typu (4.7),} \\ n+1 \quad & \text{omezení typu (4.6),} \\ n(n-1) \quad & \text{omezení typu (4.8).} \end{aligned}$$

Jako zobecnění a speciální případy můžeme dle [8] uvést následující. Obchodní cestující nemusí skončit svoji cestu ve výchozím městě. Dalším zvláštním případem je úloha s nesymetrickou maticí náročností cest, jak bylo uvedeno dříve. „Vehicle routing problem“ může být zredukován na úlohu obchodního cestujícího. „Vehicle routing problem“ je problém organizace dodávek zákazníkům, při němž využíváme množství aut různých velikostí. K tomu opačnou úlohou je „problem of picking up deliveries“. Stejně tak „job sequencing problems“ za účelem stanovení minimální ceny může být přeformulován na úlohu TSP, kde „vzdálenost mezi městy“ reprezentuje „set-up cost between operations“.

Problém obchodního cestujícího je možné řešit úplnou enumerací, tj. celkovým vyčíslením všech možných možností, ale lze zároveň ukázat, že počet možností řešení úlohy je $(n-1)!$, kde n je počet měst, tedy pro 101 měst existuje zhruba 9.33×10^{157} řešení, viz [8]. Proto volíme vhodnější postupy hledání řešení, viz formulace (4.5) - (4.10).

5 GAMS

Pro výpočty byl používán program GAMS (The General Algebraic Modeling System). Důvodem této volby je dobrá dostupnost, nepříliš složitá syntaxe a uživatelská přívětivost, viz [6]. GAMS je ve své podstatě optimalizační software s vlastním programovacím jazykem. Obsahuje řešiče pro řešení úloh optimalizace lineární (kap. 2), nelineární, celočíselné (kap. 3), binární, ale i spousty dalších typů, které lze nalézt v [10].

GAMS má mnoho důležitých vlastností. Mezi ně beze sporu patří možnost sestavování cyklů, rozhodovacích podmínek a omezení. Protože se při optimalizaci často pracuje s rozměrnými objekty (vektory, matice, ...), důležitá je možnost importovat data z programu MS Excel a zároveň exportovat data do textového editoru, pro další práci, nebo do LaTeXu, pokud je záměrem znázornit výsledky.

GAMS je volně dostupný software, který lze stáhnout z oficiálních stránek GAMSu www.gams.com. Zároveň je velmi podstatné zmínit, jaký z řešičů byl nakonec použit při zpracovávání dat. Jedná se o CPLEX, mezi jehož přednosti patří stabilita, značná rychlost a schopnost vyřešit většinu úloh lineárního programování bez zásahů uživatele. Řešič využívá k výpočtům simplexovou metodu, případně duální simplexovou metodu, viz [6].

```
----     45 PARAMETER c

            i1          i2          i3          i4          i5          i6

i1                  19.000      25.000      38.000      44.000      57.000
i2          19.000                6.000      19.000      25.000      38.000
i3          25.000       6.000              13.000      19.000      32.000
i4          38.000      19.000      13.000                6.000      19.000
i5          44.000      25.000      19.000       6.000                 13.000
i6          57.000      38.000      32.000      19.000      13.000
i7          10.000      29.000      35.000      48.000      54.000      67.000
i8           5.000      24.000      30.000      43.000      49.000      62.000
```

Obr. 5.1: Ukázka výpisu dat načtených GAMSem

GAMS má pevně definovanou délku řádku, což představuje malý problém. Jestliže bude cílem použít data pro rozměrnější úlohy, jakou je například úloha z kapitoly 6, je snazší použít pro načítání dat adekvátně upravenou verzi kódu z příkladu 5.1. Pokud bychom data chtěli zapsat maticovým způsobem, bylo by třeba rozměrná data rozdělit na menší celky a ty postupně zapisovat pod sebe a skládat. Úloha, která je zpracovávána v kapitole 6 zahrnuje matici 662×662. Z tohoto důvodu jsem se rozhodl pro předpřipravení dat v MS Excel a jejich následné načtení pomocí kódu v příkladu 5.1.

5.1 Propojení GAMS - Excel

Na začátku kapitoly bylo řečeno, že jednou z předností GAMSu je možnost importovat data z prostředí MS Excel, neboť při práci s rozměrnými objekty je jejich zápis v jazyce GAMS poměrně komplikovaný. Navíc, pracujeme-li s daty v Excelu, lze se v nich mnohem snáze orientovat a jednoduše je upravovat, dojde-li k chybě, a zároveň nedochází ke zbytečnému prodlužování zdrojového kódu. Zároveň lze kromě importu provést i export do Excelu.

