VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MATEMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF MATHEMATICS

MATEMATICKÝ MODEL DOPRAVNÍ ÚLOHY PRO OBLAST ODPADOVÉHO HOSPODÁŘSTVÍ

MATHEMATICAL MODEL FOR TRANSPORTATION PROBLEM IN WASTE MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VÍT PROCHÁZKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN PAVLAS, Ph.D.
SUPERVISOR

BRNO 2012

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav matematiky
Akademický rok: 2011/2012

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Vít Procházka

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:

Matematický model dopravní úlohy pro oblast odpadového hospodářství

v anglickém jazyce:

Mathematical model for Transportation Problem in Waste Management

Stručná charakteristika problematiky úkolu:

V práci bude aplikováno matematické programování pro oblast odpadového hospodářství. Jedná se o dopravní úlohu, která bude mít za cíl minimalizovat finanční náklady na svoz směsného komunálního odpadu do místa jeho energetického využití. Optimální návrh svozové oblasti a dopravní trasy (hlavní výstup úlohy) bude ovlivněn ekonomikou klíčových prvků v systému.

Cíle bakalářské práce:

Seznámení se základními modely matematického programování. Vytvoření dopravní úlohy a její praktické použití na základě reálných vstupních údajů, které budou poskytnuty. Úloha bude implementována a řešena v programu GAMS. Dále se pro snadnější zadávání vstupních parametrů vytvoří uživatelské rozhraní v MS Excel.

Seznam odborné literatury:

[1] BAZARAA, M. S., JARVIS, J. J.: Linear Programming and Network Flows, first edition, John Wiley & Sons, Inc., New York, 1977, ISBN 0-471-06015-1.
[2] KLAPKA, J., DVOŘAK, J., POPELA, P.: Metody operačního výzkumu, druhé vydání, Nakladatelství VUTIUM, Brno, 2001.
[3] Ucekaj, V. Analýza možností nakládání s komunálními odpady v rámci mikroregionu. Disertační práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2010. 153 s.

Vedoucí bakalářské práce: Ing. Martin Pavlas, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2011/2012.

V Brně, dne 24.11.2011

L.S.

prof. RNDr. Josef Šlapal, CSc. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

Abstrakt

Tato bakalářská práce se zabývá sestavením matematického modelu dopravní úlohy pro oblast odpadového hospodářství, který slouží k analýze současného problému ČR nakládání se směsným komunálním odpadem. V teoretické části jsou shrnuty základní matematické poznatky z teorie grafů a optimalizace, které v práci využíváme. Zaměřeno je především na lineární programování. V druhé části se práce zabývá vlastním modelem, který je poté implementován do programu GAMS. Dále je ukázáno zpracování výsledků včetně grafické vizualizace v programu ArcGIS.

Summary

This bachelor's thesis deals with the development of the mathematical model of transportation problem in waste management, which is used for analysis of current issue of mixed municipal waste treatment in the Czech Republic. The theoretical part of thesis aims to introduce basic terms of graph theory and optimization with the focus on linear programming. The second part aims to development of the model and its implementation to GAMS. And next, the post-processing of results is shown, vizualization in ArcGIS included.

Klíčová slova

dopravní úloha, optimalizace, lineární programování, směsný komunální odpad, odpadové hospodářství

Keywords

transportation problem, optimization, linear programming, mixed municipal solid waste, waste-management

PROCHÁZKA, V. *Matematický model dopravní úlohy pro oblast odpadového hospodářství.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012. 43 s. Vedoucí bakalářské práce Ing. Martin Pavlas, Ph.D.

Prohlašuji, že jsem bakalářskou práci *Matematický model dopravní úlohy pro oblast odpadového hospodářství* vypracoval samostatně pod vedením Ing. Martina Pavlase, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Vít Procházka

Děkuji Ing. Martinu Pavlasovi, PhD. za příkladné vedení této práce. Můj velký dík patří Ing. Radku Šomplákovi za poskytnutý čas a rady během konzultací této práce. Rovněž děkuji RNDr. Pavlu Popelovi, PhD. za mnohé postřehy a připomínky k práci a pánům Mgr. Vítu Pásztovi a Mgr. Lukáši Markovi za zasvěcení do práce se softwarem ArcGIS během stáže na Univerzitě Palackého v Olomouci konané v rámci projektu AMathNet.

