VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

VYBRANÉ OPTIMALIZAČNÍ MODELY V LOGISTICE

SELECTED OPTIMIZATION MODELS IN LOGISTICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MÁLEK

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. PAVEL POPELA, Ph.D.

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Martin Málek
který/která studuje v **bakalářském studijním programu**
obor: **Matematické inženýrství (3901R021)**

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:

Vybrané optimalizační modely v logistice

v anglickém jazyce:

Selected optimization models in logistics

Stručná charakteristika problematiky úkolu:

Student se seznámí s problematikou optimalizačních modelů v logistice. Zaměří se zejména na úlohy matematického programování s důrazem na návaznost dopravních úloh na související úlohy. Budou studovány vlastnosti vybraných úloh, modifikovány algoritmy a realizovány testovací výpočty s využitím dat z reálných aplikací.

Cíle bakalářské práce:

Předpokládá se studium originálních a úprava existujících optimalizačních modelů a algoritmů a jejich aplikace v návaznosti na problémy řešené v rámci spolupráce ústavu matematiky FSI VUT v Brně se specialisty v oblasti logistiky z norské Molde University College (prof. Kjetil Kare Haugen, prof. Asmund Olstad).

Seznam odborné literatury:
Christofides, N.: Graph Theory - an Algorithmic Approach. Academic Press 1975.
Wolsey, L. A.: Integer programming. John Wiley and Sons, 1998.
Ghiani, G., Laporte, G., Musmano, R.: Introduction to Logistics System Planning and Control. John Wiley and Sons, New York, 2004.

Vedoucí bakalářské práce: RNDr. Pavel Popela, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.
V Brně, dne 21.11.2014

L.S.

prof. RNDr. Josef Šlapal, CSc.
Ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
Děkan fakulty

Abstrakt
Bakalářská práce se zabývá optimalizačními modely v logistice, které jsou řešeny pomocí lineárního programování. Teoretická část je věnována základním pojmům z teorie grafů, optimalizace, lineárního programování a toků v sítích. Praktická část uvádí optimalizační modely, jejich implementací, přehled výsledků a aplikací na řešení úloh s reálnými daty.

Summary
The bachelor thesis focuses on optimization models in logistics, which are solved by linear programming. Theoretical part introduces basic terms of graph theory, optimization, linear programming and network flows. Practical part contains mathematical models, their implementation, and obtained results. Furthermore, real-world application is solved at the end.

Klíčová slova
Teorie grafů, lineární programování, optimalizace, toky v sítích

Keywords
Graph theory, linear programming, optimization, network flows

MÁLEK, M. *Vybrané optimalizační modely v logistice.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 71 s. Vedoucí RNDr. Pavel Popela, Ph.D.

Prohlašuji, že jsem bakalářskou práci Vybrané optimalizační modely v logistice vypracoval samostatně pod vedením RNDr. Pavla Popely, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Martin Málek

Rád bych poděkoval panu RNDr. Pavlu Popelovi, Ph.D. za odborné vedení mé bakalářské práce a jeho věcné připomínky. Dále bych rád poděkoval Ing. Jakubovi Kůdelovi za jeho čas a pomoc s častými dotazy a také děkuji Ing. Radovanu Šomplákovi za poskytnutí reálných dat.

Martin Málek

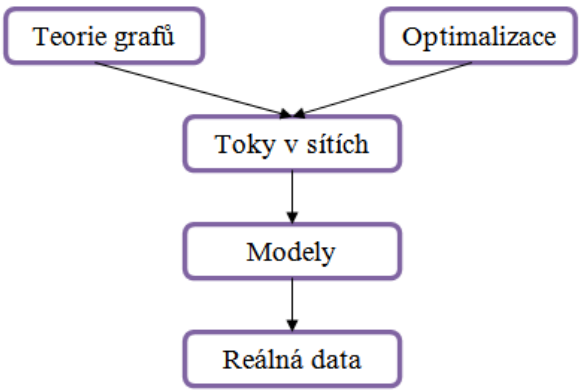
Obsah

Úvod

Ve své práci, Vybrané optimalizační modely v logistice, se zabývám modelováním dopravních úloh. Cílem bakalářské práce je tvorba modelů z oblasti logistiky a realizace výpočtu na reálných datech. Dopravní úlohy jsou zde modifikovány pro svoz komunálního odpadu z důvodu aplikační části, která je věnována modelům s reálnými daty.

Nezbytnou část k modelům tvoří teorie, která je popsána v kapitolách 1, 2 a 3. V těchto kapitolách jsou vysvětleny pojmy z oblasti teorie grafů s důrazem na toky v sítích, dále téma optimalizace a lineárního programování, které nám umožňuje formulovat a matematicky popsat daný problém. 3. kapitola, Toky v sítích, propojuje témata předchozích dvou kapitol. Spíše než definování nových pojmů je zde dopodrobna rozebrán a popsán příklad, na jehož principech se zakládají další modely. Toky v sítích jsou využívány v logistice, kde lze vhodně aplikovat pojmy z teorie grafů a optimalizace. Výsledkem jsou optimalizační modely, ve kterých hledám minimum (např. minimální cenu za dopravu) nebo maximum (např. maximální zisk) účelové funkce.

Kapitoly 4 a 5 tvoří stěžejní, původně zpracovanou část práce, ve které se věnuji jednotlivým modelům. 4. kapitola nese název Modely. Jsou zde popsány testovací sítě, které v modelech využívám. Dále zde rozebírám sedm typů modelů od jednodušších po složitější. V podkapitolách 4.2-4.4 se věnuji hledání minima a modely v částech 4.5-4.8 jsou koncipovány na úlohu maximalizace. V závěru každé části naleznete shrnutí výsledků a případné porovnání s ostatními modely. 5. kapitolu tvoří modely realizované na reálných datech. Konkrétně se jedná o svoz komunálního odpadu pro obce s rozšířenou působností. Naleznete zde například zpětně vymodelovaný tok odpadu nebo komplexnější úlohu využívající principy modelů ze čtvrté části. Závěr je věnován porovnáním kapitol 4 a 5 a možnostem dalšího využití.



Obrázek 1: Schéma bakalářské práce.

1 Teorie grafů

Teorie grafů nám umožňuje efektivně znázornit reálné situace pomocí grafů, kde jsou jednotlivé objekty nahrazeny uzly či vrcholy a vazby mezi nimi jsou interpretovány hranami. Takto vybudovanou teorii lze aplikovat v mnoha oblastech praxe.

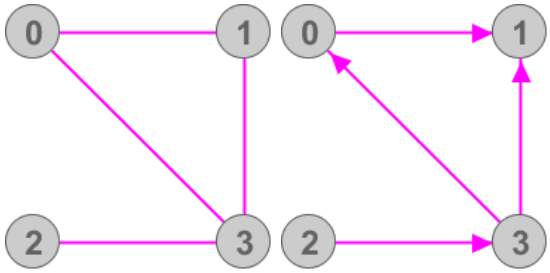
Teorie grafů sahá až do 18. století a *„za jejího zakladatele je považován Leonhard Euler, který v roce 1736 řešil problém sedmi mostů města Královce. Městem protéká řeka Pregola, která zde vytváří dva ostrovy propojené sedmi mosty. Otázka zněla, zda je možné uskutečnit procházku po městě tak, aby se po každém mostu prošlo právě jednou. Euler za pomoci teorie grafů ukázal, že to možné není“* [1].

Ve své práci se budu zabývat dopravou a logistikou, kde budu z teorie grafů využívat zejména toky v sítích. Teorie je pěkně popsána v [1], [2] a [3].

Definice 1.1: *Prostý neorientovaný graf G* je dvojice (V,E), kde V je konečná množina vrcholů grafu G a $E \subseteq P_2(V) \cup V$ je konečná množina hran grafu G. Hrana $e = \{v_1, v_2\}$ propojuje uzly v_1 a v_2. V grafu je znázorněna úsečkou.

Definice 1.2: *Prostý orientovaný graf G* je dvojice (V,E), kde V je konečná množina vrcholů grafu G a $E \subseteq V \times V$ je konečná množina hran grafu G. Hrana $e = (v_1, v_2)$ je orientovaná hrana s počátkem ve vrcholu v_1 a koncem ve v_2. V grafu je znázorněna šipkou.

Poznámka: $V^2 = V \times V = \{(v_1, v_2) \mid v_1, v_2 \in V\}$ značí orientované hrany, příp. orientované smyčky pro $v_1 = v_2$, $P_2(V) = \{\{v_1, v_2\} \mid v_1 \neq v_2, v_1, v_2 \in V\}$ značí neorientované smyčky a $\{\{v\} \mid v \in V\} \approx V$ symbolizuje neorientované smyčky.



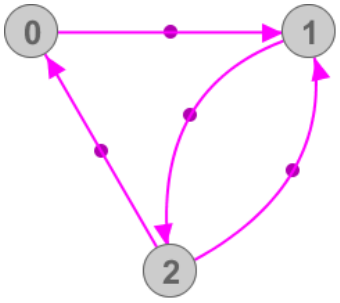
Obrázek 1.1: Neorientovaný a orientovaný graf

Poznámka: Prostý graf může mít nejvýše jednu hranu mezi dvěma vrcholy. Některé případy však vyžadují existenci více hran spojující dva vrcholy (např. silnice v obou směrech, paralelní vedení,...). Tento pojem je označován jako násobnost hrany a pomocí obecné definice grafu můžeme zadefinovat pojem multigraf.

Definice 1.3: *Obecný graf G* je trojice (V,E,ε), kde V je množina vrcholů grafu G a ε je zobrazení incidence, $\varepsilon : E \longrightarrow V^2 \cup P_2(V) \cup V$.

Definice 1.4: *Multigraf* je graf, v němž násobnost alespoň jedné hrany je větší než jedna.

Poznámka: Příkladem výše uvedených grafů jsou testovací sítě ve 4. kapitole.



Obrázek 1.2: Multigraf

DEFINICE 1.5: *Sled* délky k v grafu $G = (U, E, \varepsilon)$ z uzlu v_0 do uzlu v_k je konečná posloupnost tvaru $v_0, e_1, v_1, e_2, v_2, ..., v_{k-1}, e_k, v_k$ kde $k \geq 0$, ve které se vždy střídají uzly a hrany ($v_i \in V, e_i \in E$ pro $\forall i = 0, ..., k$) a pro každé $j = 1, ..., k$ platí, že $\varepsilon(e_j) = (v_{j-1}, v_j)$, tedy že hrana e_j spojuje uzel v_{j-1} s uzlem v_j.

DEFINICE 1.6: *Cesta* v grafu je sled, ve kterém se neopakují žádné uzly.

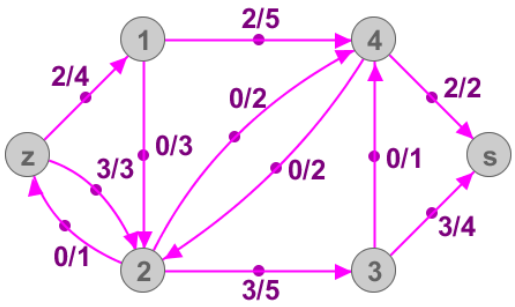
DEFINICE 1.7: *Síť* je čtveřice $S = (G, z, s, w)$, kde G je orientovaný graf, z, s dva různé vrcholy grafu G takové, že do z nevstupují žádné hrany a z vrcholu s žádné hrany nevystupují, a w je funkce $w : E(G) \to \mathbf{R}^+$, kde $E(G)$ značí množinu hran grafu G. Funkce w přiřazuje každé orientované hraně e grafu G kladné reálné číslo *w(e)*.

POZNÁMKA: Vrchol z se nazývá zdroj, vrchol s spotřebič (stok) a číslo *w(e)* kapacita (propustnost) hrany e.

DEFINICE 1.8: *Tok v síti* S = (G,z,s,w) je takové ohodnocení hran grafu reálnými čísly (tj. zobrazení $x : E(G) \to \mathbf{R}$), které splňuje následující podmínky:

- *Kapacitní omezení:* Tok v každé hraně nemůže převýšit její kapacitu a nemůže být záporný, tedy
$$\forall e \in E(G) : 0 \leq x(e) \leq w(e)$$
- *Kirchhoffův zákon zachování toku:* Přítok do každého vnitřního uzlu v je roven odtoku z uzlu v, tedy
$$\forall v \in V(G) - \{z, s\} : \sum_{e \in E_G^-(v)} x(e) = \sum_{e \in E_G^+(v)} x(e)$$

POZNÁMKA: Tok a kapacitu hran v obrázku 4 zjednodušeně zapisujeme ve formátu X/W, kde X je hodnota toku na hraně a W je její kapacita. Následující obrázek názorně ilustruje tok v síti. Velikost toku ze zdroje do stoku je 5. Jedná se o součet dílčích toků směřujících do stoku. Všimněte si, že ilustrovaný obrázek umožňuje i tok o velikosti 6. Nejedná se tedy o případ maximálního toku.



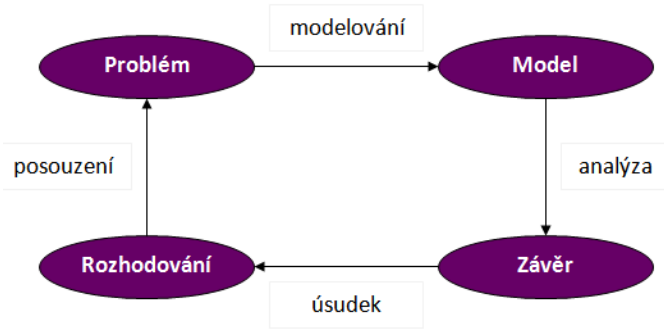
Obrázek 1.3: Ukázka toku v síti [1]

2 Optimalizace

V této kapitole projdu od obecného operačního výzkumu přes samotnou optimalizaci až po konkrétní specifikaci optimalizačních modelů. Teorii jsem čerpal z [4], [5] a [6].

2.1 Základy operačního výzkumu

„Operační výzkum se zabývá rozhodováním pomocí formulace a analýzy matematických modelů“ (Ronald L. Rardin) [4]. Na obrázku 2.1 naleznete názornou ukázku procesu operačního výzkumu.



Obrázek 2.1: Proces výzkumu operací. Převzato z [4].

Poznámka: Jedná se o smyčku, kde se na samotném začátku snažíme formulovat daný problém: Kolik je proměnných? Jaký je potřebný počet vazeb k popsání našeho problému? Další otázky pomáhající při vytváření modelu najdeme např. ve 4. kapitole. Výsledkem procesu modelování je model, jehož analýzou dojdeme k závěru a lepšímu pochopení, co model představuje a jak interpretovat výsledky s jeho pomocí. Zde na řadu přichází úsudek, zda-li model odpovídá našim předpokladům, či nikoliv, a případný návrat na začátek smyčky a korekce vstupních parametrů.

2.2 Optimalizace

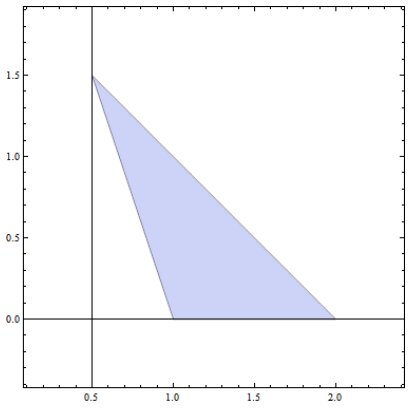
„Optimalizační modely (též zvané matematické programy) představují řešení problému, jak zvolit hodnoty rozhodovacích proměnných při hledání maxima nebo minima účelové funkce vzhledem k omezením zahrnujícím proměnné.“ (Ronald L. Rardin)[4].

Poznámka: Optimalizační modely se dále dělí na *deterministické* a *stochastické modely*. U deterministických optimalizačních modelů známe na rozdíl od stochastických modelů dopředu veškerá data s určitostí. Našimi vstupy jsou tedy konstanty. I když se realitě více přibližují modely stochastické, deterministické modely bývají zpravidla jednodušší a výsledky jejich řešení přesto bývají aplikovatelné.

Poznámka: *Řešení je přípustné*, pokud splňuje všechna omezení. *Optimální řešení* je přípustné řešení dosahující hodnot účelové funkce, které jsou lepší nebo stejné, jako pro každé jiné přípustné řešení.

PŘÍKLAD: Jedná se o příklad ze cvičení ve [4]. Úlohou je najít grafické řešení omezení:

$$x + y \leq 2; 3x + y \geq 3; x, y \geq 0.$$



Obrázek 2.2: Grafické řešení omezení.

Dříve, než uvedu zápis pro tzv. *standardní model*, se podrobněji podívám na jeho složky:

DEFINICE 2.2:

- *Proměnné* v optimalizačních modelech reprezentují rozhodnutí, které má být učiněno.
- *Meze proměnných* specifikují množinu hodnot, pro kterou mají proměnné význam.
  - Např. nezápornost toku: $x_{i,j} \geq 0$.
- *Hlavní omezení* optimalizačních modelů vymezují restrikce a interakce jiné než u mezí proměnných, které obvykle obsahují více než jednu proměnnou.

Účelová funkce nám v optimalizačních modelech říká jak ohodnotit naše rozhodnutí, zda jej minimalizovat či maximalizovat. Nyní máme rozebrané všechny složky potřebné k zavedení *standardního optimalizačního modelu*:

POZNÁMKA:

$$\begin{aligned} &\text{min nebo max} && \text{(účelová funkce)} \\ &\text{vzhledem k} && \text{(hlavní omezení)} \\ & && \text{(meze proměnné).} \end{aligned} \tag{2.1}$$

Po zbytek práce slova *vzhledem k* ponechám v angličtině a slovo *subject to* budu zapisovat zkráceně *s.t.* Matematicky lze slovní formulaci (2.1) přepsat do tvaru:

DEFINICE 2.2: Optimalizačním modelem budeme dále nazývat zápis

$$\begin{aligned} &\min && f(x) \\ &\text{s. t.} && g_i(x) \leq 0 \qquad i = 1, ..., m \\ & && x \in X \subset \mathbf{R}^n, \end{aligned} \tag{2.2}$$

kde $f(x)$ značí účelovou funkci a $g_i : \mathbf{R}^n \longrightarrow \mathbf{R}, i = 1, ..., m$ určují výrazy v omezeních.