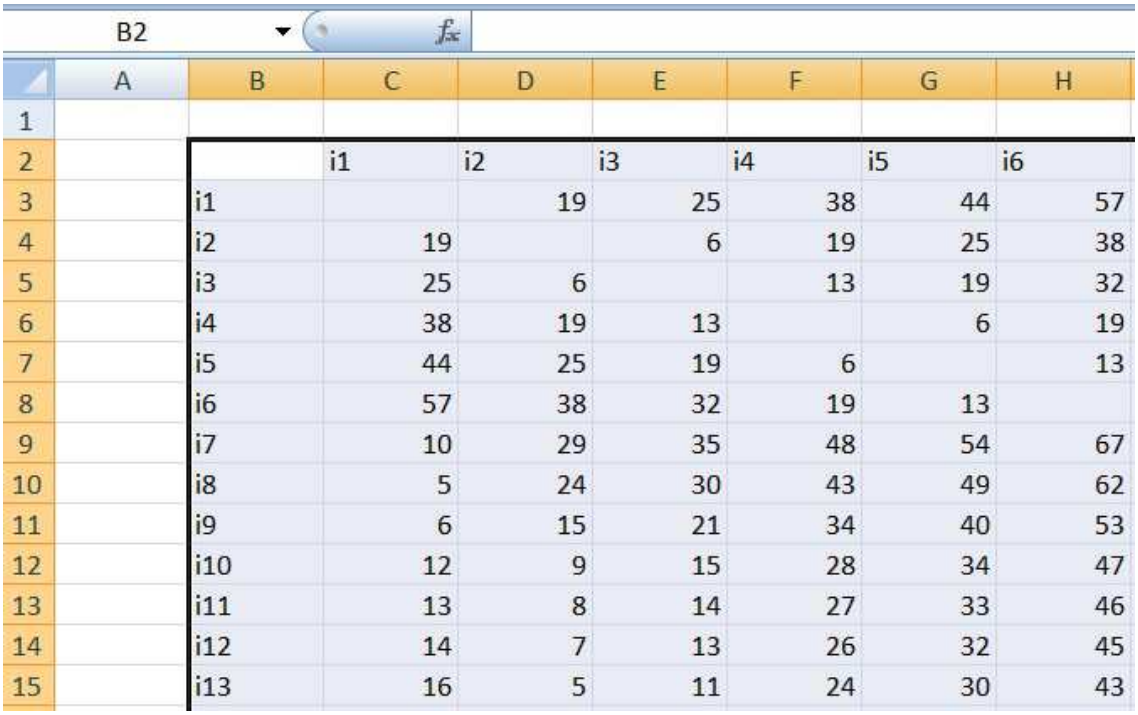
Při zpracování dat bylo nutné vybrat jednu z několika variant, jak přeložit data ze souboru vytvořeného v Excelu do souboru, který je pro GAMS čitelný. Byl užit nástroj GDXXRW, který v tomto případě posloužil především k importu dat.

Příklad 5.1 *Ukázka aplikace nástroje GDXXRW při načítání matice z* `tsp2.gms`*:*

```
parameter c(ii,jj);
$call gdxxrw.exe xql662.xlsx par=c rng=xql662
$gdxin xql662.gdx
$load c
$gdxin
```

Ze všeho nejdříve je třeba mít v GAMSu definovány množiny indexů ii a jj. Pak dle formulace naší úlohy vytvoříme příslušný parametr, v tomto případě $c(ii, jj)$. Druhý řádek spustí nástroj GDXXRW, který ze souboru `xql662.xlsx` vytvořeného v Excelu, z něhož načítáme data, načte všechna data, která mají příslušné označení, tj. označení uzlu $(i_1, i_2, ...)$, viz obr. 5.2. Označení ve zdrojovém kódu GAMSu a v Excelu musí být totožné. Ze souboru `xql662.xlsx` bude vytvořen soubor `xql662.gdx`, který už GAMS zvládne načíst. Při opakované práci s týmiž daty je vhodné tento řádek zapoznámkovat pomocí `*` na začátku řádku, neboť jinak by při opětovném spuštění programu byl daný soubor vytvářen znovu, což může být časově náročné. Zároveň je vhodné zmínit, že část příkazu `rng=xql662` načítá celou tabulku hodnot a že nezáleží na jejím umístění v Excelu, příkaz `par=c` zapisuje načítané hodnoty do definovaného parametru $c(ii, jj)$. Samozřejmě lze vybrat jen určitou část dat, například pomocí příkazu `rng=A1:D3` (rng znamená range, česky oblast), ale tento postup již vyžaduje jistou obezřetnost při práci s daty vzhledem k jejich umístění v tabulce.

Příkazem `$gdxin xql662.gdx` se určí, že bude pracováno s daty z vytvořeného souboru. Příkaz `$load c` příslušná data načte. Načítání bude ukončeno příkazem `$gdxin`.