Na posledním řádku, avšak v neposlední řadě, děkuji rodině a přátelům za toleranci a podporu (nejen) během psaní této práce.

Obsah

1. Úvod

V roce 1886 napsal Jan Neruda, český spisovatel, básník a novinář, patrně svůj nejznámější fejeton *Kam s ním?*. V tomto krátkém díle se autor s humorem a nadsázkou zamýšlí nad problémem, co si počít se starým slamníkem na vyhození.

Podobný problém, avšak v oblasti odpadového hospodářství v rámci celého území ČR, je řešen v této práci. Zabýváme se sestavením modelu dopravní úlohy, který se na onu Nerudovu otázku snaží odpovědět za pomoci optimalizačních metod. Cílem je najít způsob, jak naložit s odpadem produkovaným na území obce. Proč je tato úloha aktuální a z jaké potřeby vychází, je blíže rozepsáno v části 1.2. této kapitoly. Nejprve si ale uvedeme některé základní pojmy, které nás celou prací provází. Při jejich popisu čerpáme z [1].

1.1. Základní pojmy

Směsný komunální odpad (SKO) a jeho frakce

Komunální odpad je veškerý odpad vznikající na území obce při činnosti fyzických osob, s výjimkou odpadů vznikajících u právnických osob nebo fyzických osob oprávněných k podnikání [2]. *Směsný komunální odpad* (SKO) představuje netříděnou složku komunálního odpadu pocházející převážně z domácností. Tento odpad je obvykle ukládán a sbírán prostřednictvím běžně známých kontejnerů určených právě na sběr SKO.

Průměrné hmotnostní zastoupení jednotlivých složek SKO je následující (čerpáno z [3]):

- papír a lepenka 13%
- plasty 12%
- textil 7%
- sklo 4%
- kovy 2%
- ostatní biologicky rozložitelné složky 25%
- nápojové kartony 2%
- ostatní (minerální, nebezpečný, dřevo,..) 35%

SKO je možné ve speciálních třídicích linkách rozdělit na tzv. lehkou frakci (LF), těžkou frakci (TF), podsítnou frakci (PF) a kovy. Do LF patří většina papírů, plastů, PET lahví, nápojových kartonů a textilu. Do TF přechází většina skla. V PF je obsažena většina kuchyňského odpadu, tedy se jedná převážně o biologicky rozložitelný odpad. Kovy řadíme zvlášť.

Mechanicko-biologická úprava (MBÚ)

Mechanicko-biologická úprava (MBÚ) je proces zpracovávání SKO, který je považován za alternativu ke skládkování a spalování. Spočívá právě v principu roztřídění SKO do jeho frakcí, s kterými je možné dále samostatně nakládat. Z tohoto hlediska je důležitá LF, která je nejvýhřevnější částí SKO, a proto se dále využívá k energetickým účelům (náhrada uhlí v existujicích elektrárnách a teplárnách). Hmotnostní podíl LF v SKO je přibližně 40%.

Jak už název technologie napovídá, jedná se o více stupňů zpracování SKO. Nejprve probíhá třídění odpadu v mechanické části, poté se biologicky rozložitelná složka SKO (PF) upravuje v biologické části MBÚ. Důležité je, že podstatná část výstupu z MBÚ končí opět na skládkách (cca 50%).

Energetické využití SKO (EVO)

Další možností, jak naložit s SKO, je jeho termické zpracování. Děje se tak v zařízeních k tomu určených (spalovnách), kde probíhá nejprve nutná manipulace s SKO (jeho homogenizace) a následně spalování, při kterém se energie uvolněná z odpadu využívá k výrobě páry. Ta následně slouží k výrobě elektrické energie. Další užitek je výroba tepla, které může být dodáváno odběratelům např. formou napojení na rozvodnou síť teplárny. Mluvíme tedy o energetickém využití odpadů. V současné době se v ČR nachází tři zařízení EVO (Praha, Brno, Liberec).