Poznámka (vlastnosti modelu): V závislosti na vlastnostech problému definovaného funkcemi f a g_i a množinou X, model (2.2) se nazývá:

- *lineární*, pokud množina X je konvexní polyedrický mnohostěn a funkce $f, g_i, i = 1, ..., m$ jsou lineární.
- *nelineární*, pokud alespoň jedna z funkcí $f, g_i, i = 1, ..., m$, je nelineární nebo množina X není polyedrický mnohostěn. Nelineární program se dále dělí na:
  - *konvexní*, pokud $X \cap \{x \mid g_i(x) \leq 0, i = 1, ..., m\}$ (množina přípustných řešení) je konvexní a účelová funkce f je konvexní funkcí.
  - *nekonvexní*, pokud alespoň jedna z množiny X nebo účelová funkce f není konvexní.

Poznámka: Pokud je program konvexní, pak lokální minimum je zároveň i minimem globálním.

2.3 Lineární programování

Definice 2.4: Optimalizační model je v *lineární formě*, pokud má spojité proměnné, jednu lineární účelovou funkci, a všechna omezení jsou lineární. Lineární optimalizační model může být zapsán následovně:

$$\begin{aligned} \min \quad & \{c_1x_1 + c_2x_2 + ... + c_nx_n\} \\ \text{s. t.} \quad & m_{11}x_1 + m_{12}x_2 + ... + m_{1n}x_n = t_1 \\ & m_{21}x_1 + m_{22}x_2 + ... + m_{2n}x_n = t_2 \\ & \vdots \\ & m_{m1}x_1 + m_{m2}x_2 + ... + m_{mn}x_n = t_m \\ & x_1, x_2, ..., x_n \geq 0. \end{aligned} \tag{2.3}$$

Poznámka: Místo rovnic zde mohou být i nerovnice. Použitím maticového-vektorového zápisu lze formulaci (2.3) přepsat do tvaru:

$$\begin{aligned} \min \quad & \mathbf{c}^T\mathbf{x} \\ \text{s. t.} \quad & \mathbf{M}\mathbf{x} = \mathbf{t} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{2.4}$$

kde jednotlivé složky znamenají: $\mathbf{c}^T \in \mathbf{R}^n$ představuje řádový cenový faktor, $\mathbf{x} \in \mathbf{R}^n$ je vektorem hledaných proměnných (pro nás představuje tok), $\mathbf{M}$ je matice typu $m \times n$, $\mathbf{t} \in \mathbf{R}^m$ je vektorem požadavků a kapacit. Skalární součin $\mathbf{c}^\top\mathbf{x}$ je roven závisle proměnné z, jejíž optimální hodnota je hledaným výsledkem. Dále proto v modelech budu účelovou funkci zapisovat i se závisle proměnnou ve tvaru $z = \mathbf{c}^\top\mathbf{x}$.

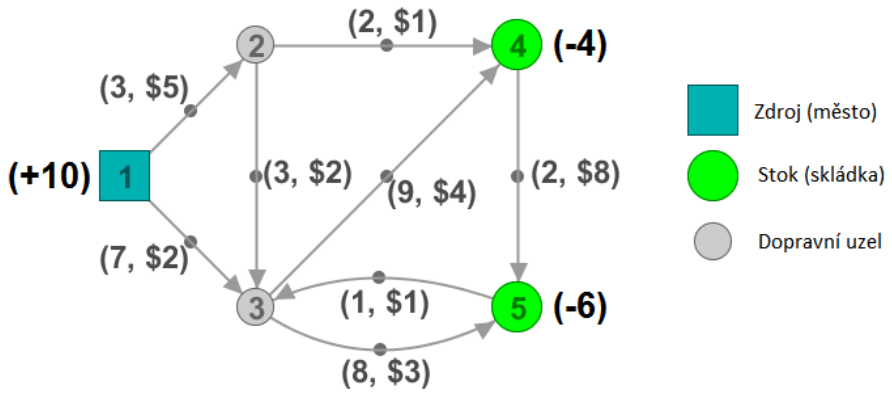
3 Toky v sítích

V kapitole se snažím provázat dosavadní teorii s dopravními modely, které budou aplikovány v páté kapitole. Zároveň popíšu pravidla dopravního toku, kterými se budu řídit, a podrobně zde rozeberu jednoduchý příklad. Teorii jsem čerpal ze [7] a [8].

Poznámka: Obecný dopravní problém lze formulovat jako úlohu minimalizace ceny za přepravu. Z hlediska optimalizace má lineární program tvar:

$$\begin{aligned} \min \quad & z = \mathbf{c}^T\mathbf{x} \\ \text{s. t.} \quad & \mathbf{M}\mathbf{x} = \mathbf{t} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{3.1}$$

kde vektor $\boldsymbol{l}$ zahrnuje dolní a vektor $\boldsymbol{u}$ horní meze proměnných složek vektoru $\mathbf{x}$. V příští kapitole se budu zabývat svozem komunálního odpadu, kde je odpad svážen z měst do skládek. Města, skládky a případné dopravní uzly (křižovatky,...) jsou obecně označovány jako *uzly*. Cesta spojující dva uzly představuje *hranu*. Tok z uzlu i do j označím pomocí indexů jako $x_{i,j}$, jehož konkrétní hodnota udává počet převezených jednotek přes tuto hranu. Vykreslením všech uzlů a tokem mezi nimi dostanu orientovaný graf. Viz obrázek 3.1.



Obrázek 3.1: Dopravní síť. Námět k ilustraci čerpán z [7].

Ilustrace 3.1 se skládá z 5 uzlů a 8 orientovaných hran. Modrý čtverec, interpretován jako město, reprezentuje zdroj, který se vyznačuje kladnou hodnotou („vytváří“ odpad). Náš má hodnotu $(+10)$. Zelené kruhy představují stok (pro nás skládky) a mají záporné hodnoty (-4) a (-6), protože odebírají daný počet jednotek odpadu. Orientace hran určuje směr možného toku a čísla $(3, \$5)$ jsou omezeními hran, kde první hodnota udává možný tok a druhá cenu za jednotku při použití dané hrany.

V obecné dopravní úloze zde bude proměnná pro každou hranu v síti a pro každý uzel bude určeno omezení. Předpokládejme, že máme m uzlů a n hran, pak matice $\mathbf{M}$ je typu $m \times n$. Řádek i omezení matice $\mathbf{M}$ koresponduje s i-tým uzlem. Řídí se následujícím pravidlem:
Pravidlo:

$$\text{(Přítok do } i\text{-tého uzlu) - (odtok z } i\text{-tého uzlu) = (kapacita uzlu } i\text{).} \tag{3.2}$$

Poznámka: Pravidlo můžeme najít v publikaci [7], kde je uvedeno s přehozenými znaménky. Ve své práci se však budu držet kladného znaménka pro přítok a záporným znaménkem budu označovat výtok. Ve výpočetních programech se zápisy provádí algebraicky za použití sum a indexů. Pravidlo (3.2) lze přepsat do algebraického tvaru:

$$\sum_{j=1}^{m} x_{i,j} - \sum_{k=1}^{m} x_{k,i} = t_i.$$

Pokud je výsledné $t_i > 0$, jedná se o zdroj. Když $t_i < 0$, pak i-tý uzel představuje stok a při $t_i = 0$ se jedná o dopravní uzel.

Příklad: Jako příklad rozepíši rovnice pro uzel 3 a 4 viz ilustrace 3.1:

$$\begin{array}{lll} 3: & -x_{1,3} - x_{2,3} + x_{3,4} - x_{5,3} + x_{3,5} & = 0 \\ 4: & -x_{2,4} - x_{3,4} + x_{4,5} & = -6. \end{array}$$

Z výše uvedených pravidel je patrné, že 3. uzel je uzlem dopravním a 4. uzel stokem o spotřebě 4 jednotek. Všechny rovnosti mohou být zapsány vektorově:

$$\mathbf{Mx} = \mathbf{t},$$

kde, jak již bylo zmíněno v předchozí kapitole, $\mathbf{M}$ je matice incidence. Skládá se pouze z $-1, 0$ a 1. Každý sloupec značí existující hranu i-j a na i-tém řádku jsou vypsány všechny hrany směřující z a do i-tého uzlu. Pro náš případ matice incidence vypadá následovně:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & & & & & & & \\ -1 & & 1 & 1 & & & & & \\ & -1 & -1 & & 1 & 1 & -1 & & \\ & & & -1 & -1 & & & 1 & \\ & & & & & -1 & 1 & -1 & \end{pmatrix}.$$

Vektor pravých stran je roven: $\mathbf{t} = (10, 0, 0, -4, -6)^\top$. Veškeré informace o daném problému lze zapsat do přehledné tabulky:

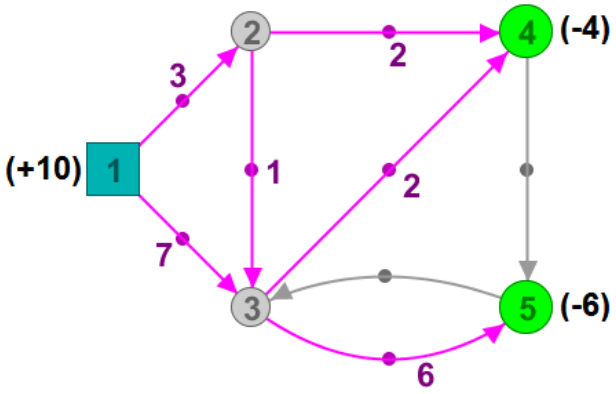
	$x_{1,2}$	$x_{1,3}$	$x_{2,3}$	$x_{2,4}$	$x_{3,4}$	$x_{3,5}$	$x_{5,3}$	$x_{4,5}$	**Vektor pravých stran** t_i
Uzel 1	1	1							10
Uzel 2	-1		1	1					0
Uzel 3		-1	-1		1	1	-1		0
Uzel 4				-1	-1			1	-4
Uzel 5						-1	1	-1	-6
Kap. hran $w_{i,j}$	3	7	3	2	9	8	1	2	
Úč. funkce $c_{i,j}$	5	2	2	1	4	3	1	8	(min)

Poznámka: Kapacita hran udává horní limit jednotlivým hranám, např. $0 < x_{1,2} < 3$. Princip zápisu do tabulky je využit v MS Excel, ze kterého jsou takto seřazená data nahrána do programů kapitol 4 a 5 pro modely psané v prostředí AIMMS. Více o programech, které jsem ve své práci využil naleznete v příloze C a D.

Model (3.1) může být modifikován do tvaru, který budu využívat v nadcházejících kapitolách:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{i,j} x_{i,j}, \\ \text{s. t.} \quad & \sum_{j=1}^{m} x_{i,j} - \sum_{k=1}^{m} x_{k,i} = t_i \quad i = 1, ..., n, \\ & l_{i,j} \leq x_{i,j} \leq u_{i,j}. \end{aligned} \tag{3.3}$$

Příklad (pokračování): Řešením našeho příkladu je: $z = \$54$, kde z představuje minimální cenu za dopravu. Výsledný tok je ilustrovaný na obrázku 3.2.



Obrázek 3.2: Výsledný tok dopravní sítí.

4 Modely

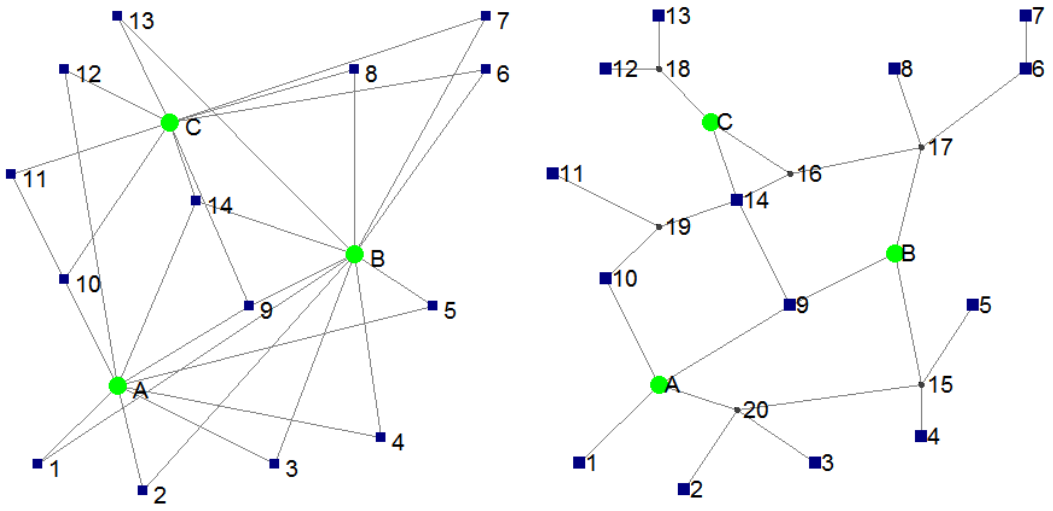
Předešlé kapitoly uvedly základní teoretické poznatky, tato kapitola je zaměřena na jejich aplikaci v logistice. Kapitola popisuje jednotlivé modely, od nejjednodušších po obtížnější. Stále se však jedná o základní „idealizované“ modely, které se od reálné situace liší. Úvodem do logistiky může být např. [10].

Ve svých modelech se zabývám svozem komunálního odpadu a na tento problém se snažím nahlížet vždy z jiného úhlu pohledu - Zda hledat minimum či maximum? Jaké a kolik je hledaných parametrů? Na závěr práce aplikuji některé modely na reálná data, která mi poskytl Ústav procesního inženýrství.

Jednotlivé modely jsou vytvořeny v prostředí programu GAMS a pro vizualizaci jsem některé z nich naprogramoval i v programu AIMMS. Oba programy slouží k optimalizačním úlohám. Inspiraci pro námět s dopravními uzly a zejména pak pro část 5.8: Dynamická cena jsem čerpal z [9]. Nejprve uvedu vytvořenou dopravní síť, kterou budu v modelech užívat k testování.

4.1 Testovací síť

Testovací síť má dvě podoby - *s* a *bez* dopravních uzlů. Na dopravní uzel můžeme nahlížet jako na křižovatku. Síť se skládá z uzlů, které jsou provázány hranami. Každý uzel je určen pomocí souřadnic $[X_{coor}; Y_{coor}]$, kde prvních 14 uzlů tj. s indexy od $(1, .., 14)$ představuje města. Uzly $15, .., 20$ jsou dopravními uzly a uzly označené A, B, C značí skládky.



Obrázek 4.1: Dopravní sítě. Vlevo bez dopravních uzlů a vpravo s dopravními uzly.

Síť bez dopravních uzlů vypadá komplikovaněji, ale opak je pravdou. Hrany zde vedou z města rovnou do skládek a z každého města vedou alespoň dvě hrany.

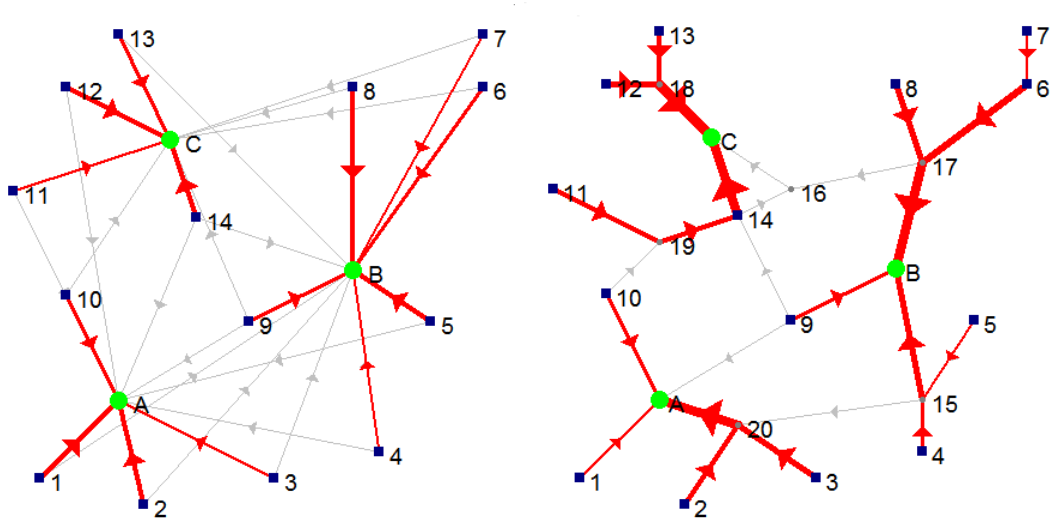
Naopak síť s dopravními uzly je přehlednější, protože dopravní uzly mohou být interpretovány jako křižovatky či překladní místa.

4.2 Minimalizace ceny za přepravu

První, nejjednodušší model se bude zabývat otázkou minimalizace ceny nákladů za přepravu. Tedy hledáme minimální částku, kterou zaplatí dopravce za cestu. Cena je závislá na vzdálenosti města od skládky, ze které se počítá cena za přepravu po dané hraně. Model vypadá následovně:

$$\begin{aligned} \min \quad & z = \textstyle\sum_i \sum_j c_{i,j} \cdot x_{i,j}, & & \\ \text{s. t.} \quad & \textstyle\sum_j M_{i,j} \cdot x_{i,j} \geq t_i, & & i = 1, ..., 14, \\ & x_{i,j} \geq 0, & & i = 1, ..., 14, j = A, B, C. \end{aligned} \tag{4.1}$$

Model se skládá z účelové funkce a jedné skupiny omezujících podmínek, která určuje odvoz veškerého odpadu z měst. Hledanou minimální částkou je proměnná z. Druhou neznámou je tok $\mathbf{x}_{i,j}$ udávající počet přepravených jednotek mezi uzly. Parametr $\mathbf{c}_{i,j}$ představuje cenu za přepravu jednotky z uzlu i do j. Parametr $\mathbf{t}_i$ určuje množství odpadu ve městech. Posledním parametrem je $\mathbf{M}_{i,j}$. Jedná se o incidenční matici, která nám vymezuje možné hrany. Kapacita hran zde není uvedena. Můžeme ji brát jako neomezenou, tj. $+\infty$.



Obrázek 4.2: Výsledné dopravní sítě. Vlevo bez a vpravo s dopravními uzly.

Z ilustrace 4.2 je patrné „roztrhání“ grafu na podgrafy, kde v každém z nich je jedna skládka. Jedná se o výslednou vizualizaci vytvořenou v programu AIMMS, kde jednotlivé hrany jsou orientovány a tloušťka hrany je úměrná toku, který přes ní protéká. Šedivé orientované hrany představují nevyužité hrany.