	A	B	C	D	E	F	G	H
1								
2			i1	i2	i3	i4	i5	i6
3		i1		19	25	38	44	57
4		i2	19		6	19	25	38
5		i3	25	6		13	19	32
6		i4	38	19	13		6	19
7		i5	44	25	19	6		13
8		i6	57	38	32	19	13	
9		i7	10	29	35	48	54	67
10		i8	5	24	30	43	49	62
11		i9	6	15	21	34	40	53
12		i10	12	9	15	28	34	47
13		i11	13	8	14	27	33	46
14		i12	14	7	13	26	32	45
15		i13	16	5	11	24	30	43

Obr. 5.2: Ukázka zápisu dat v prostředí MS Excel

5.2 Příprava dat v MS Excel

Vzhledem k tomu, že jedním z cílů bylo ukázat náročnost řešení této úlohy pomocí celočíselného lineárního programování, byla snaha vybrat rozměrnější data. Jelikož se nepovedlo najít dostatečně uspokojivá data v maticovém tvaru, byla vybrána data xql662 z [7] ve formátu sady souřadnic pro daný počet míst. Souřadnice bylo nutno zpracovat tak, aby výstupem byla matice vzdáleností. Protože data jsou uvažována jako plošná, byla při výpočtu použita klasická euklidovská norma pro výpočet vzdálenosti dvou bodů:

$$| AB | = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Zpracování dat bylo uskutečněno pomocí příkazu:

```
= KDYŽ( ZAOKR.NAHORU(ODMOCNINA(ABS(((SVYHLEDAT($E2;$A$2:$C$663;2;0) -
SVYHLEDAT(F$1;$A$2:$C$663;2;0))^2 + (SVYHLEDAT($E2;$A$2:$C$663;3;0) -
SVYHLEDAT(F$1;$A$2:$C$663;3;0))^2)));1) = 0;"";ZAOKR.NAHORU(ODMOCNINA
(ABS(((SVYHLEDAT($E2;$A$2:$C$663;2;0)-SVYHLEDAT(F$1;$A$2:$C$663;2;0))
^2+(SVYHLEDAT($E2;$A$2:$C$663;3;0)-SVYHLEDAT(F$1;$A$2:$C$663;3;0))^2)
));1))
```

Nejprve se pomocí příkazu `KDYŽ` definuje, že bude diagonála matice prázdná, tzn. že vzdálenost z města A do města A je 0, nicméně pro lepší práci v GAMSu zůstane dané pole prázdné. Z důvodu přehlednosti, protože hodnoty nevycházely celočíselně,

bylo zvoleno zaokrouhlení nahoru definované v Excelu. Příkaz `ABS` má spíše pojistný charakter a jeho implementace je spíše formální. Zbývá vysvětlit příkaz `SVYHLEDAT`. Jedná se o vyhledávací funkci, která nejprve zjistí, jaké město se nachází v i-tém řádku matice vzdáleností, dále `$A$2:$C$663` definuje, z které oblasti v Excelu budou hodnoty načítány. Dále čísla `2` a `3` definují, z kterých sloupců výše definované oblasti budou data načítána. V příkazu `SVYHLEDAT` se jako poslední přidává `0`, pokud data nejsou seřazena.

S pomocí tohoto příkazu formulovaného v jedné buňce a jeho nakopírováním do ostatních buněk obdržíme příslušnou matici vzdáleností s prázdnou diagonálou. Zároveň je vhodné tabulku zkopírovat do textového editoru a zpětně vložit do nového souboru v Excelu, neboť tak budou zkopírovány pouze hodnoty, a to nezávisle na použitých příkazech, a zároveň si uživatel může dovolit vynechat základní data, která při načítání pomocí příkazu `GDXXRW` mohou způsobovat potíže.

5.3 Jádro

Podstatnou částí práce je práce s daty, proto by neměl chybět ani komentovaný zdrojový kód, který bude nyní uveden a okomentován. Lze jej také nalézt mírně upravený v [11].

```
set        ii              body / i1*i662 /
           i(ii)           podmnožina bodů
alias      (ii,jj),(i,j);
set        ij(ii,jj)       vynechá první sloupec a řádek;
parameter  c(ii,jj);
```

Nejprve je třeba vytvořit si množiny bodů a parametry. Zavedeme množinu `ii` obsahující body `i1` až `i662` a její podmnožinu `i(ii)`. Dále pomocí příkazu `alias(ii,jj), (i,j)` vytvoříme množiny `jj` a `j`, které jsou stejné jako `ii` a `i`. Definujeme dále množinu `ij` a parametr `c`.

```
$call gdxxrw.exe xql662.xlsx par=c rng=B2:YN664
$gdxin xql662.gdx
$load c
$gdxin
```

Tato část slouží pro načítání dat z Excelu a byla vysvětlena v části 5.1.

```
ij(ii,jj) = ord(ii)>1 and ord(jj)>1;
```

Příkaz `ord` přiřazuje pořadí danému bodu v množině, například pro bod `i1` platí `ord(i1) = 1` atd. V tomto případě, jestliže žádáme hodnotu `>1` pro `ii` a `jj`, říkáme, že vycházíme z prvního města.

```
variables        x(ii,jj)  rozhodující proměnná - délka trasy
                 z         ucelova funkce;
binary variable  x;
```

Zde definujeme proměnné `x` a `z`, kde `x` představuje rozhodující binární proměnnou. Její hodnota bude 1, jestliže obchodní cestující bude cestovat z města `ii` do města `jj`, 0 v opačném případě. Účelovou funkci představuje proměnná `z`.

```
equations  objective   celková délka trasy
           rowsum(ii)  opuštění každého města právě jednou
           colsum(jj)  příchod do každého města právě jednou;
variable   u(ii)       strategie eliminace subtras 3
equation   se(ii,jj)   omezení pro eliminaci subtras;
```