Skládky

Skládkování je nejstarší a technologicky nejméně náročný způsob nakládání s SKO. V současné době se v ČR skládkuje 88% SKO [3].

1.2. Motivace

Česká republika se ke dni 1. ledna 2013 na základě směrnice Rady evropské unie 1999/31/ES (tzv. skládkovací směrnice), implementované vyhláškou č. 294/2005 Sb., zavázala do roku 2020 odklonit od skládkování 65% biodegradabilního materiálu (tvoří cca 50% hmotnosti SKO) v porovnání s množstvím ukládaným v roce 1995. Jako jediná možnost, jak splnit legislativní požadavky EU na odklonění od skládkování, se jeví výstavba nových zařízení EVO a MBÚ.

Podstatným faktorem jak pro rozhodnutí o počáteční investici a zvážení finanční udržitelnosti projektu, tak pro navržení důležitých parametrů těchto zařízení (např. zpracovatelská kapacita), je odhad množství SKO, které se bude svážet z obcí do jednotlivých zařízení.

A právě cílem této práce je vytvoření nástroje k možné analýze této situace. Konkrétně se jedná o dva optimalizační modely (viz kapitolu 7) vytvořené v programu GAMS, které mají každý jiný účel, ale oba vychází ze stejného

předpokladu, a sice zajištění minimálních nákladů obcí na zpracování odpadu. Tyto náklady můžeme rozdělit do dvou skupin: náklady na přepravu a náklady za zpracování, tzv. poplatky na bráně u jednotlivých zařízení.

Součástí práce je ukázka dalšího zpracování výsledků výpočtů pomocí vytvořených matematických modelů, včetně grafické vizualizace situace, k níž využijeme program pro tvorbu kartografických výstupů ArcGIS. Tuto ukázku provedeme na datech, která byla poskytnuta Ústavem procesního a ekologického inženýrství VUT v Brně (dále pouze ÚPEI).

V dalších třech kapitolách si nejprve uvedeme matematický aparát, který pro řešení úlohy používáme, a představíme software GAMS a jeho možnosti. Další kapitoly se už věnují vlastní tvorbě modelů a jejich popisům, přípravě dat, která do modelů vstupují, a zpracování výsledků.

2. Teorie grafů

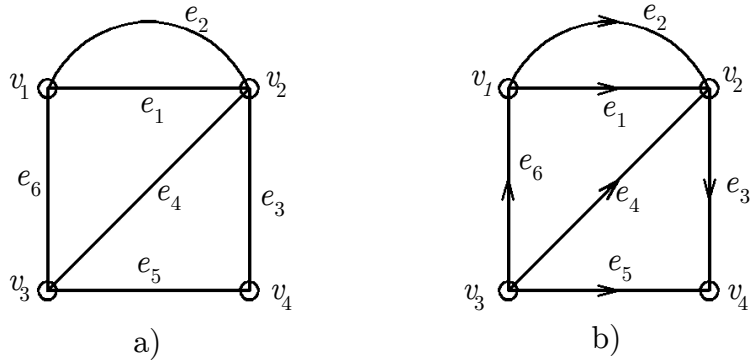
Teorie grafů je matematická disciplína, která zkoumá struktury zvané *grafy*. Za jejího zakladatele je považován Leonhard Euler, který roku 1736 řešil úlohu sedmi mostů města Královce. Od té doby je teorie grafů užívána v mnoha aplikacích pro řešení nejrůznějších úloh.

V této kapitole představíme některé základní definice a vlastnosti, které budeme dále využívat v naší úloze. Při jejich formulaci čerpáme z [6].

2.1. Definice. *Neorientovaný graf* je trojice $G = (V, E, \varepsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinu E, jejíž prvky nazýváme *neorientovanými hranami*, a zobrazením $\varepsilon : E \to V^2$, které nazýváme *vztahem incidence*. Toto zobrazení přiřazuje každé hraně $e \in E$ jedno nebo dvouprvkovou množinu vrcholů.