Popsaný model (4.1) koresponduje s dopravní sítí bez dopravních uzlů (viz. strana 40 - Seznam zkratek). Model si můžete prohlédnout na přiloženém CD pod názvem `Minimizing_total_cost01` jak v programu GAMS, tak i AIMMS. Úloha s dopravními uzly se liší od (4.1). Zásadní změna je v nahlížení na hrany. Na rozdíl od (4.1), kde byl tok $\mathbf{x}_{i,j}$ zapisován pomocí dvou indexů i a j, zde přímo uvažujeme tok přes hranu e, tj. $\mathbf{x}_e$. Incidenční matice nám pak určí orientaci dané hrany. Model je zapsán následovně:

$$
\begin{array}{lll}
\min & z = \sum_e c_e \cdot x_e, & \\
\text{s. t.} & \sum_e -M_{Icit,e} \cdot x_e \geq t_{Icit}, & Icit = 1, ..., 14, \\
& \sum_e M_{Itra,e} \cdot x_e = 0, & Itra = 15, ..., 20, \\
& \sum_e M_{Ilan,e} \cdot x_e \leq t_{Ilan} & Ilan = A, B, C, \\
& x_e \geq 0, & e = 1, ..., 26.
\end{array} \tag{4.2}
$$

Index i určující uzly je zde pro názornost doplněn pomocnými indexy *Icit* (pro města), *Ilan* (pro skládky) a *Itra* (pro dopravní uzly). První omezující podmínka je totožná s podmínkou z modelu (4.1). Znaménko minus je zde z důvodu zavedeného pravidla (3.2) o toku přes uzel. Druhá podmínka se vztahuje k dopravním uzlům a říká, že přítok je roven odtoku, tj. výsledný tok přes dopravní uzel je roven nule. Třetí podmínka je vztažena pro skládky. Omezuje množství odpadu dovezeného do skládek, kde parametr $\mathbf{t}_{Ilan}$ je jejich kapacitou. V praxi se však skládky staví tak, aby vydržely několika roční provoz, který může být až v řádu desítek let. Jejich kapacita je mnohem větší než množství sváženého odpadu v modelech, a proto tuto podmínku můžeme zanedbat. Skládky ovšem můžeme omezit i jiným způsobem. Například v modelu (4.3) jsou skládky omezeny procentuálně.

Výsledkem v GAMSu je textový výčet. Pro model (4.2) je zde zobrazen tok $\mathbf{x}_e$ a celková cena za dopravu z. Jednotlivé hrany jsou označeny dvojicí, např. hrana *1-A* vede z města *1* do skládky *A* a v tomto směru je orientována.

```
----     98 VARIABLE x.L  variable of quantity transported through a edge e

1-A   14.000,    2-20  23.000,    20-A  49.000,    3-20  26.000,    4-15  16.000
5-15  12.000,    15-B  28.000,    9-B   15.000,    10-A  16.000,    11-19 22.000
14-C  41.000,    17-B  61.000,    8-17  28.000,    6-17  33.000,    7-6   13.000
18-C  48.000,    12-18 25.000,    13-18 23.000,    19-14 22.000


----     98 VARIABLE z.L                  =     2342.000  total transportation
                                                          costs
```

Model (4.2) si rovněž můžete prohlédnout na CD pod názvem: `Minimizing_total_cost05`. Čísla v názvech modelů korespondují s pořadím, ve kterém byly řešeny.

4.3 Kapacitní omezení

U druhého modelu je otázka stejná jako u modelu předchozího - hledáme minimální částku za dopravu. Zahrneme zde však kapacitní omezení dvojího typu: omezení toku přes hrany, tj. stanovení maximálního toku, a omezíme skládky. Vlivem omezení očekáváme nárůst celkové ceny a použití více hran. Doplňující omezení se pro model bez dopravních uzlů od modelu s dopravními uzly výrazně neliší, a proto rozepíši pouze druhý z nich.

Model (4.2) se modifikuje na:

$$\begin{array}{lcll} \min & z = \sum_e c_e \cdot x_e, & & \\ \text{s. t.} & \sum_e -M_{Icit,e} \cdot x_e \geq t_{Icit}, & Icit = 1, ...14, & \\ & \sum_e M_{Itra,e} \cdot x_e = 0, & Itra = 15, ..., 20, & (4.3) \\ & x_e \leq w_e, & e = 1, ..., 26, & \\ & \sum_e x_e \cdot M_{Ilan,e} \leq f_{Ilan} \cdot \sum_{Icit} t_{Icit}, & Ilan = A, B, C, & \\ & x_e \geq 0, & e = 1, ..., 26. & \end{array}$$

Červené části jsou nové. První nerovnost z nich omezuje tok $\mathbf{x}_e$ přidáním horní meze $\mathbf{w}_e$. Druhá slouží jako kapacitní omezení skládek, kde parametr $\mathbf{f}_{Ilan} \in \langle 0; 1 \rangle$ procentuálně omezuje množství dovezeného odpadu na skládky. Suma $\sum_{Icit} t_{Icit}$ představuje celkové množství odpadu. Jejich vynásobením získáme procentuální množství celkového odpadu odváženého na jednotlivé skládky. Nerovnost je v programu pro skládky rozepsána jako:

$$\sum_e x_e \cdot M_{A,e} \leq 0.3 \cdot \sum_{Icit} t_{Icit},$$

$$\sum_e x_e \cdot M_{B,e} \leq 0.4 \cdot \sum_{Icit} t_{Icit},$$

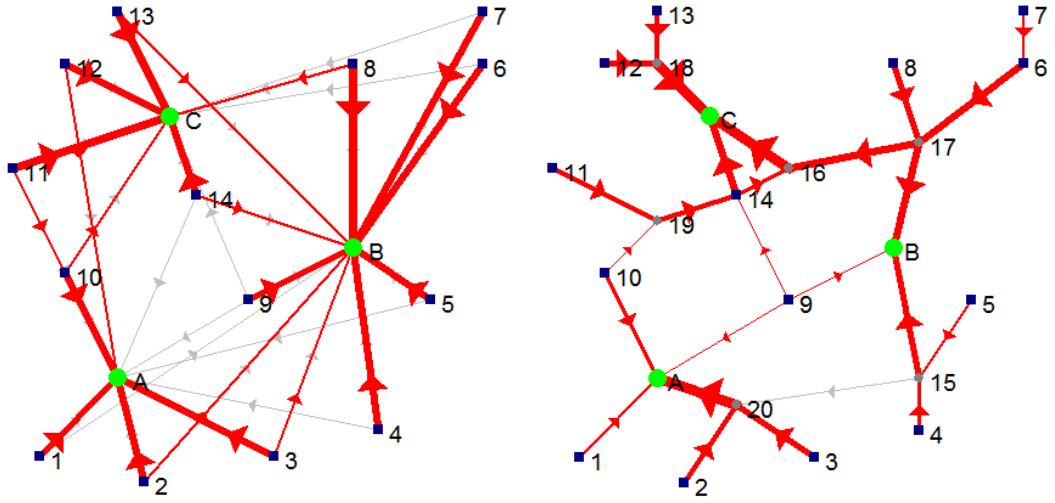
$$\sum_e x_e \cdot M_{C,e} \leq 0.5 \cdot \sum_{Icit} t_{Icit}.$$

Tedy můžeme říci, že se do skládky A odveze nejvýše třicet procent z celkového odpadu. Řešení modelu (4.3):

```
----    109 VARIABLE x.L  variable of quantity transported through a edge e

1-A   14.000,    2-20  23.000,    20-A  49.000,    3-20  26.000,    4-15  16.000
5-15  12.000,    15-B  28.000,    9-B    8.000,    9-A    3.600,    10-A  15.000
9-14   3.400,    10-19  1.000,    11-19 22.000,    14-C  30.000,    14-16 15.400
17-B  30.000,    17-16 31.000,    16-C  46.400,    8-17  28.000,    6-17  33.000
7-6   13.000,    18-C  48.000,    12-18 25.000,    13-18 23.000,    19-14 23.000


----    109 VARIABLE z.L                  =     2827.600  total transportation
                                                          costs
```

Celková cena za dopravu činí 2827.6. V porovnání s modelem (4.2) je doprava dražší. Z níže přiloženého výpisu o dovezeném odpadu na skládky vidíme, kolik odpadu se na skládky odvezlo. Na skládku A se dovezlo maximální možné množství odpadu.

```
PARAMETER tot                =      272.000  total amount of waste
PARAMETER totA               =       81.600  total amount of waste
                                             shipping to landfill
                                             A
PARAMETER totB               =       66.000  total amount of waste
                                             shipping to landfill
                                             B
PARAMETER totC               =      124.400  total amount of waste
                                             shipping to landfill
                                             C
```



Obrázek 4.3: Výsledné dopravní sítě. Vlevo bez a vpravo s dopravními uzly.

Z grafického výstupu je na první pohled patrné využití více hran. U modelu (4.3) nebyla použita pouze hrana *15-20*.

Můžeme tedy říci, že přidáním kapacitního omezení se výrazně změní výsledný graf, cena za dopravu se zvedne a omezením hran přinutíme program jednotlivé množství odpadu rozdělit.

Oba modely jsou k dispozici na přiloženém CD pod názvy: `Minimizing_total_cost04` a `Minimizing_total_cost06`.

4.4 Hledání optimálního rozložení skládek

Cílem úlohy je najít optimální souřadnice skládek při minimalizaci ceny za dopravu. Předpokládáme dosažení lepšího výsledku než u modelu (4.1).

Neznámými zde budou souřadnice skládek $\mathbf{n2}_{j,k}$, kde index k představuje X a Y složky souřadnic, dále pak vzdálenost $\mathbf{d}_{i,j}$ měst i od skládek j, počet převezených jednotek odpadu $\mathbf{x}_{i,j}$ a celková cena za dopravu z.

Model bez dopravních uzlů (4.1) se rozšíří do tvaru:

$$\begin{aligned}
\min \quad & z = \textstyle\sum_i \sum_j c_{i,j} \cdot x_{i,j}, & \\
\text{s. t.} \quad & \textstyle\sum_j M_{i,j} \cdot x_{i,j} \geq t_i, & i = 1, ..., 14, \\
& d_{i,j} = \sqrt{(\textstyle\sum_k (n2_{j,k} - n1_{i,k})^2}, & i = 1, ..., 14, j = A, B, C, \\
& n2_{j,k} \geq 0, & i = 1, ..., 14, j = A, B, C, \\
& n2_{j,k} \leq 100, & i = 1, ..., 14, j = A, B, C, \\
& d_{i,j} \geq 0, & i = 1, ..., 14, j = A, B, C, \\
& x_{i,j} \geq 0, & i = 1, ..., 14, j = A, B, C.
\end{aligned} \tag{4.4}$$

Vzdálenost $\mathbf{d}_{i,j}$ měříme pomocí euklidovské metriky. Parametrem $\mathbf{n1}_{i,k}$ rozumíme známé souřadnice měst. Na rozdíl od předchozích modelů není model (4.4) konvexní, a tedy podle poznatků kapitoly 2 je nelineární. Řešič pro nelineární programování vybere startovací bod, který opakovaně volí náhodně, a z něj dokonverguje k nejbližšímu lokálnímu optimu. Samotná účelová funkce však může mít lokálních optim více. Tímto při každém spuštění programu můžeme dostat různá lokální optima, pokud existují. Proto jsem model (4.4) při implementaci rozšířil o jeho cyklické n-násobné spuštění. Jednotlivé výsledky jsou zapsány do textového souboru, ze kterého jsem vybral nejlepší optimální řešení.

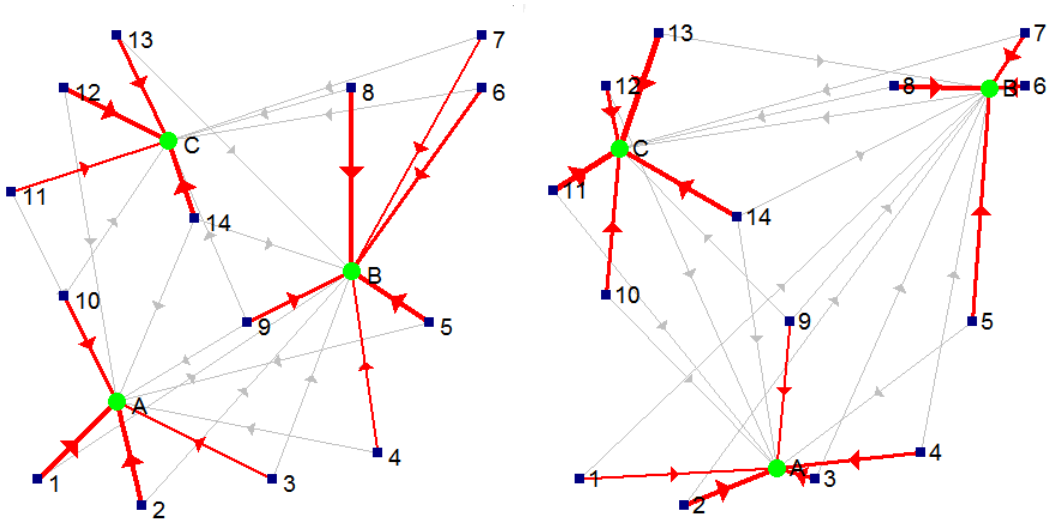
Tabulka porovnává jednotlivá řešení pro modely (4.1), (4.4) a jejich modifikace na modely, ve kterých je každé město spojeno hranou se všemi skládkami, v tabulce označeny připsáním v.h. (volné hrany).

Modely	Celkové náklady za odvoz odpadu
Model 4.1:	z = 784 063,7850
Model 4.1, v.h.:	z = 689 122,2450
Model 4.4:	z = 571 453,7789
Model 4.4, v.h.:	z = 524 369,0920

Z tabulky je na první pohled patrné, že jakékoli omezení se projeví na vyšší ceně za dopravu. Naopak, když programu dáme větší volnost, lze najít i nižší náklady. Druhá tabulka porovnává rozmístění skládek s testovací sítí.

	Souřadnice testovací sítě		Souřadnice modelu 4.4	
	X	Y	X	Y
A	25,00	25,00	53,35	12,83
B	70,00	50,00	70,00	85,00
C	35,00	75,00	13,36	64,19

A následná vizualizace ukazuje, že se skládky přemístily blíže k městům, která mají větší množství odpadu. V modelu (4.4) se generuje odpad v rozmezí $t_i \in \langle 50; 100 \rangle$. Kdyby jedno z měst výrazně převyšovalo ostatní množstvím odpadu, pak by skládka „splývala“ s daným městem, viz. model (5.3).
Model (4.4) si můžete prohlédnout na CD pod názvem:
`Minimizing_total_cost_location01_scenarios`.



Obrázek 4.4: Vlevo model (4.1), vpravo výsledná síť pro model (4.4).

4.5 Maximalizace zisku

V předešlých modelech jsem se zabýval minimalizací účelové funkce, na které jsem hledal minimální cenu za odvoz odpadu. V této i následujících kapitolách budu zkoumat druhý extrém. Jinými slovy budu hledat maximum účelové funkce, kterou v modelech interpretujeme jako hledání maximálního zisku. Model s dopravními uzly (4.2) se rozšíří do tvaru:

$$\begin{aligned} \max \quad & z = \textcolor{red}{\textstyle\sum_{Icit} \sum_e (-M_{Icit,e} \cdot x_e) \cdot g} - \textstyle\sum_e c_e x_e, & & \\ \text{s. t.} \quad & \textstyle\sum_e -M_{Icit,e} \cdot x_e \leq t_{Icit}, & Icit &= 1, ..., 14, \\ & \textstyle\sum_e M_{Itra,e} \cdot x_e = 0, & Itra &= 15, ..., 20, \\ & x_e \geq 0, & e &= 1, ..., 26. \end{aligned} \tag{4.5}$$

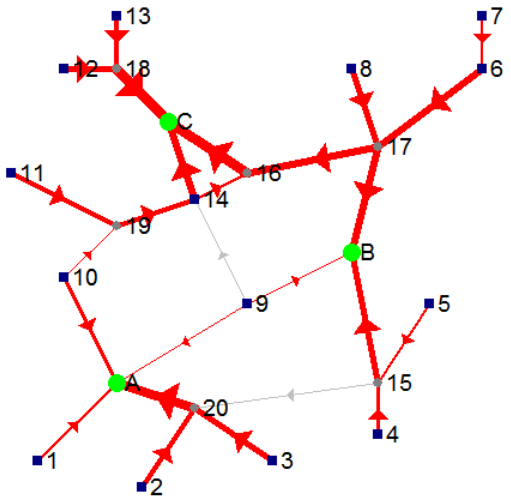
Účelová funkce modelu (4.5) se dá rozdělit na dvě části. První, červeně vyznačená, představuje čistý celkový zisk, od kterého se odečítá cena za dopravu. Úloha maximalizace tedy v sobě zahrnuje i původní účelovou funkci. Skalár g představuje fixní částku, kterou město zaplatí za svoz jedné jednotky odpadu a výraz $\sum_{Icit} \sum_e (-M_{Icit,e} \cdot x_e)$ je interpretován jako výsledné množství odvezeného odpadu z měst.

```
VARIABLE z.L                    =    10820.000  maximal profit
PARAMETER tot_profit            =    13600.000  pure profit
PARAMETER tot_shipping          =     2780.000  total transportation
                                                 costs
```

Obrázek 4.5: Maximální zisk a jeho složky: čistý zisk a náklady za dopravu

```
110 VARIABLE x.L  variable of quantity transported through a edge e

1-A   14.000,    2-20  23.000,    20-A  49.000,    3-20  26.000,    4-15  16.000
5-15  12.000,    15-B  28.000,    9-B    8.000,    9-A    7.000,    10-A  15.000
10-19  1.000,    11-19 22.000,    14-C  30.000,    14-16 12.000,    17-B  30.000
17-16 31.000,    16-C  43.000,    8-17  28.000,    6-17  33.000,    7-6   13.000
18-C  48.000,    12-18 25.000,    13-18 23.000,    19-14 23.000
```

Model (4.5) si můžete prohlédnout na CD pod názvem: `Maximizing_of_profit06_tra`.



Obrázek 4.6: Výsledná síť pro model (4.5).