Pomocí příkazu `equations` se definují názvy jednotlivých rovnic, které bude třeba při řešení modelu vypočítat.

```
se(ij(i,j))..  u(i) - u(j) + card(i)*x(i,j) =l= card(i) - 1;
objective..    z =e= sum((i,j), c(i,j)*x(i,j));
rowsum(i)..    sum(j, x(i,j)) =e= 1;
colsum(j)..    sum(i, x(i,j)) =e= 1;
```

Zde jsou zapsána omezení a účelová funkce z podkapitoly 4.2.

```
model  tsp / objective, rowsum, colsum, se /;
       i(ii) = ord(ii) <= 20;
```

Pomocí příkazu `model` označíme model, zde jako `tsp` a dále mezi lomítky určujeme, které rovnice chceme při jeho řešení počítat.

```
option  optcr=0.00;
option  ResLim = 100000000
option  IterLim = 1000000000;
```

Tato část je volitelná, nicméně podstatná. GAMS má totiž v základním nastavení omezení na výpočetní čas, paměť i množství iterací. To lze pomocí příkazu `option` předefinovat. Zde pomocí `ResLim` upravujeme množství využívané paměti a pomocí

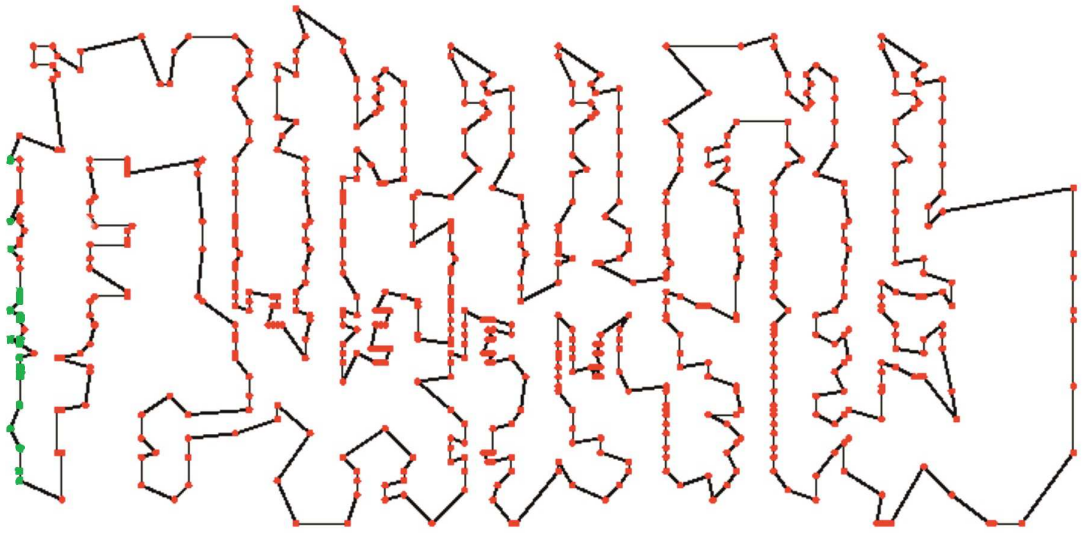
`IterLim` upravujeme množství iterací, které lze využít. Dále lze pomocí příkazu `optcr` nastavit relativní toleranci při hledání řešení, jestliže je hodnota nastavena na 0, pak hledáme optimální řešení.

```
solve    tsp min z using mip;
display  x.l;
```

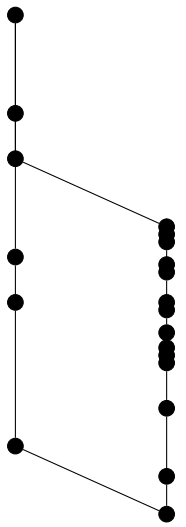
Zde příkaz `solve` slouží k vyřešení dříve definovaného modelu `tsp`. Minimalizována je hodnota účelové funkce `min z`. K minimalizaci se užívá smíšené celočíselné programování, zapsáno jako `using mip`. Příkaz `display` slouží k zobrazení výsledku, tedy GAMS na výstupu kromě hodnoty účelové funkce zobrazí také trasu vyjádřenou pomocí proměnné `x`.

6 TESTOVACÍ PŘÍKLAD

Aby byla dobře popsána a vysvětlena složitost a náročnost této formulace, byla vybrána data s názvem xql662 z [7]. Vzhledem k jejich původnímu souřadnicovému formátu byla zpracována způsobem zmíněným v části 5.1. Pro zajímavost nyní uvedu optimální řešení celé úlohy vyřešené řešičem Concorde, viz [7], a dílčí optimální řešení, které se podařilo vypočítat s pomocí programu GAMS.