2.2. Definice. *Orientovaný graf* je trojice $G = (V, E, \varepsilon)$ tvořená neprázdnou konečnou množinou V, jejíž prvky nazýváme *vrcholy*, konečnou množinu E, jejíž prvky nazýváme *orientovanými hranami*, a zobrazením $\varepsilon : E \to V^2$, které nazýváme *vztahem incidence*. Toto zobrazení přiřazuje každé hraně $e \in E$ uspořádanou dvojici vrcholů (x, y). Prvý z nich, x, nazýváme *počátečním vrcholem hrany* e a y nazýváme *koncovým vrcholem hrany* e. Oba vrcholy (x, y) také souhrnně nazýváme *krajními vrcholy hrany* e. Jestliže pro nějakou hranu e je $x = y$, pak hranu e nazýváme (orientovanou) *smyčkou*.

Grafy můžeme názorně zakreslit. Obvykle vrcholy kreslíme jako body (kroužky) a hrany jako čáry spojující příslušné dvojice vrcholů. Je-li hrana orientovaná, značíme orientaci šipkou od počátečního ke koncovému vrcholu.



Obr. 2.1. znázorňující a) neorientovaný graf b) orientovaný graf

V četných aplikacích grafy samotné nepostačují k adekvátnímu popisu situace. Proto se často ke hranám či vrcholům přidávají nějaké hodnoty (obvykle číselné), které reprezentují např. doby trvání nebo náklady činností, propustnosti potrubí,

pravděpodobností události apod. V naší úloze se bude jednat o vzdálenosti mezi vrcholy.

2.3. Definice. Graf, jehož hrany nebo vrcholy jsou opatřeny číselnými hodnotami, nazýváme *ohodnoceným grafem* nebo též *sítí*.

2.4. Definice. Posloupnost vrcholů a hran v_0, e_1, v_1, e_2, v_2,...,e_n, v_n nazýváme (*neorientovaný*) *sled*, jestliže každá hrana e_i této posloupnosti spojuje vrcholy v_{i-1} a v_i .

Jestliže rozlišujeme počáteční vrchol v_{i-1} a koncový vrchol v_i , hovoříme o *orientovaném sledu*. Ve sledu se vrcholy i hrany mohou opakovat.

2.5. Definice. Orientovaný (neorientovaný) sled, v němž se žádná hrana neopakuje, nazýváme *orientovaným* (*neorientovaným*) *tahem*.

Orientovaný (neorientovaný) sled, v němž se neopakuje žádný vrchol, nazýváme *orientovanou* (*neorientovanou*) *cestou*.

V ohodnocených grafech je často smysluplné hovořit o součtu ohodnocení hran v nějakém sledu (cestě, tahu). Tento součet nazýváme *délka sledu* (*cesty*, *tahu*). V našem případě, kdy ohodnocení hran představuje vzdálenosti mezi vrcholy, které spojuje, má tento termín význam celkové vzdálenosti mezi vrcholy v_0 a v_n ve sledu (cestě, tahu).

Při výpočtech s grafy budeme využívat tzv. incidenční matici, kterou definujeme tímto způsobem:

2.6. Definice. Buď G orientovaný graf bez smyček. Zvolíme-li pořadí vrcholů v_1,..., v_n a pořadí hran e_1,..., e_m , můžeme grafu G přiřadit *matici incidence* (též *incidenční matici*) A typu $n \times m$ předpisem

$$a_{ij} \begin{cases} 1, \text{ jestliže } v_i \text{ je koncový vrchol hrany } e_j \\ -1, \text{ jestliže } v_i \text{ je počáteční vrchol hrany } e_j \\ 0, \text{ jinak} \end{cases}$$

2.1. Příklad. Incidenční matice grafu na obr. 2.1. b) vypadá následovně:

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

3. Optimalizace

V naší práci se zabýváme sestavením dopravního modelu - konkrétně modelu svozu směsného komunálního odpadu (SKO). Budeme předpokládat, že cílem producenta (obce) je minimalizovat celkové náklady na zpracování SKO (zpracovatelský poplatek + náklady na dopravu). A právě hledáním minima funkce (obecně extrémů) za určitých podmínek se zabývá optimalizace.