4.6 Sankce při maximalizaci zisku

V předešlých modelech se svážel veškerý odpad z měst. Nyní přejdeme do situace, kdy tomu tak být nemusí. Dáme programu možnost výběru, zda-li sveze vše, nebo se mu vyplatí odvézt pouze část a za zbytek zaplatit penále. Model s dopravními uzly (4.5) se modifikuje na:

$$
\begin{array}{lcll}
\max & z = \sum_{Icit} \sum_e (-M_{Icit,e} \cdot x_e) \cdot g - \sum_e c_e x_e - \sum_{Icit} p_{Icit} \cdot q, & & \\
\text{s. t.} & \sum_e -M_{Icit,e} \cdot x_e \leq t_{Icit}, & Icit = 1, ..., 14, & \\
& \sum_e M_{Itra,e} \cdot x_e = 0, & Itra = 15, ..., 20, & \\
& t_{Icit} - \sum_e (-M_{Icit,e} \cdot x_e) = p_{Icit}, & Icit = 1, ..., 14, & (4.6) \\
& x_e \leq w_e, & e = 1, ..., 26, & \\
& p_{Icit} \geq 0, & e = 1, ..., 26, & \\
& x_e \geq 0, & e = 1, ..., 26. &
\end{array}
$$

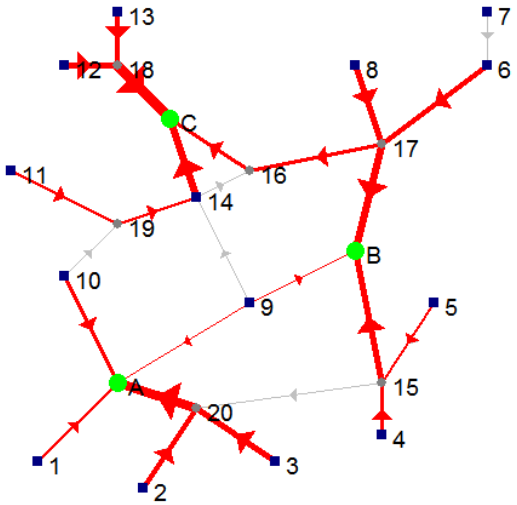
Počet neodvezených jednotek odpadu zde počítáme jako rozdíl množství odpadu ve městě s odvezeným množstvím, který je roven proměnné $\mathbf{p}_{Icit}$. Proměnná je následně násobena konstantní částkou q.

```
        106 VARIABLE x.L  quantity of trash transported through a edge e

1-A   14.000,    2-20  23.000,    20-A  49.000,    3-20  26.000,    4-15  16.000
5-15  12.000,    15-B  28.000,    9-B    8.000,    9-A    7.000,    10-A  15.000
11-19 11.000,    14-C  30.000,    17-B  30.000,    17-16 18.000,    16-C  18.000
8-17  28.000,    6-17  20.000,    18-C  48.000,    12-18 25.000,    13-18 23.000
19-14 11.000
```

Z výpisu je patrné, že ve městech 7, 10 a 11 se neodvezl veškerý odpad. Hledaná maximální částka z je záporná vzhledem k nastavení jednotlivých parametrů v modelu (4.6). Stávající případ by se dal chápat jako nevhodný či prodělečný pro firmu zabývající se svozem odpadu.

```
        106 VARIABLE penalty.L

7  13.000,    10  1.000,    11 11.000


        106 VARIABLE z.L                  =    -7774.000  maximal profit
            PARAMETER tot_profit          =    12350.000  pure profit
            PARAMETER tot_shipping        =    17624.000  total transportation
                                                          costs
            PARAMETER tot_penalty         =     2500.000  penalty
```

K sankcím či penalizacím dojde v momentě, kdy se vyrovná částka za penalizaci s částkou za zisk pokrácený cenou za odvoz, neboli když je cena za odvoz vyšší, naopak zisk z odvozu nižší a případné penále není vysoké.



Obrázek 4.7: Výsledná síť pro model (4.6).

Aby k sankcím došlo, musel jsem navýšit cenu za dopravu $\mathbf{c}_e$, která se původně pohybovala v rozmezí $\langle 1; 10\rangle$ na $\mathbf{c}_e \in \langle 30; 50\rangle$. Pokud by k navýšení ceny nedošlo, výsledky modelu by byly stejné jako v případě modelu (4.5). Ostatní parametry zůstávají stejné jako u minulé úlohy. V ilustraci 4.7 je vidět, že u města č. 7 se nevyplatí za daných podmínek svážet odpad a je lepší zaplatit pokutu.

Model (4.6) je přiložen na CD pod názvem `Maximizing_of_profit06_tra_penalty`. V příloze přikládám i model bez dopravních uzlů, který rovněž řeší sankce a je zde vidět, že i se sankcemi lze dosáhnout kladného výsledku. Model naleznete pod názvem: `Maximizing_profit04_penalty`.

4.7 Dělený odpad

V této části navážeme na předchozí model (4.6), který se zabýval penalizací za neodvezení odpadu. Doposud jsme nahlíželi na odpad jako na celek. V reálném světě se však odpad dále dělí na odpad, který putuje na skládky, do spaloven, recyklujeme jej apod. Cílem modelu je popsat dopravu dvou odlišných druhů odpadů, které jsou zároveň sváženy na různá místa. Reálně si pod modelovanou situací můžeme představit běžný odpad odvážený na skládky a na odpad, který putuje do spaloven. Nejdříve uvedu vstupní parametry:

```
            PARAMETER c  transportation cost

1-A    2.000,    2-20   9.000,    20-A   6.000,    3-20   4.000,    15-20  3.000
4-15   3.000,    5-15   4.000,    15-B   9.000,    9-B    1.000,    9-A    6.000
10-A  10.000,    9-14   6.000,    10-19 10.000,    11-19  8.000,    14-C   2.000
14-16  7.000,    17-B   2.000,    17-16  3.000,    16-C   7.000,    8-17   5.000
6-17   4.000,    7-6    4.000,    18-C   2.000,    12-18  2.000,    13-18  6.000
19-14  9.000,    16-D   3.000,    9-D    7.000,    B-D    8.000,    C-16   4.000
A-9    2.000
            PARAMETER t1  quantity of trash1

1  20.000,    2  13.000,    3  28.000,    4  15.000,    5  16.000,    6  22.000
7  25.000,    8  23.000,    9  19.000,    10 18.000,    11 12.000,    12 16.000
13 10.000,    14 17.000
            PARAMETER t2  quantity of trash1

1  6.000,    2  8.000,    3  8.000,    4  9.000,    5  6.000,    6  8.000
7  9.000,    8  8.000,    9  6.000,    10 5.000,    11 5.000,    12 8.000
13 8.000,    14 5.000
            PARAMETER w  upper bound for edges

1-A   30.000,    2-20  30.000,    20-A  80.000,    3-20  30.000,    15-20 20.000
4-15  30.000,    5-15  30.000,    15-B  40.000,    9-B    8.000,    9-A    8.000
10-A  15.000,    9-14  14.000,    10-19 15.000,    11-19 30.000,    14-C  30.000
14-16 30.000,    17-B  30.000,    17-16 60.000,    16-C  60.000,    8-17  30.000
6-17  60.000,    7-6   30.000,    18-C  60.000,    12-18 30.000,    13-18 30.000
19-14 30.000,    16-D  30.000,    9-D   30.000,    B-D   30.000,    C-16  30.000
A-9   30.000
```

Testovací síť (4.1) se rozšířila o jednu spalovnu, označenou fialově, a byly přidány hrany, aby se modelu umožnilo najít proveditelné řešení. Vznikly tak hrany propustné v obou směrech, tzv. násobné hrany. Jedná se tedy o multigraf. Index $Ilan$ dříve označující skládky se rozdělil na $Ilan$ pro skládky a $Ispa$ pro spalovnu. V městech se generují dva typy odpadů $\mathbf{t}_{1Icit}$ a $\mathbf{t}_{2Icit}$. Rovněž výsledný tok je rozdělen na $\mathbf{x}_e$ pro odpad odvážený do skládek a $\mathbf{y}_e$ pro odpad svážený do spalovny.

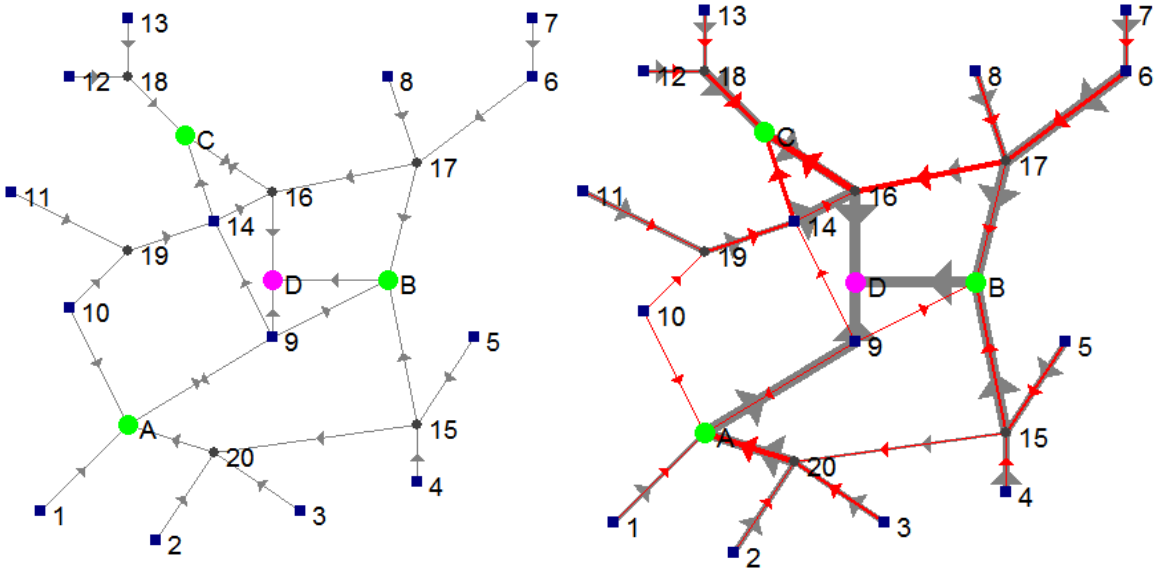
Model s dopravními uzly (4.6) upravíme do tvaru:

$$
\begin{aligned}
\max \quad & z = \textstyle\sum_{Icit} \sum_e [(-M_{Icit,e} \cdot x_e \cdot g_1) + (-M_{Icit,e} \cdot y_e \cdot g_2)] \\
& - \textstyle\sum_e [c_e(x_e + y_e)] - \sum_{Icit} (p_{1Icit} \cdot q_1 + p_{2Icit} \cdot q_2), \\
\text{s. t.} \quad & \textstyle\sum_e -M_{Icit,e} \cdot x_e \leq t_{1Icit}, && Icit = 1, ..., 14, \\
& \textstyle\sum_e -M_{Icit,e} \cdot y_e \leq t_{2Icit}, && Icit = 1, ..., 14, \\
& \textstyle\sum_e M_{Itra,e} \cdot x_e = 0, && Itra = 15, ..., 20, \\
& \textstyle\sum_e M_{Itra,e} \cdot y_e = 0, && Itra = 15, ..., 20, \\
& t_{1Icit} - \textstyle\sum_e (-M_{Icit,e} \cdot x_e) = p_{1Icit}, && Icit = 1, ..., 14, \\
& t_{2Icit} - \textstyle\sum_e (-M_{Icit,e} \cdot y_e) = p_{2Icit}, && Icit = 1, ..., 14, \\
& \textstyle\sum_e (M_{Ispa,e} \cdot x_e) = 0, && Ispa = D, \\
& \textstyle\sum_e (M_{Ilan,e} \cdot y_e) = 0, && Ilan = A, B, C, \\
& x_e + y_e \leq w_e, && e = 1, ..., 31, \\
& x_e \geq 0, && e = 1, ..., 31, \\
& y_e \geq 0, && e = 1, ..., 31, \\
& p_{1Icit} \geq 0, && e = 1, ..., 31, \\
& p_{2Icit} \geq 0, && e = 1, ..., 31.
\end{aligned}
\tag{4.7}
$$

Pro lepší porozumění se můžete podívat do seznamu zkretek na straně 40. Změny oproti předchozím modelům jsou komplexní, a proto nebyly změny modelu označeny červeně. Účelová funkce se skládá ze tří částí. První, kladná část, představuje čistý zisk, ze kterého jsou odečítány náklady za dopravu a pokuty za nesvezení odpadu. Jednotlivé omezující podmínky jsou zde „dvakrát“, pro každý odpad zvlášť. Podmínka 1 a 2 omezuje množství odvezeného odpadu z měst, 3. a 4. je vztažena k dopravním uzlům, 5. a 6. vypočítává počet neodvezených jednotek odpadu, 7. a 8. zamezuje toku vedoucímu do skládky, aby se nesvezl do spalovny a naopak, 9. podmínka omezuje propustnost hran. Následující omezení proměnných zaručuje jejich nezápornost. Tímto způsobem by se dala vymodelovat i síť s více než dvěma druhy odpadu.

Výsledky hledaných parametrů:

```
           VARIABLE x.L  quantity of trash1 transported to landfill

1-A   20.000,    2-20  13.000,    20-A  40.000,    3-20  22.000,    15-20  5.000
4-15  15.000,    5-15  16.000,    15-B  26.000,    9-B    8.000,    9-A    8.000
10-A  14.000,    9-14   3.000,    10-19  2.000,    11-19 12.000,    14-C  30.000
14-16  4.000,    17-B  14.000,    17-16 56.000,    16-C  60.000,    8-17  23.000
6-17  47.000,    7-6   25.000,    18-C  26.000,    12-18 16.000,    13-18 10.000
19-14 14.000

           VARIABLE y.L  quantity of trash2 transported to incinerator

1-A    6.000,    2-20   8.000,    20-A  17.000,    3-20   8.000,    15-20  1.000
4-15   9.000,    5-15   6.000,    15-B  14.000,    10-A   1.000,    11-19  5.000
14-16 10.000,    17-B  16.000,    17-16  4.000,    8-17   7.000,    6-17  13.000
7-6    5.000,    18-C  16.000,    12-18  8.000,    13-18  8.000,    19-14  5.000
16-D  30.000,    9-D   30.000,    B-D   30.000,    C-16  16.000,    A-9   24.000
                VARIABLE penalty1.L                    VARIABLE penalty2.L

  3  6.000,    10 2.000                  7  4.000,    8  1.000,    10 4.000
    PARAMETER tot_shipping          =     4012.000  transportation costs
    PARAMETER tot_penalty           =     6100.000  penalty
    PARAMETER tot_profit            =    42600.000  pure profit
    VARIABLE z.L                    =    32488.000  maximal profit
```



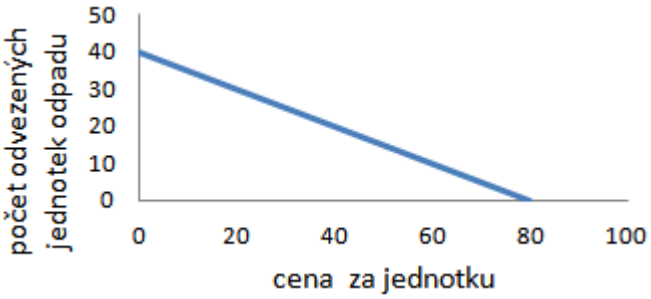
Obrázek 4.8: Vlevo rozšířená testovací síť, vpravo výsledná síť pro model (4.7).

Z výčtu výsledků vychází, že model (4.7) u obou typů odpadu za část zaplatil penále. Celkový zisk z však zůstal kladný. Model si můžete prohlédnout na CD pod názvem: `Different_trash02`.

4.8 Dynamická cena

V předešlých modelech byla cena za odvoz odpadu fixní a pro všechna města stejná. Motivací pro další postup může být případ ze života, kde si zákazník objednává zboží. Pokud si ale objedná zboží ve větším množství, dodavatel mu poskytne množstevní slevu. Takto lze určit závislost mezi zbožím a cenou.
V našem případě se sváží odpad z měst do skládek a cena s rostoucím množství odvezeného odpadu bude klesat (v našem případě lineárně). Závislost uvedeme s funkčním předpisem:

$$t_{Icit} = 40 - 0.5 \cdot g_{Icit}$$



Kde cena je ohraničena a leží v intervalu $\langle 20; 80 \rangle$. Model s dopravními uzly (4.6) se rozepíše do tvaru:

$$\begin{array}{lll}
\max & z = \sum_{Icit} \sum_e (-M_{Icit,e} \cdot x_e) \cdot g_{Icit} - \sum_e c_e x_e - \sum_{Icit} p_{Icit} \cdot q, & \\
\text{s. t.} & \sum_e -M_{Icit,e} \cdot x_e = \beta - \alpha \cdot g_{Icit}, & Icit = 1, ..., 14, \\
& t_{Icit} \geq \beta - \alpha \cdot g_{Icit}, & Icit = 1, ..., 14, \\
& \sum_e M_{Itra,e} \cdot x_e = 0, & Itra = 15, ..., 20, \\
& t_{Icit} - \sum_e (-M_{Icit,e} \cdot x_e) = p(Icit), & Icit = 1, ..., 14, \quad (4.8) \\
& x_e \leq w_e, & e = 1, ..., 26, \\
& x_e \geq 0, & e = 1, ..., 26, \\
& p_{Icit} \geq 0, & e = 1, ..., 26, \\
& l \leq g_{Icit}, & Icit = 1, ..., 14, \\
& g_{Icit} \geq u, & Icit = 1, ..., 14.
\end{array}$$

Výše uvedená přímka je v modelu zapsána ve tvaru: $\beta - \alpha \cdot g_{Icit}$, kde koeficienty $\beta = 40$ a $\alpha = -0.5$. Novou neznámou je zde cena za svoz g_{Icit}. Model (4.8) je nelineární vzhledem k účelové funkci, ve které jsou násobeny dvě neznámé $x_e \cdot g_{Icit}$. Přikládám výsledky z GAMSu a Aimmsu. Vstupní parametry byly zvoleny stejně jako v případě (4.6):