Obr. 6.1: Úplné řešení úlohy xql662 dle [7]

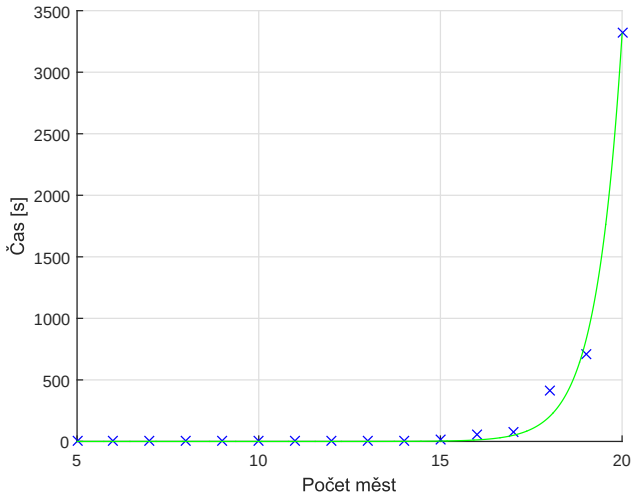


Obr. 6.2: Vypočítané dílčí optimální řešení úlohy xql662

Počet měst	Čas [s]	Počet iterací	Počet měst	Čas [s]	Počet iterací
5	0,685	85	13	2,316	25075
6	1,537	254	14	4,393	60181
7	1,136	327	15	18,156	395824
8	0,897	370	16	53,191	927502
9	0,938	617	17	76,477	1360011
10	1,239	3899	18	412,907	5420780
11	1,249	3673	19	704,085	8184507
12	1,246	11499	20	3326,349	27704237

Tab. 6.1: Tabulka dat pro hledání optimálního řešení

Data jsem zpracovával v programu GAMS. Důsledkem této volby i zvolené formulace je poměrně pomalý průběh výpočtu. Výpočet byl uskutečněn pouze pro prvních 20 měst z daného souboru (v obr. 6.1 vyznačena zeleně). Pro ilustraci výpočtu bylo dílčí optimální řešení vykresleno do obr. 6.2. Na obrázek je třeba se dívat velmi detailně, neboť optimální řešení celého problému v porovnání s jeho dílčí variantou zkreslují. Dílčí optimální řešení představuje první a spodní polovinu druhého sloupce bodů. Navíc, aby bylo dílčí řešení zobrazitelné a jednotlivé body dobře viditelné, bylo nutné y-ové složky souřadnic bodů zmenšit na desetinu.

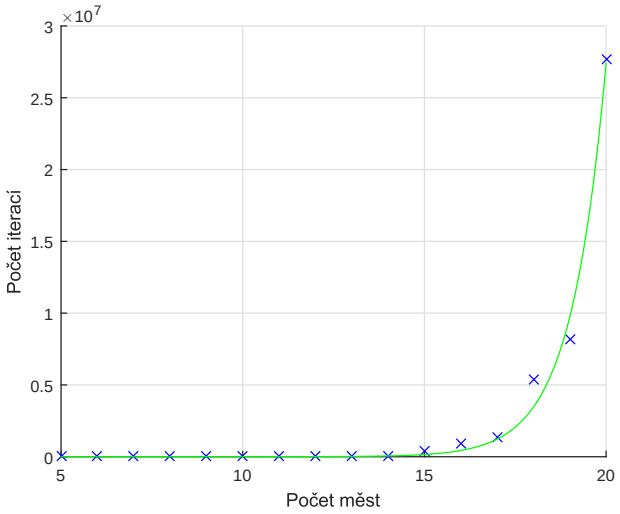


Obr. 6.3: Optimální řešení úlohy xql662

Důvod hledání optimálního řešení pro takto podstatně redukovaný problém je pragmatický. Jak lze vidět v 6.1, výpočet pro 20 měst trval podstatně déle. Pokus

vypočítat optimální trasu pro 25 měst byl po dni výpočtů ukončen, neboť držela-li by se časová náročnost výpočtu trendu z 6.3, trval by výpočet přibližně týden a bylo by třeba vypočítat zhruba dvě miliardy iterací. Naměřená data jsem se rozhodl zpracovat pomocí matematického softwaru Matlab. Pro proložení jsem zvolil exponenciální křivku sestrojenou pomocí metody nejmenších čtverců.

Z 6.4 a 6.3 lze vidět pomalou vzestupnou tendenci v počátcích výpočtu. Důvodem je fakt, že program musel zpracovat relativně malé množství proměnných a omezení. Hlavní vliv na tento průběh mělo aktuální vytížení počítače. V pozdějších fázích výpočtu, při hledání optimální cesty pro patnáct a více měst se do popředí dostala náročnost úlohy způsobená značných objemem dat. Naměřené časy jsou uvažovány pouze pro samotný výpočet. Součástí těchto hodnot tedy není čas nutný k překladu dat mezi MS Excel a GAMS.