Optimalizační úlohu můžeme obecně zapsat takto:

$$\begin{array}{ll} \text{minimalizuj} & f(\mathbf{x}) \\ \text{za podmínek} & \boldsymbol{g}(\mathbf{x}) \leq 0\ , \end{array} \tag{3.1}$$

kde $\mathbf{x} = (x_1, x_2, .., x_n)^T$ je vektor dimenze n; x_j , $j = 1, 2, ..., n$ jsou tzv. rozhodovací proměnné (dále pouze proměnné),
$f : \mathbb{R}^n \to \mathbb{R}$ je funkce těchto proměnných, kterou nazýváme účelová funkce neboli kritérium,
funkce $g_1 : \mathbb{R}^n \to \mathbb{R}, ..., g_m : \mathbb{R}^n \to \mathbb{R}$ pak nazveme omezujícími podmínkami neboli omezeními (souhrnně zapisujem $\boldsymbol{g}(\mathbf{x})$).

Pozn.: Je zřejmé, že se v praxi setkáme i s případem, kdy chceme účelovou funkci maximalizovat. Jednoduchou transformací: $\max f(\boldsymbol{x}) = \min(-f(\boldsymbol{x}))$ však můžeme tuto situaci převést na úlohu, ve které hledáme minimum funkce. Podobně je možné omezující podmínky v jiném než uvedeném tvaru převést na tento tvar (viz [7]):

např. $g^*(\mathbf{x}) > c$ na $g(\mathbf{x}) \leq 0$, kde $g(\mathbf{x}) = -g^*(\mathbf{x}) + c$
nebo $g(\mathbf{x}) = 0$ na $g(\mathbf{x}) \leq 0 \wedge -g(\mathbf{x}) \leq 0$

3.1. Definice. Množina $X = \{\mathbf{x} \mid g_1(\mathbf{x}) \leq 0, \ ... \ , g_m(\mathbf{x}) \leq 0\}$ se nazývá *množina přípustných řešení* (MPŘ), její prvky $\mathbf{x} \in X$ pak *přípustná řešení* úlohy (3.1)

3.2. Definice. Řekneme, že funkce $f : \mathbb{R}^n \to \mathbb{R}$ má v bodě x_0 *ostré lokální minimum*, jestliže existuje ryzí okolí $\overline{O}(x_0)$ tak, že $\forall x \in \overline{O}(x_0)$ je

$$f(x) - f(x_0) > 0.$$

3.3. Definice. Řekneme, že funkce $f : \mathbb{R}^n \to \mathbb{R}$ má v bodě x_0 *ostré globální minimum*, jestliže $\forall x \in \text{Dom } f$, $(x \neq x_0)$ platí

$$f(x) - f(x_0) > 0.$$

Pozn. Při definici *neostrého minima* (*globálního* i *lokálního*) změníme $>$ na $\geq$.

3.4. Definice. Množinu $S \subset \mathbb{R}^n$ nazveme *konvexní množinou*, jestliže pro

libovolné dva body $\mathbf{x}$, $\mathbf{y} \in S$ a pro libovolné $\alpha \in (0,1)$ platí

$$\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \in S$$

Deifinice 3.1., 3.4. čerpáme z [5], difinice 3.2. a 3.3. z [8]

Podle vlastností účelové funkce f, omezeních $\boldsymbol{g}$ a proměnných $\mathbf{x}$ rozdělujeme optimalizační úlohy na úlohy:

- *lineárního programování* (LP) – funkce f a $\boldsymbol{g}$ jsou v lineárním tvaru
- *nelineárního programování* (NLP) – jestliže alespoň jedna z funkcí f, $\boldsymbol{g}_j$, $j = 1,..., m$ je v nelinárním tvaru
- *celočíselného programování* (MIP) – když požadujeme, aby alespoň jedna z proměnných x_i byla z množiny celých čísel.

3.1. Lineární programování

O úloze lineárního programování (LP) hovoříme, jestliže účelová funkce i všechny omezující podmínky jsou lineárními výrazy.