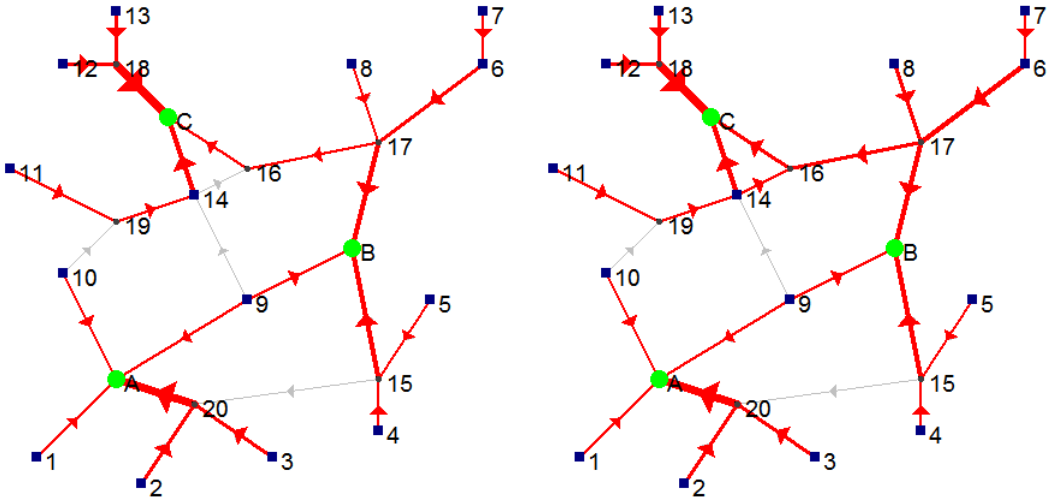
```
            VARIABLE g.L

1  52.000,    2  34.000,    3  28.500,    4  48.000,    5  56.000,    6  48.000
7  66.500,    8  49.000,    9  50.000,    10 50.000,    11 58.000,    12 30.000
13 34.000,    14 42.000


            PARAMETER av_g                =       46.143

            VARIABLE x.L  variable of quantity transported through a edge e

1-A   14.000,    2-20  23.000,    20-A  48.750,    3-20  25.750,    4-15  16.000
5-15  12.000,    15-B  28.000,    9-B    8.000,    9-A    7.000,    10-A  15.000
11-19 11.000,    14-C  30.000,    17-B  30.000,    17-16  8.250,    16-C   8.250
8-17  15.500,    6-17  22.750,    7-6    6.750,    18-C  48.000,    12-18 25.000
13-18 23.000,    19-14 11.000


            VARIABLE penalty.L

3   0.250,    6   4.000,    7   6.250,    8  12.500,    10  1.000,    11 11.000


            VARIABLE z.L                  =   -10070.250  total transportation
                                                          costs
            PARAMETER tot_profit          =    10128.250  pure profit
            PARAMETER tot_shipping        =    16698.500  total transportation
                                                          costs
            PARAMETER tot_penalty         =     3500.000  penalty
```

Výsledek je překvapivý, v porovnání s modelem (4.6) horší. Celkový zisk $z = -10070.250$ je ještě menší. Kde se stala chyba? Předpokladem modelu s dynamickou cenou bylo získání lepších výsledků.
Pokud se podíváme na parametr av_g, který nám vypisuje průměrnou hodnotu ceny za svoz, zjistíme, že je $av\_g = 46.143$, což je méně, než u modelu (4.6), kde jsme měli fixně nastavenou cenu na hodnotě $g = 50$. Odpověď na naši otázku se nachází v ceně, kterou určujeme pomocí přímky, jejíž koeficienty jsou konstanty. Pokud například zvolíme β neznámou, docílíme větší variability. Výsledky jsou následující:

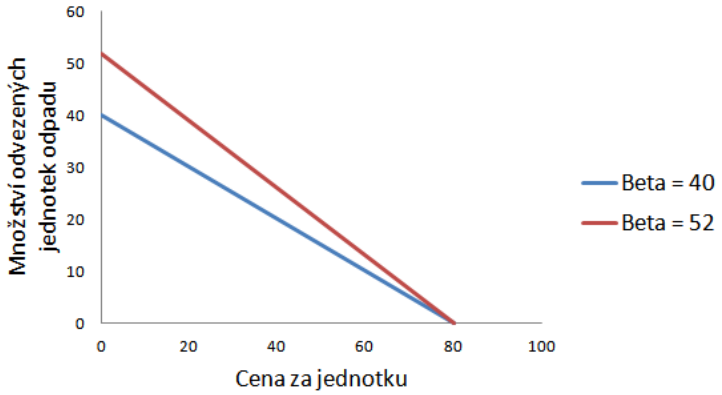
```
    VARIABLE beta.L               =       52.000
    PARAMETER av_g                =       67.679
    VARIABLE z.L                  =    -4190.375  total transportation
                                                  costs
    PARAMETER tot_profit          =    16572.375  pure profit
    PARAMETER tot_shipping        =    18987.750  total transportation
                                                  costs
    PARAMETER tot_penalty         =     1775.000  penalty
```

Z výsledků plyne, že se β zvětšila, z čehož vyplývá i nárůst průměrné ceny za svoz, která je již v porovnání s modelem (4.6) větší. Celkově tedy můžeme říci, že jsme utržili více peněz. K tomu přispělo i zmenšení penále, ze kterého vyplývá, že se odvezlo více jednotek

odpadu. To je patrné v ilustraci 4.9, kde se použila navíc hrana $14-16$ z důvodů dosažení maximálního toku na hraně $14-C$.



Obrázek 4.9: Výsledné sítě. Vlevo pro $\beta = 40$. Vpravo pro $\beta = 52$.



Obrázek 4.9: Grafy cen v závislosti na počtu odvezeného odpadu.

Model je přiložen na CD pod názvem: `Dynamic_pricing07_pen02`.

5 Reálná data

5.1 Zpětná modelace toku

Reálná data mi poskytl Ing. Radovan Šomplák z Odboru procesního inženýrství, který se zabývá problematikou svozu a využití odpadu.

Mou úlohou je data zpracovat a „zpětně vymodelovat“ tok při známé produkci odpadu ve městech a množstvím odpadu dovezeném do jednotlivých skládek a následně vymodelovat tok pro situaci, kde skládky nejsou kapacitně omezené, a oba modely porovnat. Přitom toky jsou vždy rekonstruovány tak, aby minimalizovaly úhrnné přepravní náklady.

Samotné modely jsou značně zjednodušeny. Města tvoří množina obcí s rozšířenou působností, tzv. ORP. Spalovny nejsou od skládek rozlišovány. Pro modely je použita pouze možnost dopravy po silnici. Nejsou zde zahrnuty mýtné brány ani další vlivy.
Rozhodujícím faktorem je tedy vzdálenost měst a skládek. V modelech je zavedeno označení:

- Množinu měst tvoří 206 ORP, označených číslem bez koncovky: například 155.
- Množinu skládek a spaloven činí 114 uzlů, jejichž koncovkou za číslem je ka nebo s: například $11ka; 68s$.

Celkově máme 320 uzlů, které jsou mezi sebou propojeny 2040 hranami. Účelovou funkci budeme minimalizovat a z pro nás bude představovat nejnižší možnou částku za dopravu. První model se dá srovnat s modelem (5.3). Je zapsán ve tvaru:

$$\begin{aligned} \min \quad & z = \textstyle\sum_e c_e \cdot x_e, \\ \text{s.t..} \quad & \textstyle\sum_e M_{i,e} \cdot x_e = t_i, \quad i = 1, ..., 320, \\ & x_e \geq 0, \qquad\quad e = 1, ..., 2040, \end{aligned} \tag{5.1}$$

kde index i značí uzly a index e hrany. V jediné omezující podmínce je zapsána rovnost z důvodu vymodelování toku tak, jak byl. Tedy aby odpovídalo odvezené množství odpadu dovezenému. Druhý model, založený na minimalizaci ceny za přepravu bez kapacitního omezení skládek, získáme úpravou předchozího modelu do tvaru:

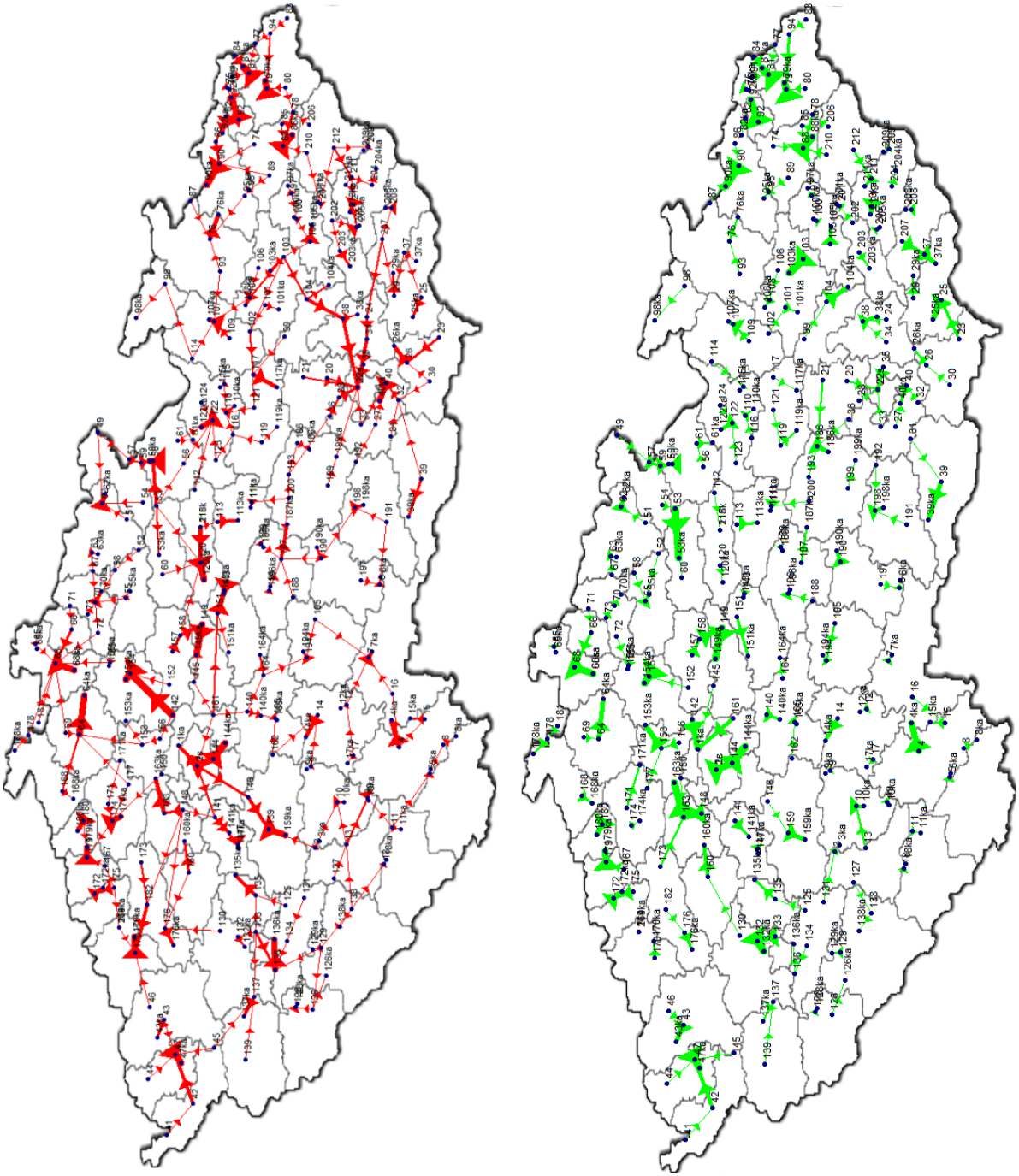
$$\begin{aligned} \min \quad & z = \textstyle\sum_e c_e \cdot x_e, \\ \text{s.t.} \quad & \textstyle\sum_e M_{Icit,e} \cdot x_e = t_{Icit}, \quad Icit = 1, ..., 206, \\ & x_e \geq 0, \qquad\quad e = 1, ..., 2040, \end{aligned} \tag{5.2}$$

Výsledné ceny pro oba modely jsou zapsány v tabulce:

Model	Celkové náklady za dopravu
Model 1	212 675 557 Kč
Model 2	118 540 321 Kč

První model, (5.1), byl zpětným vymodelováním toku. Spíše než o optimalizaci zde šlo pouze o grafické znázornění toku odpadu. Druhý model, (5.2), nemá kapacitní omezení skládek. Určuje nám, kam a v jakém množství by se odpadl svezl v ideálním případě.

Z porovnání modelů plyne, že celkové náklady za dopravu se mohou zmenšit o více než polovinu původní částky. Zde je názorně vidět, že optimalizace má své uplatnění a i změna pouhých pár procent může znamenat veliké úspory. Výsledné grafické sítě si můžete prohlédnout v ilustraci 5.1. Oba modely si můžete prohlédnout na CD pod názvy: `Real_data` a `Real_data02`.



Obrázek 5.1: Výsledné sítě. Vlevo pro model (5.1) (červeně). Vpravo pro model (5.2) (zeleně).

5.2 Model pro optimální rozložení spaloven

V této části bude vytvořen komplikovanější model, ve kterém budou provázány modely z kapitoly 4 s reálnými daty. Zadání: *Ke stávajícímu stavu skládek najít 10 optimálních míst pro spalovny, kde každá z nich bude kapacitně omezena na 1/10 celkového množství odpadu. Za dovezený odpad do spalovny dostane dopravní společnost zaplaceno, naopak za svezení odpadu na skládku zaplatí dopravní společnost malé penále. Pokud však odpad z měst nevyveze vůbec, zaplatí několikanásobně vyšší pokutu.*

Samotnou úlohu jsem rozdělil na dvě části z důvodu nelinearity účelové funkce. V první části hledám 10 vhodných míst pro spalovny. Tato část je identická s modelem (4.4). Hlavní myšlenkou zde je propojení všech měst se spalovnami, které je zprostředkováno incidenční maticí a následně pomocí euklidovské vzdálenosti hledáme lokálně optimální souřadnice spaloven. Takto formulovaný model je zapsán ve tvaru:

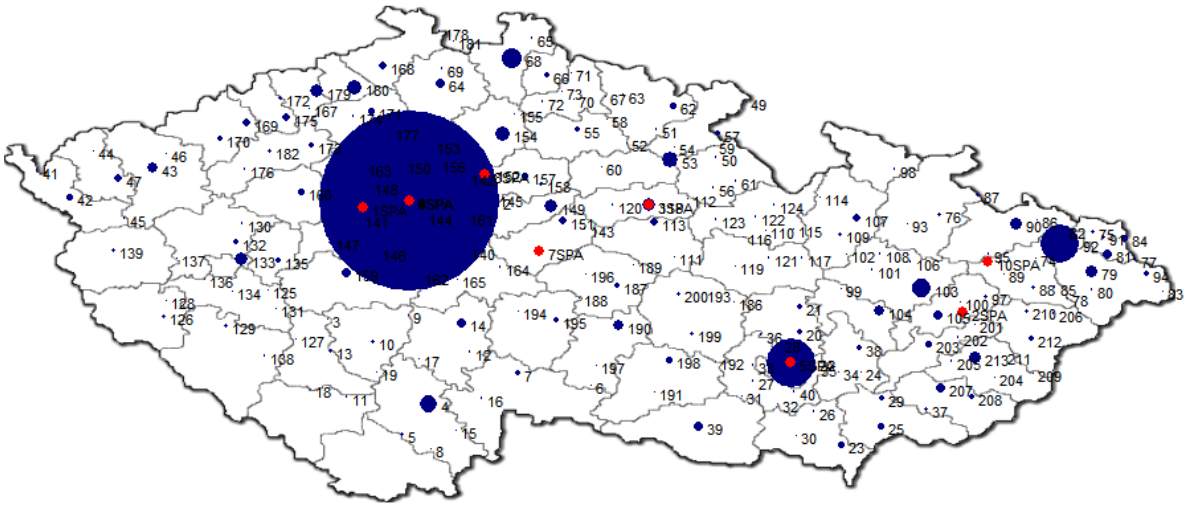
$$
\begin{aligned}
\min \quad & z = \textstyle\sum_{Icit} \sum_{Ispa} d_{Icit,Ispa} \cdot x_{Icit,Ispa}, & & \\
\text{s.t.} \quad & d_{Icit,Ispa} = \sqrt{\textstyle\sum_k (n3_{Ispa,k} - n1_{Icit,k})^2}, & & Ispa = 321,...,330, Icit = 1,...,206, \\
& \textstyle\sum_{Ispa} M_{Ispa,Icit} \cdot x_{Icit,Ispa} = t_{Icit}, & & Icit = 1,...,206, \\
& \textstyle\sum_{Icit} M_{Ispa,Icit} \cdot x_{Icit,Ispa} = t_{Ispa}, & & Ispa = 321,...,330, \\
& x_{Icit,Ispa} \geq 0, & & Icit = 1,...,206, Ispa = 321,...,330, \\
& d_{Icit,Ispa} \geq 0, & & Icit = 1,...,206, Ispa = 321,...,330, \\
& n3_{Ispa,k} \geq 0, & & Ispa = 321,...,330, k = X, Y.
\end{aligned}
\tag{5.3}
$$

Oproti minulým modelům zde přibyl index $Ispa \subset i$, který označuje spalovny. Pomocí 100 simulací můžeme najít lepší lokálně optimální řešení. Pár simulací a výsledků účelových funkcí si můžete prohlédnout v níže přiložené tabulce.

Simulace	Výsledná hodnota účelové funkce
1	2552145,637
20	2534102,554
60	2489109,681
100	2489109,668

Výsledné souřadnice spaloven poté vezmeme ze simulace, jejíž účelová funkce dosáhla nejmenší hodnoty. Druhá tabulka ukazuje souřadnice jednotlivých skládek, jejichž rozložení si můžete prohlédnout na přiložené ilustraci 5.2.

Spalovny	X	Y
1SPA	14,09410116	56,82412641
2SPA	17,59979680	56,21654270
3SPA	15,77038461	56,84360380
4SPA	14,36548808	56,86151748
5SPA	16,59542022	55,91753972
6SPA	14,36549645	56,86152188
7SPA	15,12560199	56,56668440
8SPA	14,80559639	57,01758982
9SPA	14,36548338	56,86152912
10SPA	17,75285681	56,50854595



Obrázek 5.2: Rozložení spaloven (červené), velikost měst (modré) je úměrná množství odpadu.