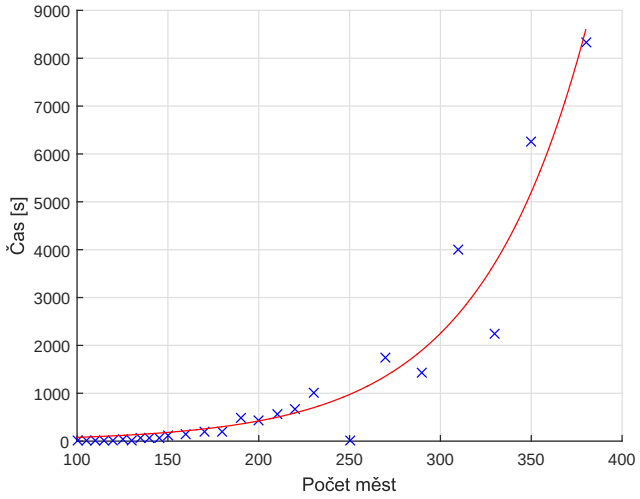
Obr. 6.4: Optimální řešení úlohy xql662

Během výpočtu samozřejmě vyvstala otázka, zdali by nemělo smysl místo optimálního řešení hledat jakékoli řešení z množiny přípustných řešení. Pokud bychom se vrátili zpět do podkapitoly 2.3, kde je popsána ukázková množina přípustných řešení a kapitoly 4, kde je formulována úloha obchodního cestujícího, bylo by zřejmé, že takovým přípustným řešením je řešení pro konstantní účelovou funkci. Data z tohoto výpočtu jsou uvedena v 6.2.

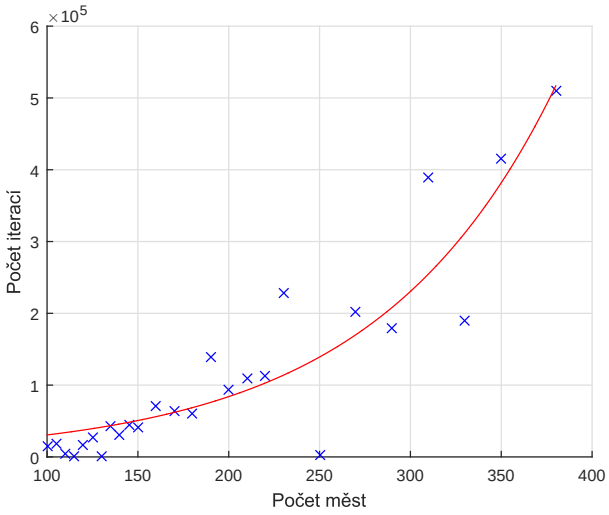
Počet měst	Čas [s]	Počet iterací	Počet měst	Čas [s]	Počet iterací
100	14,312	14348	180	189,651	59880
105	20,231	17926	190	494,331	138556
110	9,315	3610	200	426,191	93420
115	0,741	856	210	560,838	109018
120	23,93	16730	220	672,716	112709
125	38,22	27865	230	1002,714	227677
130	1,74	890	250	4,183	2303
135	65,956	43527	270	1745,107	201739
140	59,763	31080	290	1428,961	178663
145	78,017	44558	310	3997,651	389816
150	105,669	40706	330	2238,315	189723
160	155,725	70350	350	6249,536	416215
170	187,398	63667	380	8328,573	510493

Tab. 6.2: Tabulka dat pro hledání jakéhokoli přípustného řešení

I z těchto dat je znatelný exponenciální trend, zpočátku opět zkreslovaný především vytížeností počítače. V grafu jsou znatelné skoky způsobené odebíráním většího množství výpočetní paměti. Při porovnání dvojic grafů 6.4 a 6.6, 6.3 a 6.5 je zřetelně viditelné, že k nalezení libovolného přípustného řešení je třeba podstatně menší počet iterací a výpočetního času.



Obr. 6.5: Přípustné řešení úlohy xql662



Obr. 6.6: Přípustné řešení úlohy xql662

7 JINÉ FORMULACE ÚLOHY OBCHODNÍHO CESTUJÍCÍHO

Úlohu obchodního cestujícího lze formulovat více způsoby, z nichž některé lze nalézt v [9] a v [4]. Nabízí se tedy možnost porovnat formulace z hlediska efektivity výpočtu. Rozhodl jsem se pro tzv. Dantzigovu formulaci úlohy obchodního cestujícího, kterou lze nalézt v [4]. Tato formulace je již naprogramována, je dostupná z [11], a proto ji využiji.

7.1 Dantzigova formulace

V této podkapitole uvedeme formulaci naprosto stejné úlohy jako v podkapitole 4.2, ale trochu jiným způsobem. G. Dantzig úlohu formuloval takto:

$$\text{minimalizovat} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{7.1}$$

za podmínek

$$\sum_{\substack{j \\ i\neq j}}^{n} x_{ij} = 1, \qquad j = 1, ..., n, \tag{7.2}$$

$$\sum_{\substack{i \\ i\neq j}}^{n} x_{ij} = 1, \qquad i = 1, ..., n, \tag{7.3}$$

$$\sum_{\substack{i\in M \\ j\in \overline{M}}} x_{ij} \geq 1, \qquad \forall M \subset N, \{1\} \notin M, \overline{M} = N - M, \tag{7.4}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., n, \tag{7.5}$$

kde

$$x_{ij} = \begin{cases} 1 & \text{pokud obchodní cestující cestuje z města } i \text{ do města } j, \\ 0 & \text{v opačném případě.} \end{cases} \tag{7.6}$$

$\mathbf{C} = (c_{ij})$ představuje matici vzdáleností mezi jednotlivými body. $N = \{1, 2, ..., n\}$ je množství bodů.