Úlohu je možno zapsat ve tvaru

$$\begin{aligned} &\min && \mathbf{c}^T\mathbf{x} \\ &\text{za podmínek} && \mathbf{A}\mathbf{x} - \mathbf{b} \leq 0\ , \end{aligned}$$

kde A je matice typu $m \times n$ a její složky a_{ij} , $i = 1, 2,..., m$ $j = 1, 2,..., n$ jsou konstanty, $\mathbf{x} = (x_1, x_2,.., x_n)^T$ je sloupcový vektor s n řádky a $\mathbf{b} = (b_1, b_2,.., b_m)^T$ je sloupcový vektor, jehož prvky jsou konstanty.

Pozn. Úlohy LP se obvykle formulují tak, aby obsahovaly podmínky nezápornosti všech proměnných, tj. $x_j \geq 0$, $j = 1,2,..., n$

3.5. Definice. *Konvexní polyedrická množina* $M \subset \mathbb{R}^n$ je taková množina, kterou lze vyjádřit jako průnik konečného počtu uzavřených poloprostorů. [5]
Konvexní polyedrická množina je speciálním případem konvexní množiny.

3.6. Definice. Nechť $S \subset \mathbb{R}^n$ je libovolná množina. Bod $\mathbf{s} \in S$ nazveme *krajním bodem* množiny S, jestliže neexistují body $\mathbf{x},\mathbf{y} \in S$ a číslo $\alpha \in (0,1)$ tak, že $\mathbf{x} \neq \mathbf{y}$ a $\mathbf{s} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y}$. [5]

3.1. Věta. *Konvexní polyedrická množina má konečný počet krajních bodů.* [5]

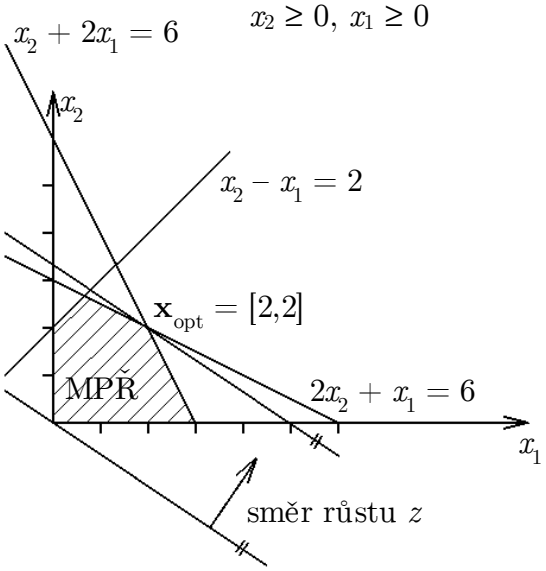
Tyto definice a věta mají v lineárním programování tento význam. Množina přípustných řešení lineární úlohy je konvexní polyedrická množina a dále pokud existuje optimální řešení úlohy, pak se nachází v krajním bodě. Tyto poznatky

využívá simplexová metoda, kterou si o pár řádků níže představíme.

Úlohy LP můžeme pro dvě proměnné řešit graficky. Tento postup si vysvětlíme na příkladu.

3.1. Příklad. Graficky vyřešte úlohu

$$\begin{aligned} \max\ z &= 2x_1 + 3x_2 \\ x_1 + 2x_2 &\le 6 \\ 2x_1 + x_2 &\le 6 \\ x_2 - x_1 &\le 2 \\ x_2 \ge 0,\ x_1 &\ge 0 \end{aligned}$$