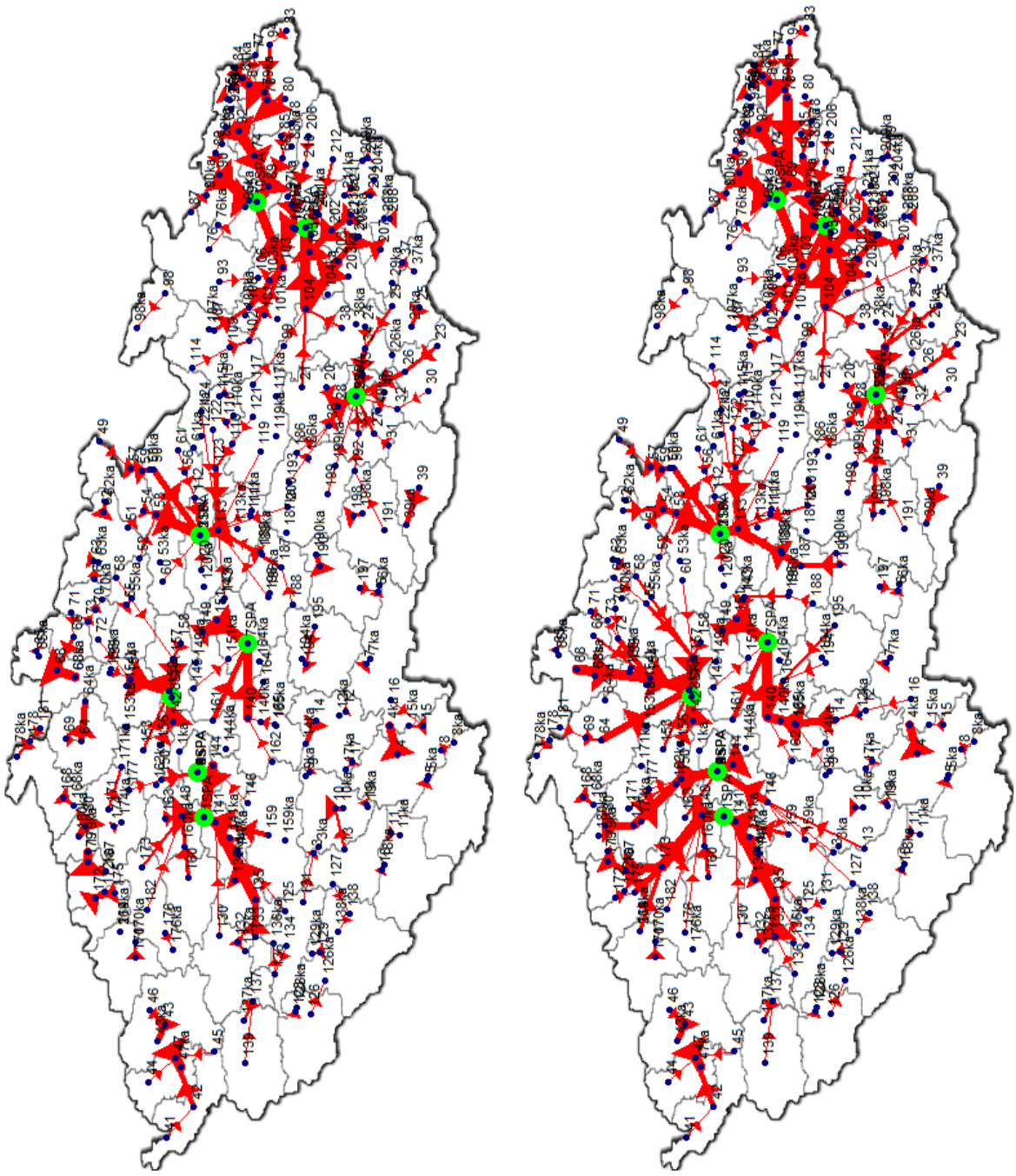
Ve druhé části již budu brát místa spaloven jako známé parametry a pro sestavení programu využiji modely (4.5) a (4.6). Zatímco jsem v první části hledal minimum, nyní je úlohou maximalizace zisku. Přičemž jsou dvě místa, kam se může odpad svážet: skládky a spalovny. Model je ve tvaru:

$$
\begin{aligned}
\max \quad & z = \textstyle\sum_{Ispa} \sum_e M_{Ispa,e} \cdot x_e \cdot g_1 - \sum_e d_e \cdot x_e - \sum_{Ilan} p1_{Ilan} \cdot g_2 - \sum_{Icit} p2_{Icit} \cdot g_3, & \\
\text{s.t.} \quad & p1_{Ilan} = \textstyle\sum_e x_e \cdot M_{Ilan,e}, & Ilan = 207, ..., 320, \\
& p2_{Icit} = t_{Icit} + \textstyle\sum_e x_e \cdot M_{Icit,e}, & Icit = 1, ..., 206, \\
& \textstyle\sum_e M_{Icit,e} \cdot x_e \leq t_{Icit}, & Icit = 1, ..., 206, \\
& \textstyle\sum_e M_{Ispa,e} \cdot x_e \leq t_{Ispa}, & Ispa = 321, ..., 330, \\
& x_e \geq 0, & e = 1, ..., 2069, \\
& p1_{Ilan} \geq 0, & Ilan = 207, ..., 320, \\
& p2_{Icit} \geq 0, & Icit = 1, ..., 206,
\end{aligned}
\tag{5.4}
$$

kde g_1, g_2, g_3 jsou konstanty. První z nich představuje zisk za odpad dovezený do spalovny a druhé dvě penalizační částky za odpad dovezený do skládek nebo nevyvezený odpad z měst. Proměnná $p1_{Ilan}$ je množství odpadu odvezeného na skládky a proměnná $p2_{Icit}$ značí množství nevyvezeného odpadu z měst. Vhodnou volbou zisku a penalizací ovlivňujeme celkový tok. V případě vyšší ceny zisku se zvětšuje oblast, ze které se odpad sváží do spaloven. Pokud je zisk nižší, daleko více odpadu se sveze pouze na skládky i za cenu zaplacení penále. Pro simulace jsem použil hodnoty g_1, g_2, g_3 uvedené v níže přiložené tabulce:

Simulace	Zisk ze spalovny	Sankce za odvoz na skládku	Sankce za nevyvezení odpadu
1. nižší zisk	300	10	1000
2. vyšší zisk	500	10	1000

Výsledné grafické řešení pro obě simulace si můžete prohlédnout v ilustraci 5.3.



Obrázek 5.3: Výsledné sítě. Vlevo model s nižším ziskem ze spaloven, vpravo model při příznivém stavu zisku.

Modely naleznete na CD pod názvy: `Real_data03` a `Real_data05`.

6 Závěr

První tři kapitoly tvořily teoretickou část práce, ve které jsem uvedl základní pojmy z teorie grafů, optimalizace a v tocích v sítích podrobně rozebral úlohu z logistiky.

Zadefinované pojmy jsem aplikoval v částech 4 a 5. Ve 4. kapitole jsem se věnoval modelům z oblasti logistiky, konkrétně modelům svozu komunálního odpadu. Pro grafické znázornění jsem využil dva typy testovacích sítí. Síť bez dopravních uzlů a síť s dopravními uzly. Výchozím, nejjednodušším modelem se pro mne stala úloha minimalizace ceny za přepravu, kterou jsem dále postupně rozšiřoval, nejprve do tvaru úlohy s kapacitním omezením, a následně zkoumal možnosti optimálnějšího rozložení skládek. Otázku minimalizace ceny za dopravu jsem převedl na otázku maximalizace zisku a logistickou úlohu více komplikoval přidáním sankcí při nedodržení podmínek, rozdělením odpadu na více druhů, kde se rozlišovalo svážení odpadu na skládky neb do spaloven. Posledním modelem 4. části pak byla úloha, kdy cena za svezený odpad nebyla konstantní, ale měla lineární závislost.

Kapitola 5 je věnována aplikacím modelů z předchozí části na reálná data o svozu komunálního odpadu. Za spolupráce s Ing. Radovanem Šomplákem z Odboru procesního inženýrství jsem řešil úlohu rekonstrukce toku odpadu při minimalizaci přepravních nákladů a následně ji porovnával s idealizovaným modelem. Ve druhé části jsem se zabýval komplikovanější a komplexnější úlohou pro optimální rozložení spaloven, ve které jsou aplikovány modely a postupy ze 4. kapitoly. Vzhledem k rozdílnosti jednotlivých úloh ze 4. a 5. kapitoly nemohu všechny výsledky modelů porovnat mezi sebou. Dílčí výsledky jsou uvedeny vždy na konci příslušné části, kde naleznete i případné porovnání výsledků s podobnými modely.

Veškeré vstupy jsou v modelech známé hodnoty, konstanty. Modely v práci tedy můžeme označit jako deterministické. V reálném světě se však hodnoty pohybují v určitých intervalech. Příkladem může být kolísání ceny za litr benzínu nebo množství odpadu ve městech, které není při každém svozu stejné. Tomuto problému se věnuje stochastické programování, kterým bychom se opět o kus přiblížili k realitě.

7 Literatura

[1] FOLTÝNEK, T.: *Teorie grafů*, První vydání, Mendelova univerzita, Provozně ekonomická fakulta, 2011, 90 s, ISBN 978-80-7375-500-3.

[2] ŠEDA, M., DANNHOFEROVÁ, J.: *Teorie grafů*, Vysoké učení technické v Brně, Fakulta strojního inženýrství, UAI, 2003.
Dostupné z $\langle http://www.uai.fme.vutbr.cz/\ mseda/TG03\_MS.pdf \rangle$.

[3] NEŠETŘIL, J.: *Teorie grafů*, SNTL, Nakladatelství technické literatury, 1979, 320 s.

[4] RARDIN, R. L.:*Optimization in Operations Research*, Prentice Hall, 1998, 919 s., ISBN 9780023984150.

[5] DANTZIG, G. B., THAPA, M. N.:*Linear Programming 1: Introduction*, první vydání, Stanford University, Stanford, 1997, ISBN 0-387-94833-3.

[6] KALL, P., WALLACE, S. W.:*Stochastic Programming*, první vydání, John Wiley & Sons, Chichester, 315 s.

[7] BRADLEY, S. P., HAX, A. C., MAGNANTI, T. L.: *Applied Mathematical Programming*, Addison-Wesley, 1977, 716 s., ISBN 978-0201004649,
dostupné z $\langle http://web.mit.edu/15.053/www/AppliedMathematicalProgramming.pdf \rangle$.

[8] GRIVA, I., NASH, S. G., SOFER, A.: *Linear and Nonlinear Optimization*, druhé vydání, Society for Industrial Mathematics, 2008, 764 s., ISBN 978-0898716610.

[9] HRABEC, D.:*Modely stochastického programování pro inženýrský návrh*, diplomová práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, ÚM, 2011, 70 s.

[10] GHIANI, G., LAPORTE, G., MUSMANNO, R.: *Introduction to Logistic Systems Planning and Control*, John Wiley & Sons, Chichester, 2004.

8 Seznam zkratek

1. Seznam zkratek pro modely bez dopravních uzlů

Indexy	
i	města
j	skládky
k	souřadnice
Skaláry	
l, u	dolní, horní mez
z	účelová funkce
Parametry	
$\mathbf{d}_{i,j}$	vzdálenost město-skládka
$\mathbf{c}_{i,j}$	cena za dopravu
$\mathbf{t}_i$	kapacita města
$\mathbf{w}_e$	horní mez hran
$\mathbf{M}_{i,j}$	incidenční matice
$\mathbf{x}_{i,j}$	množství převezeného odpadu (tok)
$\mathbf{n1}_{i,k}$	souřadnice měst
$\mathbf{n2}_{j,k}$	souřadnice skládek

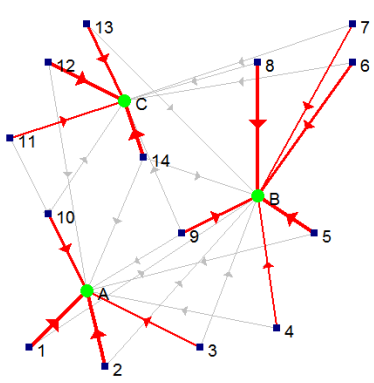
2. Seznam zkratek pro modely s dopravními uzly

Indexy	
i, j	uzly
Icit	města
Ilan	skládky
Itra	dopravní uzly
Ispa	spalovny
e	hrany
k	souřadnice
Skaláry	
l, u	dolní, horní mez
α, β	koeficienty přímky
g, *g1*, *g2*, *g3*	cena za jednotku odpadu
q, *q1*, *q2*	pokuta za jednotku odpadu
z	účelová funkce
Parametry	
$\mathbf{d}_e$	vzdálenost mezi uzly
$\mathbf{c}_e$	cena za dopravu
$\mathbf{t}_i$	kapacita uzlu
$\mathbf{w}_e$	horní mez hran
$\mathbf{M}_{i,e}$	incidenční matice
$\mathbf{x}_e, \mathbf{y}_e$	množství převezeného odpadu (tok)
$\mathbf{n1}_{Icit,k}$	souřadnice měst
$\mathbf{n2}_{Ilan,k}$	souřadnice skládek
$\mathbf{n3}_{Ispa,k}$	souřadnice spaloven
$\mathbf{p}_{Icit}$, $\mathbf{p1}_{Ilan}$, $\mathbf{p2}_{Icit}$	množství neodvezených jednotek odpadu
$\mathbf{g}_{Icit}$	zisk za jednotku odpadu
$\mathbf{f}_{Ilan}$	procentuální omezení

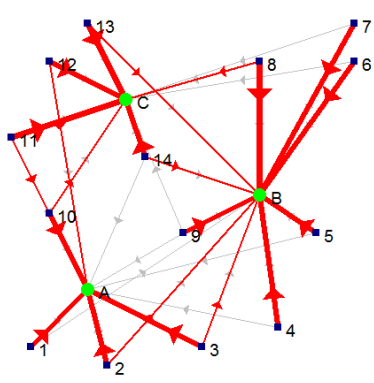
9 Seznam příloh

A Přehled ilustrací modelů 4. kapitoly

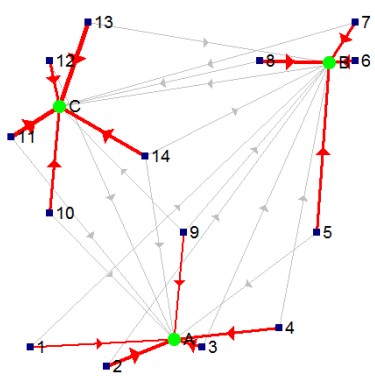
A1 Modely 4. kapitoly



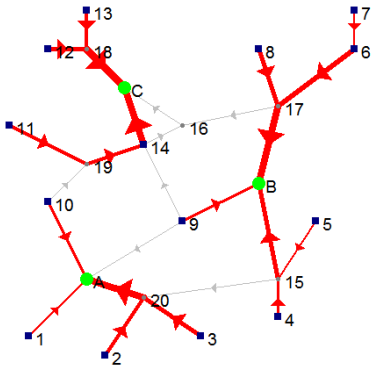
Obrázek 9.1: Model 4.2.



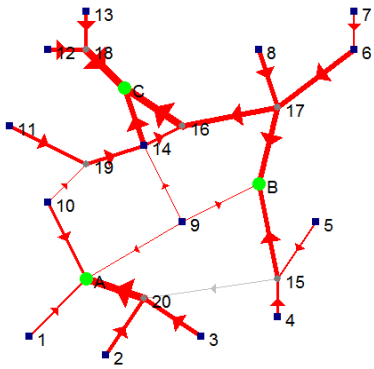
Obrázek 9.2: Model 4.3.



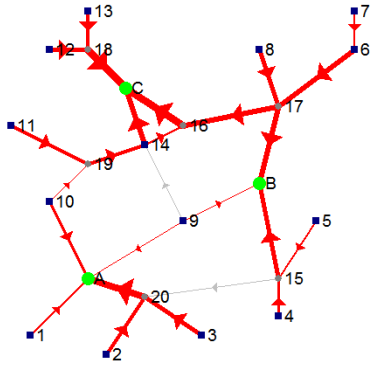
Obrázek 9.3: Model 4.4.



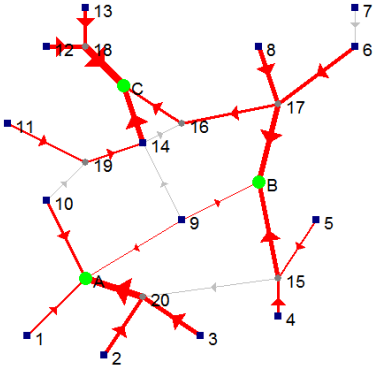
Obrázek 9.4: Model 4.2.



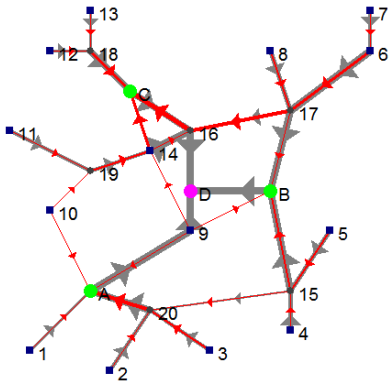
Obrázek 9.5: Model 4.3.



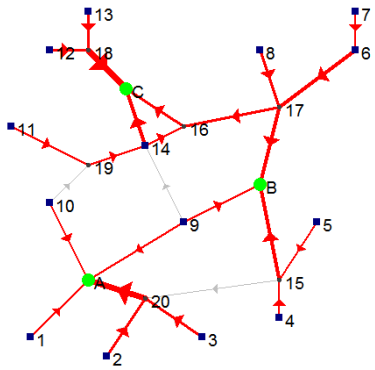
Obrázek 9.6: Model 4.5.



Obrázek 9.7: Model 4.6.

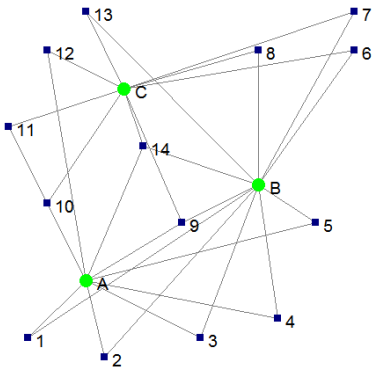


Obrázek 9.8: Model 4.7.

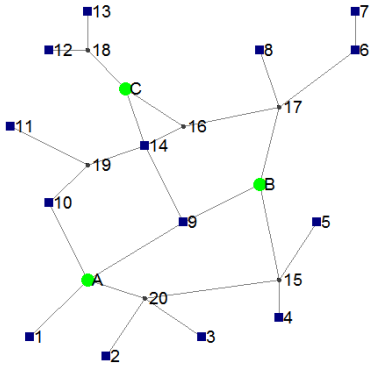


Obrázek 9.9: Model 4.8.

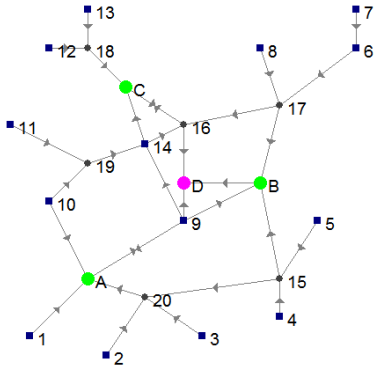
A2 Testovací sítě



Obrázek 9.10: Testovací síť bez dopravních uzlů.



Obrázek 9.11: Testovací síť s dopravními uzly.



Obrázek 9.12: Testovací síť pro model 4.8.