7.2 Variace Dantzigovy formulace

Předchozí formulaci lze mírně upravit, jestliže by omezení (7.4) bylo nahrazeno jiným, c_{ij} i N zde mají stejný význam jako v předchozí podkapitole. Jinou variantu

formulace z podkapitoly 7.1 lze dle [4] zapsat takto:

$$\text{minimalizovat} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{7.7}$$

za podmínek

$$\sum_{\substack{j \\ i\neq j}}^{n} x_{ij} = 1, \qquad j = 1, ..., n, \tag{7.8}$$

$$\sum_{\substack{i \\ i\neq j}}^{n} x_{ij} = 1, \qquad i = 1, ..., n, \tag{7.9}$$

$$\sum_{\substack{i,j\in M \\ i\neq j}} x_{ij} \leq |M| - 1, \qquad \forall M \subset N, \{1\} \notin M, |M| \geq 2, \tag{7.10}$$

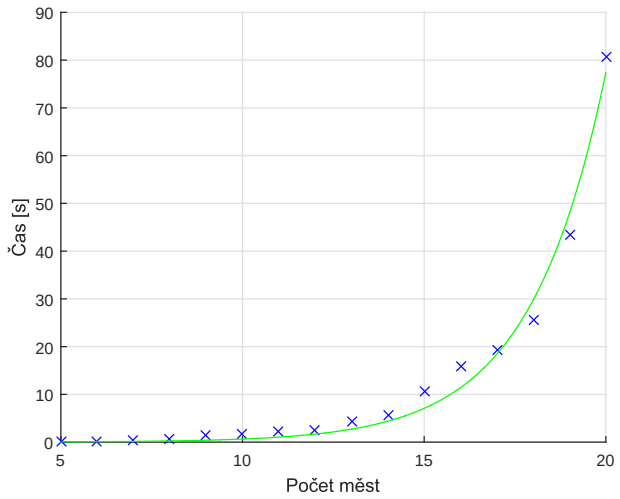
$$x_{ij} \in \{0, 1\}, \qquad i, j = 1, ..., n, \tag{7.11}$$

kde

$$x_{ij} = \begin{cases} 1 & \text{pokud obchodní cestující cestuje z města } i \text{ do města } j, \\ 0 & \text{v opačném případě.} \end{cases} \tag{7.12}$$

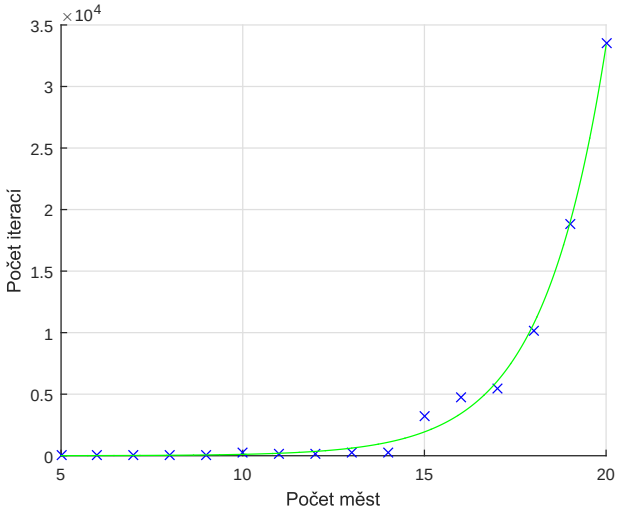
Jak uvádí [4], tato formulace má $2^n + 2n - 2$ omezení a $n(n-1)$ proměnných x_{ij}. Vzhledem ke značnému množství omezení je tedy nepraktické úlohu řešit přímo. Z tohoto důvodu se obvykle používají přiřazovací omezení (7.8) - (7.9) a dále se připojují omezení (7.10), když jsou porušena.

Vzhledem k tomu, že tato formulace je již v [11] naprogramována, dovolím si ji využít k výpočtům, díky nimž budu moci porovnat formulaci z podkapitoly 4.2.



Obr. 7.1: Optimální řešení úlohy xql662 podle Dantzigovy formulace

Pokud bychom porovnali grafy časů hledání optimálního řešení a počtů iterací nutných k jeho nalezení s grafy z kapitoly 6, je znatelné, že Dantzigova formulace je v tomto ohledu efektivnější.



Obr. 7.2: Optimální řešení úlohy xql662 podle Dantzigovy formulace

Podobně jako v kapitole 6 byla k sestrojení grafů použita metoda nejmenších čtverců.