B Přílohy na CD

KAPITOLA 4

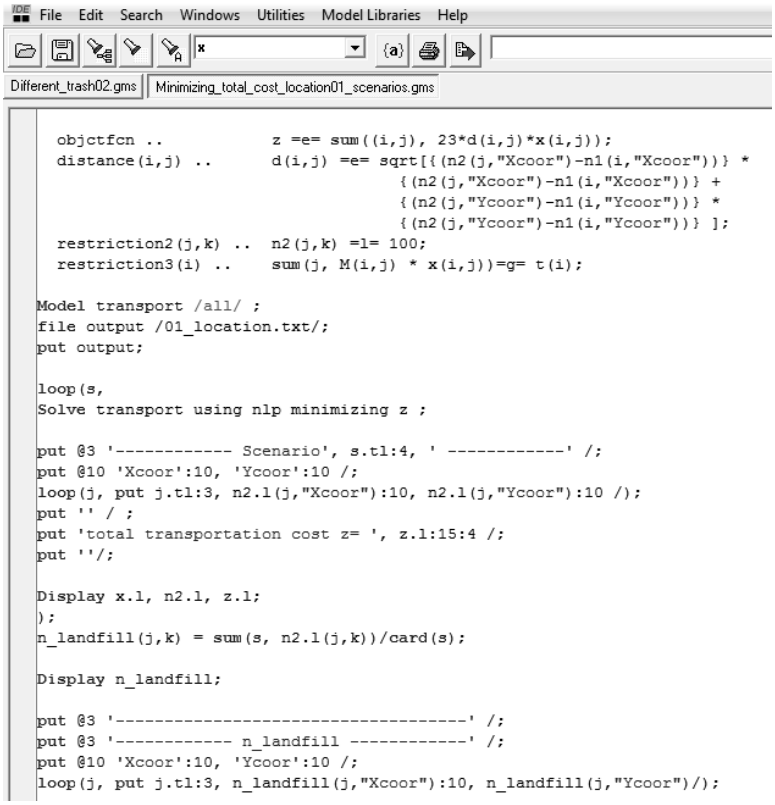
- Pro kapitolu 4.2:
  - `Minimizing_total_cost01`
  - `Minimizing_total_cost05`
- Pro kapitolu 4.3:
  - `Minimizing_total_cost04`
  - `Minimizing_total_cost06`
- Pro kapitolu 4.4:
  - `Minimizing_total_cost_location01_scenarios`
- Pro kapitolu 4.5:
  - `Maximizing_of_profit06_tra`
- Pro kapitolu 4.6:
  - `Maximizing_of_profit06_tra_penalty`
  - `Maximizing_profit04_penalty`
- Pro kapitolu 4.7:
  - `Different_trash02`
- Pro kapitolu 4.8:
  - `Dynamic_pricing07_pen02`

KAPITOLA 5

- Pro kapitolu 5.1:
  - `Real_data`
  - `Real_data02`
- Pro kapitolu 5.2:
  - `Real_data03`
  - `Real_data05`

C GAMS

Program GAMS, celým názvem General Algebraic Modeling System, je program pro matematické modelování a optimalizaci na vysoké úrovni. Skládá se z kompilátoru jazyka a integrovaných vysoce výkonných řešičů. Program je vhodný pro komplexní aplikace. Dovoluje vytvářet rozsáhlé snadno udržovatelné modely, které se mohou rychle adaptovat novým podmínkám.
Následující ilustrace přibližuje prostředí programu GAMS.



Obrázek C.1: Ukázka z prostředí programu GAMS.

Zápis v GAMSu se provádí postupně od deklarace indexů příkazem `Set` přes zápis známých parametrů, které jsou děleny na skaláry (`Scalar`), jednoindexové parametry (`Parameter`) a víceindexové parametry (`Table`) až po deklaraci proměnných příkazem `Variables`. Následuje trojice příkazů, klíčových pro deklaraci rovnic, omezení a jejich vyřešení. Jedná se o příkazy `Equations, Model` a `Solve`. Pro výpis výsledků se používá příkaz `Display`, ve kterém specifikujeme, co chceme vypsat.

Pro lepší porozumění jsem do příloh přidal dva zdrojové kódy napsané v programu GAMS. Ostatní zdrojové kódy jsou přiloženy na CD.

Více informací o programu a jeho využití naleznete na webových stránkách softwaru.

MODEL 4.7: Different_trash02

```
sets i index of nodes /1*20, A, B,C/
     k index of coordinates /Xcoor, Ycoor/
     e index of edges /1-A, 2-20, 20-A, 3-20, 15-20, 4-15, 5-15, 15-B, 9-B,
                       9-A, 10-A, 9-14, 10-19, 11-19, 14-C, 14-16, 17-B, 17-16,
                       16-C, 8-17, 6-17, 7-6, 18-C, 12-18, 13-18, 19-14/;

sets Icit(i)     index of cities /1*14/,
     Itra(i)     index of traffic points /15*20/,
     Ilan1(i)    index of landfills1 /A,B,C/
     Ilan2(i)    index of landfill2 /D/;

Table M(i,e) incidence matrix
       1-A      2-20    20-A    3-20    15-20   4-15    5-15    15-B    9-B
1      -1
2               -1
3                               -1
4                                               -1
5                                                       -1
6
7
8
9                                                                               -1
10
11
12
13
14
15                                      -1      1       1       -1
16
17
18
19
20              1       -1      1       1
A      1                1
B                                                               1       1
C

+
       9-A     10-A    9-14    10-19   11-19   14-C    14-16   17-B    17-16
1
2
3
4
5
6
```

```
7
8
9       -1              -1
10              -1              -1
11                                      -1
12
13
14                      1                       -1      -1
15
16                                                      1               1
17                                                              -1      -1
18
19                              1       1
20
A       1       1
B                                                               1
C                                               1

+

        16-C    8-17    6-17    7-6     18-C    12-18   13-18   19-14
1
2
3
4
5
6                       -1      1
7                               -1
8               -1
9
10
11
12                                              -1
13                                                      -1
14                                                              1
15
16      -1
17              1       1
18                                      -1      1       1
19                                                              -1
20
A
B
C       1                               1
;
```

```
Parameter c(e) transportation cost;
c(e) = uniformint(1,10);

Parameter t1(Icit) quantity of trash1;
t1(Icit) = uniformint(10,30);

Parameter t2(Icit) quantity of trash1;
t2(Icit) = uniformint(5,10);

scalars g1 constant price which each city will pay for trash 1 /100/,
        g2 constant price which each city will pay for trash 2 /200/,
        r  total transportation costs,
        q1 penalty 1 /200/,
        q2 penalty 2 /500/,
        tot_shipping     transportation costs,
        tot_penalty      penalty,
        tot_profit       pure profit;

Parameter w(e) upper bound for edges
/1-A 30, 2-20 30, 20-A 80, 3-20 30, 15-20 20, 4-15 30,
5-15 30, 15-B 40, 9-B 8, 9-A 8, 10-A 15, 9-14 14,
10-19 15, 11-19 30, 14-C 30, 14-16 30, 17-B 30,
17-16 60, 16-C 60, 8-17 30, 6-17 60, 7-6 30, 18-C 60,
12-18 30, 13-18 30, 19-14 30, 16-D 30, 9-D 30, B-D 30,
C-16 30, A-9 30/;

variables
         x(e)            quantity of trash1 transported to landfill
         y(e)            quantity of trash2 transported to incinerator
         z               maximal profit
         p1(Icit)        amount of not-transported waste 1
         p2(Icit)        amount of not-transported waste 2;

Positive Variable x, y, p1, p2 ;

Equations
       cost                 define objective function (maximizing of profit)
       restriction1(Icit)   use only existing arcs
       restriction2(Icit)
       restriction3(Itra)   restriction for traffic points
       restriction4(Itra)
       restriction5(e)      upper bound for edges
       restriction6(Icit)   amount of not-transported waste
       restriction7(Icit)
       restriction8(Ilan2)  transport waste 1 to landfills 1
       restriction9(Ilan1);
```

```
cost ..                z =e= sum((Icit, e), (-M(Icit,e) * x(e)) * g1
                           + (-M(Icit,e) * y(e)) * g2)
                           - sum(e, c(e)*(x(e) + y(e)))
                           - sum(Icit, p1(Icit)*q1 + p2(Icit)*q2);
restriction1(Icit) ..  sum(e, -M(Icit,e)*x(e)) =l= t1(Icit);
restriction2(Icit) ..  sum(e, -M(Icit,e)*y(e)) =l= t2(Icit);
restriction3(Itra) ..  sum(e, M(Itra,e)*x(e)) =e= 0;
restriction4(Itra) ..  sum(e, M(Itra,e)*y(e)) =e= 0;
restriction5(e)    ..  y(e) + x(e) =l= w(e);
restriction6(Icit) ..  t1(Icit) + sum(e, x(e)*M(Icit,e)) =e= p1(Icit);
restriction7(Icit) ..  t2(Icit) + sum(e, y(e)*M(Icit,e)) =e= p2(Icit);
restriction8(Ilan2) .. sum(e, M(Ilan2,e)*x(e)) =e= 0;
restriction9(Ilan1) .. sum(e, M(Ilan1,e)*y(e)) =e= 0;

Model transport /all/ ;

Solve transport using lp maximizing z ;

tot_shipping =  sum(e, c(e)*(x.l(e) + y.l(e)));
tot_penalty  =  sum(Icit, p1.l(Icit)*q1 + p2.l(Icit)*q2);
tot_profit   =  sum((Icit, e), (-M(Icit,e) * x.l(e)) * g1
               + (-M(Icit,e) * y.l(e)) * g2);

Display c, t1, t2, z.l, x.l, y.l, w, p1.l, p2.l, tot_shipping,
        tot_penalty, tot_profit, z.l;
```

MODEL 4.4: Minimizing_total_cost_location01_scenarios

```
Sets    s scenarios /1*10/;
  i cities / 1*14 /
      j landfills /A, B, C/
      k coordinates / Xcoor, Ycoor /;

Table n1(i,k) coordinates of cities
                Xcoor    Ycoor
       1        10       10
       2        30       5
       3        55       10
       4        75       15
       5        85       40
       6        95       85
       7        95       95
       8        70       85
       9        50       40
       10       15       45
       11       5        65
       12       15       85
       13       25       95
       14       40       60     ;

Table n2(j,k) coordinates of landfills
               Xcoor    Ycoor
        A      25       25
        B      70       50
        C      35       75      ;

Table M(i,j) node-arc incidence matrix
                A       B       C
           1    1       1       0
           2    1       1       0
           3    1       1       0
           4    1       1       0
           5    1       1       0
           6    0       1       1
           7    0       1       1
           8    0       1       1
           9    1       1       1
           10   1       0       1
           11   1       0       1
           12   1       0       1
           13   1       1       0
           14   1       1       1;
```

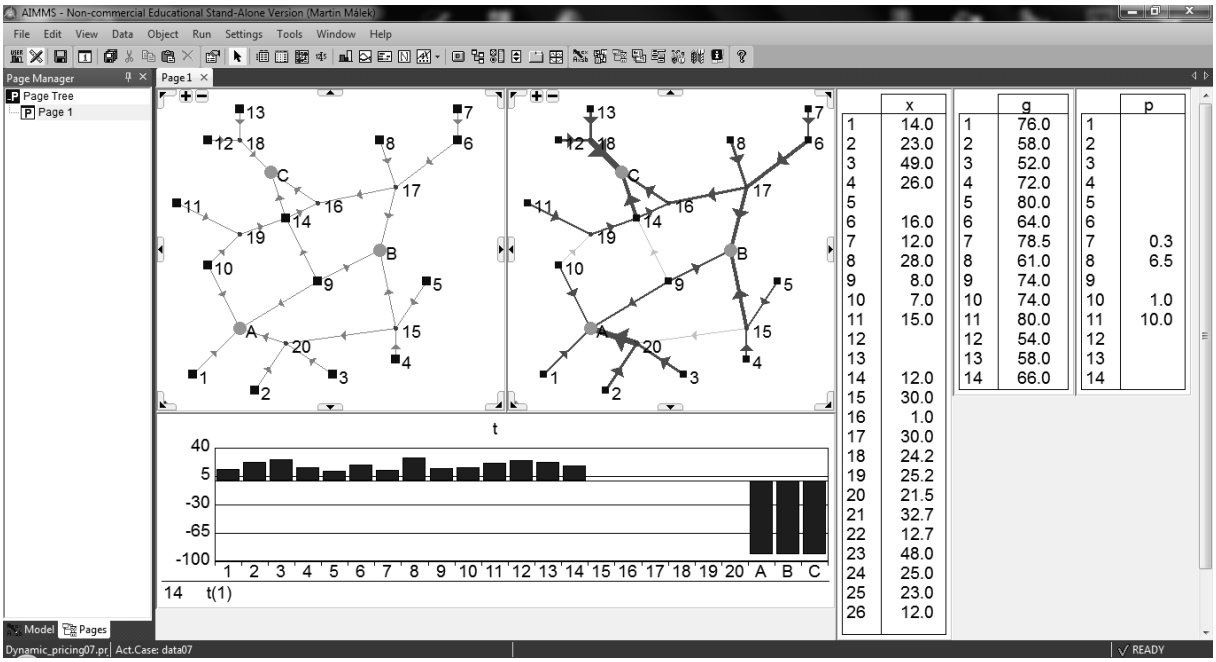
```
Parameter t(i) quantity of trash in cities;
t(i) = uniformint(50,100);

scalar f freight in coins per case using lorry /23/;

Parameter c(i,j) cost;
Parameter n_landfill(j,k);
Display M, t;

Variables
         z       objective function
         x(i,j)  amount of transported waste through edge
         n2(j,k) coordination of landfills
         d(i,j)  distance;

Positive variables
         x, n2, d;
Equations
         objctfcn
         distance(i,j)       distance count from i to j
         restriction2(j,k)
         restriction3(i);

  objctfcn ..           z =e= sum((i,j), 23*d(i,j)*x(i,j));
  distance(i,j) ..      d(i,j) =e= sqrt[{(n2(j,"Xcoor")-n1(i,"Xcoor"))} *
                                        {(n2(j,"Xcoor")-n1(i,"Xcoor"))} +
                                        {(n2(j,"Ycoor")-n1(i,"Ycoor"))} *
                                        {(n2(j,"Ycoor")-n1(i,"Ycoor"))} ];
  restriction2(j,k) ..  n2(j,k) =l= 100;
  restriction3(i) ..    sum(j, M(i,j) * x(i,j))=g= t(i);

Model transport /all/ ;
file output /01_location.txt/;
put output;

loop(s,
Solve transport using nlp minimizing z ;

put @3 '------------ Scenario', s.tl:4, ' ------------' /;
put @10 'Xcoor':10, 'Ycoor':10 /;
loop(j, put j.tl:3, n2.l(j,"Xcoor"):10, n2.l(j,"Ycoor"):10 /);
put '' / ;
put 'total transportation cost z= ', z.l:15:4 /;
put ''/;

Display x.l, n2.l, z.l;
);
```

```
n_landfill(j,k) = sum(s, n2.l(j,k))/card(s);

Display n_landfill;

put @3 '-----------------------------------' /;
put @3 '------------ n_landfill ------------' /;
put @10 'Xcoor':10, 'Ycoor':10 /;
loop(j, put j.tl:3, n_landfill(j,"Xcoor"):10, n_landfill(j,"Ycoor")/);
```

D AIMMS

Výstupem GAMSu je stránka formátu .lst, která se podobá klasickému textovému souboru. Logistické úlohy však umožňují celou situaci graficky znázornit pomocí teorie grafů. Většinu modelů jsem proto napsal i v programu AIMMS, který umožňuje celou situaci vymodelovat.

AIMMS byl poprvé uveden jako nástroj pro matematické modelování v roce 1993. Jedná se integrovanou kombinaci modelovacího jazyka, grafického uživatelského prostředí a numerických řešičů. Program je vhodný pro tvorbu pokročilých prototypů, které jsou snadně transformovatelné pro koncové uživatele. AIMMS našel uplatnění v oblastech zásobovacího řetězce a logistiky, automobilovým, chemickým, energetickým, ropným průmyslem aj. Je využíván velkými firmami po celém světě, např. Shell, Heineken, Lufthansa nebo PriceWaterhouseCoopers.



Obrázek D.1: Ukázka z uživatelského prostředí programu AIMMS.

Pro lepší představu přidávám zdrojové kódy z kapitoly 5: Reálná data, kterou jsem řešil pouze v programu AIMMS. Ostatní zdrojové kódy z kapitoly 4 jsou k dispozici na přiloženém CD.

Další informace o programu naleznete na oficiálních webových stránkách produktu, ze kterých jsem čerpal.

MODEL 5.1: Real_data

```
MAIN MODEL Main_real_data

  DECLARATION SECTION

    SET:
       identifier   :  nodes
       indices      :  i, j ;

    SET:
       identifier   :  coor
       index        :  k ;

    SET:
       identifier   :  i_cities
       subset of    :  nodes
       index        :  Icit ;

    SET:
       identifier   :  i_landfills
       subset of    :  nodes
       index        :  Ilan ;

    SET:
       identifier   :  edge
       index        :  e ;

    PARAMETER:
       identifier   :  i_coor
       index domain :  (i,k) ;

    PARAMETER:
       identifier   :  M
       index domain :  (i,e) ;

    PARAMETER:
       identifier   :  c
       index domain :  (e) ;

    PARAMETER:
       identifier   :  t
       index domain :  (i) ;

    VARIABLE:
       identifier   :  z
```

```
    range        :  free
    definition   :  sum(e, 5*c(e)*x(e))  ;

  VARIABLE:
    identifier   :  x
    index domain :  (e)
    range        :  nonnegative ;

  CONSTRAINT:
    identifier   :  r1
    index domain :  i
    definition   :  sum(e, M(i,e)*x(e)) = t(i)  ;

  MATHEMATICAL PROGRAM:
    identifier   :  Least_cost_transport_plan
    objective    :  z
    direction    :  minimize
    constraints  :  AllConstraints
    variables    :  AllVariables
    type         :  Automatic ;

ENDSECTION  ;

SECTION Excel_Input


  DECLARATION SECTION

    STRING PARAMETER:
      identifier :  WorkbookName
      definition :  "Real_data.xlsx" ;

    PARAMETER:
      identifier :  ExcelStatus ;

    STRING PARAMETER:
      identifier :  ExcelErrorMessage ;

    ELEMENT PARAMETER:
      identifier :  err
      range      :  errh::PendingErrors ;

  ENDSECTION  ;

  PROCEDURE
    identifier :  ReadDataFromExcel
    body       :
```

```
        Spreadsheet::SetActiveSheet( WorkbookName, "uzly" );

        Spreadsheet::RetrieveSet( WorkbookName, nodes, "A1:A320" );
        Spreadsheet::RetrieveSet( WorkbookName, coor, "K322:L322" );
        Spreadsheet::RetrieveParameter( WorkbookName, t, "J1:J320" );
        Spreadsheet::RetrieveTable( WorkbookName, i_coor, "coor3", "A1:A320",
"K322:L322");

        Spreadsheet::SetActiveSheet( WorkbookName, "incid" );

        Spreadsheet::RetrieveSet( WorkbookName, edge, "B1:BZM1" );
        Spreadsheet::RetrieveTable( WorkbookName, M, "incidence", "A2:A321",
"B1:BZM1" );
        Spreadsheet::RetrieveParameter( WorkbookName, c, "B323:BZM323" );

    ENDPROCEDURE  ;

  ENDSECTION Excel_Input ;

  SECTION Get_edges


    DECLARATION SECTION

      PARAMETER:
         identifier   :  flow
         index domain :  (e,i,j) ;

      PARAMETER:
         identifier   :  real_flow
         index domain :  (i,j) ;

    ENDSECTION  ;

    PROCEDURE
      identifier :  GetEdges
      body       :
        flow(e,i,j):= 0;
        for e in edge do
        for i in nodes do
        if M(i,e) = -1 then
        for j in nodes do
        if M(j,e) = 1 then
        flow(e,i,j):= x(e); break; endif; endfor; break; endif;

        endfor;
```

```
        endfor;

        real_flow(i,j):=sum(e,flow(e,i,j));

    ENDPROCEDURE  ;

  ENDSECTION Get_edges ;

  PROCEDURE
    identifier :  MainInitialization

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainExecution
    body       :
      solve Least_cost_transport_plan;

      if ( Least_cost_transport_plan.ProgramStatus <> 'optimal' ) then
                empty z;
           endif;

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainTermination
    body       :
      return DataManagementExit();

  ENDPROCEDURE  ;

ENDMODEL Main_real_data ;
```

MODEL 5.2: Real_data02

```
MAIN MODEL Main_Real_data02

  DECLARATION SECTION

    SET:
       identifier   :  nodes
       indices      :  i, j ;

    SET:
       identifier   :  edge
       index        :  e ;

    SET:
       identifier   :  coor
       index        :  k ;

    SET:
       identifier   :  i_cities
       subset of    :  nodes
       index        :  Icit ;

    SET:
       identifier   :  i_landfills
       subset of    :  nodes
       index        :  Ilan ;

    PARAMETER:
       identifier   :  i_coor
       index domain :  (i,k) ;

    PARAMETER:
       identifier   :  M
       index domain :  (i,e) ;

    PARAMETER:
       identifier   :  c
       index domain :  (e) ;

    PARAMETER:
       identifier   :  t
       index domain :  (Icit) ;

    VARIABLE:
       identifier   :  x
       index domain :  (e)
```

```
    range        :  nonnegative ;

  VARIABLE:
    identifier   :  z
    range        :  free
    definition   :  sum(e, 5*c(e)*x(e))  ;

  CONSTRAINT:
    identifier   :  r1
    index domain :  Icit
    definition   :  sum(e, M(Icit,e) * x(e)) = t(Icit)  ;

  MATHEMATICAL PROGRAM:
    identifier   :  Least_cost_transport_plan
    objective    :  z
    direction    :  minimize
    constraints  :  AllConstraints
    variables    :  AllVariables
    type         :  Automatic ;

ENDSECTION  ;

SECTION Excel_Input


  DECLARATION SECTION

    STRING PARAMETER:
      identifier :  WorkbookName
      definition :  "Real_data.xlsx" ;

    PARAMETER:
      identifier :  ExcelStatus ;

    STRING PARAMETER:
      identifier :  ExcelErrorMessage ;

    ELEMENT PARAMETER:
      identifier :  err
      range      :  errh::PendingErrors ;

  ENDSECTION  ;

  PROCEDURE
    identifier :  ReadDataFromExcel
    body       :
      Spreadsheet::SetActiveSheet( WorkbookName, "uzly" );
```

```
        Spreadsheet::RetrieveSet( WorkbookName, nodes, "A1:A320" );
        Spreadsheet::RetrieveSet( WorkbookName, coor, "K322:L322" );
        Spreadsheet::RetrieveSet( WorkbookName, i_cities, "A1:A206" );
        Spreadsheet::RetrieveSet( WorkbookName, i_landfills, "A207:A320" );
        Spreadsheet::RetrieveParameter( WorkbookName, t, "t_cit" );
        Spreadsheet::RetrieveTable( WorkbookName, i_coor, "coor3", "A1:A320",
"K322:L322");

        Spreadsheet::SetActiveSheet( WorkbookName, "incid" );

        Spreadsheet::RetrieveSet( WorkbookName, edge, "B1:BZM1" );
        Spreadsheet::RetrieveTable( WorkbookName, M, "incidence", "A2:A321",
"B1:BZM1" );
        Spreadsheet::RetrieveParameter( WorkbookName, c, "B323:BZM323" );

    ENDPROCEDURE  ;

  ENDSECTION Excel_Input ;

  SECTION Edges


    DECLARATION SECTION Auxiliary_Parameters

      PARAMETER:
         identifier   :  flow
         index domain :  (e,i,j) ;

      PARAMETER:
         identifier   :  real_flow
         index domain :  (i,j) ;

    ENDSECTION  ;

    PROCEDURE
      identifier :  GetEdges
      body       :
        flow(e,i,j):= 0;
        for e in edge do
        for i in nodes do
        if M(i,e) = -1 then
        for j in nodes do
        if M(j,e) = 1 then
        flow(e,i,j):= x(e); break; endif; endfor; break; endif;

        endfor;
```

```
        endfor;

        real_flow(i,j):=sum(e,flow(e,i,j));

    ENDPROCEDURE  ;

  ENDSECTION Edges ;

  PROCEDURE
    identifier :  MainInitialization

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainExecution
    body       :
      solve Least_cost_transport_plan;

      if ( Least_cost_transport_plan.ProgramStatus <> 'optimal' ) then
               empty z;
           endif;

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainTermination
    body       :
      return DataManagementExit();

  ENDPROCEDURE  ;

ENDMODEL Main_Real_data02 ;
```

MODEL 5.3: Real_data04

```
MAIN MODEL Main_Real_data04

  DECLARATION SECTION

    SET:
       identifier   :  nodes
       indices      :  i, j ;

    SET:
       identifier   :  coor
       index        :  k ;

    SET:
       identifier   :  i_cities
       subset of    :  nodes
       index        :  Icit ;

    SET:
       identifier   :  i_landfills
       subset of    :  nodes
       index        :  Ilan ;

    SET:
       identifier   :  i_spalovna
       subset of    :  nodes
       index        :  Ispa ;

    PARAMETER:
       identifier   :  node_coor
       index domain :  (i,k) ;

    PARAMETER:
       identifier   :  M
       index domain :  (i,j) ;

    PARAMETER:
       identifier   :  t
       index domain :  (i) ;

    PARAMETER:
       identifier   :  d
       index domain :  (i,j)
       definition   :  sqrt[sum(k, (node_coor(i,k) - node_coor(j,k))*
(node_coor(i,k) - node_coor(j,k)))] ;
```

```
    PARAMETER:
       identifier   :  c
       index domain :  (i,j)
       definition   :  M(i,j)*d(i,j) ;

    VARIABLE:
       identifier   :  x
       index domain :  (i,j)
       range        :  nonnegative ;

    VARIABLE:
       identifier   :  z
       range        :  free
       definition   :  sum[(Icit,Ispa), M(Icit,Ispa) * x(Icit,Ispa) * 100]
       -sum[(Icit,j),500*c(Icit,j)*x(Icit,j)]+pen1*100-pen2*1000;

    VARIABLE:
       identifier   :  pen1
       range        :  nonnegative
       definition   :  sum((Icit,Ilan), x(Icit,Ilan)*M(Icit,Ilan)) ;

    VARIABLE:
       identifier   :  pen2
       range        :  nonnegative
       definition   :  sum(Icit, [t(Icit) + sum(Icit2, M(Icit2,Icit)
*x(Icit2,Icit))] - sum(j, x(Icit,j)*M(Icit,j))) ;

    CONSTRAINT:
       identifier   :  r1
       index domain :  Icit
       definition   :  sum(j, M(Icit,j)*x(Icit,j)) <= t(Icit)   ;

    CONSTRAINT:
       identifier   :  r2
       index domain :  (Ispa)
       definition   :  sum(Icit, M(Icit,Ispa)*x(Icit,Ispa)) <= t(Ispa)  ;

    MATHEMATICAL PROGRAM:
       identifier   :  Maximize_profit
       objective    :  z
       direction    :  maximize
       constraints  :  AllConstraints
       variables    :  AllVariables
       type         :  Automatic ;

    PARAMETER:
```

```
      identifier   :  Spa_count
      index domain :  (Ispa)
      definition   :  sum(i, x(i,Ispa)) ;

  ENDSECTION  ;

  SECTION Excel_Input


    DECLARATION SECTION

      STRING PARAMETER:
         identifier :  WorkbookName
         definition :  "Real_data.xlsx" ;

      PARAMETER:
         identifier :  ExcelStatus ;

      STRING PARAMETER:
         identifier :  ExcelErrorMessage ;

      ELEMENT PARAMETER:
         identifier :  err
         range      :  errh::PendingErrors ;

    ENDSECTION  ;

    PROCEDURE
      identifier :  ReadDataFromExcel
      body       :
        Spreadsheet::SetActiveSheet( WorkbookName, "uzly" );

        Spreadsheet::RetrieveSet( WorkbookName, nodes, "A1:A330" );
        Spreadsheet::RetrieveSet( WorkbookName, i_cities, "A1:A206" );
        Spreadsheet::RetrieveSet( WorkbookName, i_landfills, "A207:A320");
        Spreadsheet::RetrieveSet( WorkbookName, i_spalovna, "A321:A330" );
        Spreadsheet::RetrieveSet( WorkbookName, coor, "E332:F332" );

        Spreadsheet::RetrieveParameter( WorkbookName, t, "D1:D330" );

        Spreadsheet::RetrieveTable( WorkbookName, node_coor, "node_coor",
"A1:A330", "E332:F332");
        Spreadsheet::RetrieveTable( WorkbookName, c_coor, "city_coor",
"A1:A206", "E332:F332");
        Spreadsheet::RetrieveTable( WorkbookName, l_coor, "landfill_coor",
"A207:A320", "E332:F332");
        Spreadsheet::RetrieveTable( WorkbookName, s_coor, "s_coorRange",
```

```
"A321:A330", "E332:F332");

        Spreadsheet::SetActiveSheet( WorkbookName, "incid" );

        Spreadsheet::RetrieveSet( WorkbookName, edge, "B1:CBK1" );
        Spreadsheet::RetrieveTable( WorkbookName, M1, "matrix_M", "A2:A331",
"B1:CBK1");

    ENDPROCEDURE  ;

  ENDSECTION Excel_Input ;

  SECTION Incidence_Matrix


    DECLARATION SECTION

      SET:
         identifier   :  edge
         index        :  e ;

      SET:
         identifier   :  i_spalovna2
         subset of    :  nodes
         index        :  Ispa2 ;

      SET:
         identifier   :  i_spalovna3
         subset of    :  nodes
         index        :  Ispa3 ;

      SET:
         identifier   :  i_cities2
         subset of    :  nodes
         index        :  Icit2 ;

      PARAMETER:
         identifier   :  M1
         index domain :  (i,e) ;

      PARAMETER:
         identifier   :  M2
         index domain :  (e,i,j) ;

    ENDSECTION  ;

    PROCEDURE
```

```
    identifier :  GetIncicenceMatrix
    body       :
      M2(e,i,j):= 0;
      for e in edge do
      for i in nodes do
      if M1(i,e) = 1 then
      for j in nodes do
      if M1(j,e) = -1 then
      M2(e,i,j):= M1(i,e); break; endif; endfor; break; endif;

      endfor;

      endfor;

      M(i,j):=sum(e,M2(e,i,j));

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  AddAndEraseEdges
    body       :
      M(Icit,Ispa2):=1;
      M(i,Ilan) := M(Ilan,i);
      M(Ispa2,j):=0;
      M(Ilan,j):=0;

  ENDPROCEDURE  ;

ENDSECTION Incidence_Matrix ;

SECTION Graphic_Output


  DECLARATION SECTION

    PARAMETER:
       identifier   :  c_coor
       index domain :  (Icit,k) ;

    PARAMETER:
       identifier   :  l_coor
       index domain :  (Ilan,k) ;

    PARAMETER:
       identifier   :  s_coor
       index domain :  (Ispa,k) ;
```

```
    ENDSECTION  ;

  ENDSECTION Graphic_Output ;

  PROCEDURE
    identifier :  MainInitialization

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainExecution
    body       :
      solve Maximize_profit;

      if ( Maximize_profit.ProgramStatus <> 'optimal' ) then
                empty z;
           endif;

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainTermination
    body       :
      return DataManagementExit();

  ENDPROCEDURE  ;

ENDMODEL Main_Real_data04 ;
```

MODEL 5.4: Real_data05

```
MAIN MODEL Main_Real_data05

  DECLARATION SECTION

    SET:
       identifier   :  nodes
       indices      :  i, j ;

    SET:
       identifier   :  coor
       index        :  k ;

    SET:
       identifier   :  edge
       index        :  e ;

    SET:
       identifier   :  i_cities
       subset of    :  nodes
       index        :  Icit ;

    SET:
       identifier   :  i_landfills
       subset of    :  nodes
       index        :  Ilan ;

    SET:
       identifier   :  i_spalovna
       subset of    :  nodes
       index        :  Ispa ;

    PARAMETER:
       identifier   :  node_coor
       index domain :  (i,k) ;

    PARAMETER:
       identifier   :  M
       index domain :  (i,e) ;

    PARAMETER:
       identifier   :  t
       index domain :  (i) ;

    PARAMETER:
       identifier   :  c
```

```
      index domain :  (e) ;

    VARIABLE:
      identifier   :  x
      index domain :  (e)
      range        :  nonnegative ;

    VARIABLE:
      identifier   :  z
      range        :  free
      definition   :  sum((Ispa,e), M(Ispa,e)*x(e)*300) - sum(e, 5*c(e)*x(e))
- sum(Ilan, pen1(Ilan)*10) - sum(Icit, pen2(Icit)*1000);

    VARIABLE:
      identifier   :  pen1
      index domain :  (Ilan)
      range        :  nonnegative
      definition   :  sum(e, x(e)*M(Ilan,e)) ;

    VARIABLE:
      identifier   :  pen2
      index domain :  (Icit)
      range        :  nonnegative
      definition   :  t(Icit)+sum(e, M(Icit,e)*x(e)) ;

    CONSTRAINT:
      identifier   :  r1
      index domain :  (Icit)
      definition   :  sum(e, M(Icit,e)*x(e)) <= t(Icit) ;

    CONSTRAINT:
      identifier   :  r2
      index domain :  (Ispa)
      definition   :  sum(e, M(Ispa,e)*x(e)) <= t(Ispa)  ;

    MATHEMATICAL PROGRAM:
      identifier   :  Maximize_profit
      objective    :  z
      direction    :  maximize
      constraints  :  AllConstraints
      variables    :  AllVariables
      type         :  Automatic ;

  ENDSECTION  ;

  SECTION Excel_Input
```

```
    DECLARATION SECTION

      STRING PARAMETER:
         identifier :  WorkbookName
         definition :  "Real_data.xlsx" ;

      PARAMETER:
         identifier :  ExcelStatus ;

      STRING PARAMETER:
         identifier :  ExcelErrorMessage ;

      ELEMENT PARAMETER:
         identifier :  err
         range      :  errh::PendingErrors ;

    ENDSECTION  ;

    PROCEDURE
      identifier :  ReadDataFromExcel
      body       :
        Spreadsheet::SetActiveSheet( WorkbookName, "uzly" );

        Spreadsheet::RetrieveSet( WorkbookName, nodes, "A1:A330" );
        Spreadsheet::RetrieveSet( WorkbookName, i_cities, "A1:A206" );
        Spreadsheet::RetrieveSet( WorkbookName, i_landfills, "A207:A320" );
        Spreadsheet::RetrieveSet( WorkbookName, i_spalovna, "A321:A330" );
        Spreadsheet::RetrieveSet( WorkbookName, coor, "E332:F332" );

        Spreadsheet::RetrieveParameter( WorkbookName, t, "D1:D330" );

        Spreadsheet::RetrieveTable( WorkbookName, node_coor, "node_coor",
"A1:A330", "E332:F332");
        Spreadsheet::RetrieveTable( WorkbookName, l_coor, "landfill_coor",
"A207:A320", "E332:F332");
        Spreadsheet::RetrieveTable( WorkbookName, s_coor, "s_coorRange",
"A321:A330", "E332:F332");

        Spreadsheet::SetActiveSheet( WorkbookName, "incid" );

        Spreadsheet::RetrieveSet( WorkbookName, edge, "B1:CAP1" );
        Spreadsheet::RetrieveTable( WorkbookName, M, "new_matrix2", "A2:A331",
"B1:CAP1" );
        Spreadsheet::RetrieveParameter( WorkbookName, c, "B333:CAP333" );

    ENDPROCEDURE  ;
```

```
 ENDSECTION Excel_Input ;

 SECTION Distance

   DECLARATION SECTION

     PARAMETER:
        identifier   :  d
        index domain :  (i,j)
        definition   :  sqrt(sum(k, (node_coor(i,k)-node_coor(j,k))
* (node_coor(i,k)-node_coor(j,k))))  ;

   ENDSECTION  ;

 ENDSECTION Distance ;

 SECTION Get_Edges

   DECLARATION SECTION

     PARAMETER:
        identifier   :  flow
        index domain :  (e,i,j) ;

     PARAMETER:
        identifier   :  real_flow
        index domain :  (i,j) ;

   ENDSECTION  ;

   PROCEDURE
     identifier :  GetEdges
     body       :
       flow(e,i,j):= 0;
       for e in edge do
       for i in nodes do
       if M(i,e) = -1 then
       for j in nodes do
       if M(j,e) = 1 then
       flow(e,i,j):= x(e); break; endif; endfor; break; endif;

       endfor;

       endfor;

       real_flow(i,j):=sum(e,flow(e,i,j));
```

```
    ENDPROCEDURE  ;

  ENDSECTION Get_Edges ;

  SECTION Graphic_Output

    DECLARATION SECTION

      PARAMETER:
         identifier   :  s_coor
         index domain :  (Ispa,k) ;

      PARAMETER:
         identifier   :  l_coor
         index domain :  (Ilan,k) ;

    ENDSECTION  ;

  ENDSECTION Graphic_Output ;

  PROCEDURE
    identifier :  MainInitialization

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainExecution
    body       :
      solve Maximize_profit;

      if ( Maximize_profit.ProgramStatus <> 'optimal' ) then
               empty z;
           endif;

  ENDPROCEDURE  ;

  PROCEDURE
    identifier :  MainTermination
    body       :
      return DataManagementExit();

  ENDPROCEDURE  ;

ENDMODEL Main_Real_data05 ;
```