8 ZÁVĚR

V mé práci jsou shrnuty základní poznatky problematiky modelování dopravních úloh. Důraz je kladen na formulaci a řešení úlohy obchodního cestujícího s využitím celočíselného lineárního programování.

Ve své práci nejprve uvádím několik historických zajímavostí týkajících se úlohy obchodního cestujícího. Dále pak uvádím obecnou formulaci úlohy lineárního programování, na níž navazuje krátký příklad řešený graficky a s pomocí simplexové metody.

Na obecnou formulaci úlohy celočíselného lineárního programování navazuje formulace přiřazovacího problému, speciálního případu úlohy obchodního cestujícího, a formulace úlohy obchodního cestujícího, jejíž princip je vysvětlen.

Výpočty byly uskutečněny v programu GAMS, a tak nemůže chybět kapitola, kde je GAMS představen. Zároveň je v této kapitole, vzhledem k rozměrnosti zvolených dat, uveden způsob zpracování dat v MS Excel a následně je vysvětleno propojení Excelu a GAMSu.

Kapitoly 6 a 7 mají výpočtový charakter. Na vybraných datech byly provedeny výpočty s cílem najít optimální řešení a porovnat efektivitu formulací úlohy obchodního cestujícího.

LITERATURA

[1] APPLEGATE, David L. *The traveling salesman problem: a computational study*. Princeton: Princeton University Press, 2006. Princeton series in applied mathematics.

[2] KLAPKA, J., DVOŘÁK, J., POPELA, P. *Metody operačního výzkumu*. Vyd. 1. Brno: Vysoké učení technické, 1996.

[3] KŮDELA, J. *Optimalizační úlohy v programu AIMMS*, bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012.

[4] ORMAN, A., WILLIAMS H.P. *A survey of different integer programming formulations of the traveling salesman problem* [online]. [cit. 25-05-2016].

[5] PASCHKEOVÁ (VOTAVOVÁ), H. *Matematické modely dopravních úloh*, bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012.

[6] PROCHÁZKA, V. *Matematický model dopravní úlohy pro oblast odpadového hospodářství*, bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012.

[7] UNIVERSITY OF WATERLOO. *VLSI Data Sets: XQL662* [online]. [cit. 31-03-2016]. Dostupné z: `http://www.math.uwaterloo.ca/tsp/vlsi/index.html`

[8] WILLIAMS, H. P. *Model Building in Mathematical Programming*. Chichester: John Wiley & Sons, 1978.

[9] WOLSEY, Laurence A. *Integer programming*. New York: Wiley, 1998.

[10] *GAMS Documentation 24.7* [online]. [cit. 05-04-2016]. Dostupné z: `http://www.gams.com/help/index.jsp`

[11] *GAMS Model library* [online]. [cit. 13-03-2016]. Dostupné z: `http://gams.com/modlib/libhtml/alfindx.htm`

9 PŘÍLOHY

9.1 Zdrojový kód tsp2.gms

Zde je uveden zdrojový kód hledající optimální řešení úlohy obchodního cestujícího.

```
set         ii              body / i1*i662 /
            i(ii)           podmnožina bodů
alias       (ii,jj),(i,j);
set         ij(ii,jj)       vynechá první sloupec a řádek;
parameter   c(ii,jj);
$call gdxxrw.exe xql662.xlsx par=c rng=B2:YN664
$gdxin xql662.gdx
$load c
$gdxin

ij(ii,jj) = ord(ii)>1 and ord(jj)>1;

variables         x(ii,jj)  rozhodující proměnná - délka trasy
                  z         ucelova funkce;
binary variable   x;
equations   objective   celková délka trasy
            rowsum(ii)  opuštění každého města právě jednou
            colsum(jj)  příchod do každého města právě jednou;
variable    u(ii)       strategie eliminace subtras 3
equation    se(ii,jj)   omezení pro eliminaci subtras;
se(ij(i,j))..  u(i) - u(j) + card(i)*x(i,j) =l= card(i) - 1;
objective..    z =e= sum((i,j), c(i,j)*x(i,j));
rowsum(i)..    sum(j, x(i,j)) =e= 1;
colsum(j)..    sum(i, x(i,j)) =e= 1;

model  tsp / objective, rowsum, colsum, se /;
       i(ii) = ord(ii) <= 20;

option  optcr=0.00;
option  ResLim = 100000000
option  IterLim = 1000000000;

solve    tsp min z using mip;
display  x.l;
```

9.2 Seznam příloh na CD

Data jsou rozdělena do tří adresářů podle příslušných kapitol v textu.

1. GAMS
    - `tsp2.gms` - zdrojový kód úlohy obchodního cestujícího
2. Testovací příklad
    - `tsp2.gms` - zdrojový kód úlohy obchodního cestujícího
    - `xql662.xlsx` - data využívaná k výpočtům
3. Jiné formulace úlohy obchodního cestujícího
    - `tsp42.gms` - zdrojový kód úlohy obchodního cestujícího pro Dantzigovu formulaci
    - `xql662.xlsx` - data využívaná k výpočtům