3.1.1. Simplexová metoda

Simplexová metoda je nejpoužívanější metodou pro řešení úloh lineárního programování. Zde si vysvětlíme její základní myšlenku. V případě lineární úlohy je MPŘ konvexní polyedrická množina. Najdeme jeden krajní bod x_0 této množiny a zjistíme hodnotu účelové funkce v bodě x_0. Z tohoto bodu přejdeme do „sousedního“ krajního bodu tak, aby se hodnota účelové funkce zlepšila (v případě minimalizační úlohy byla menší). Jestliže z bodu x_0 vede neomezená hrana, na níž existuje bod, pro který je hodnota účelové funkce lepší než v bodě x_0, nemá úloha optimální řešení. Jestliže sousední krajní bod s lepší hodnotou účelové funkce neexistuje, pak bod x_0 je hledaným řešením. Pokud sousední krajní bod s lepší hodnotou účelové funkce existuje, budeme stejný postup jako na x_0 aplikovat na tento nový bod. Protože krajních bodů MPŘ existuje pouze konečný počet, končí úloha po konečném počtu iterací nalezením optimálního řešení nebo zjištěním, že daná úloha optimální řešení nemá.

Toto je základní myšlenka simplexové metody. Její přesný popis můžeme najít v [4], [5].

4. GAMS

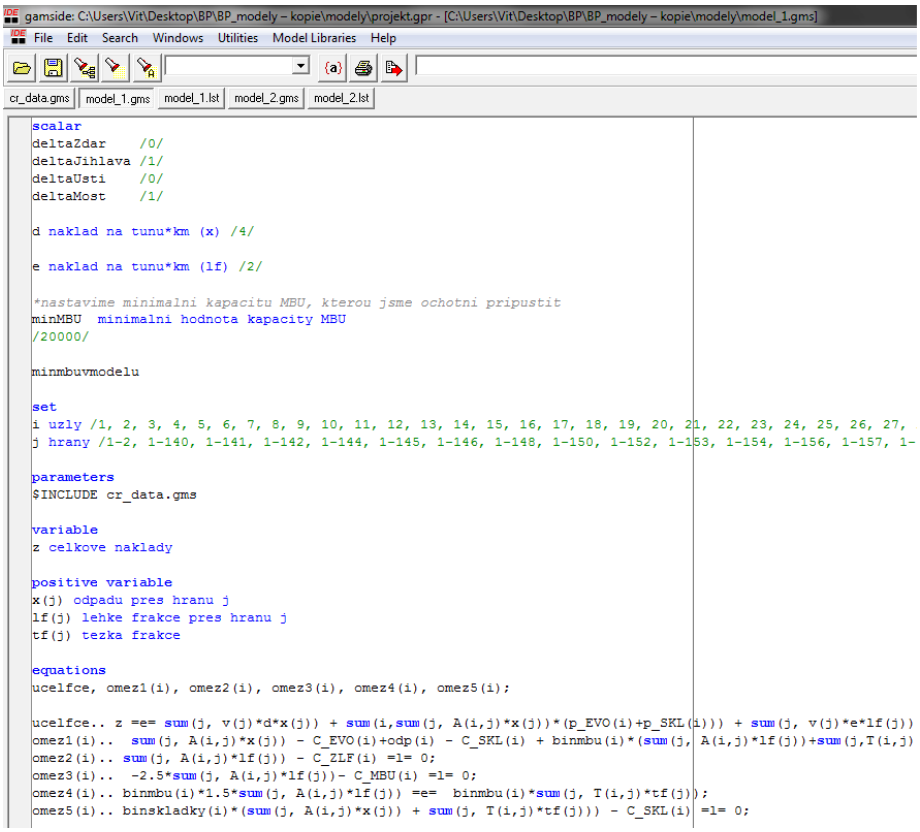
GAMS (The General Algebraic Modeling System) je pokročilý modelovací systém s vlastním programovacím jazykem určený pro řešení úloh matematického programování. Je zvláště vhodný pro rozsáhlé komplexní optimalizační úlohy. GAMS se skládá z kompilátoru jazyka a sady integrovaných řešičů pro různé typy optimalizačních úloh. Jazyk s jednoduchou syntaxí umožňuje i základní programovací techniky jako např. konstrukci cyklů, rozhodovacích podmínek a další.

Velkou výhodou GAMSu, která je využita v této práci, je možnost propojení s tabulkovým softwarem MS Excel. Takto činíme z důvodu práce s daty, která je mnohem snadnější v prostředí MS Excel, jenž je k tomu primárně určen. Načítání dat je blíže popsáno v kapitole IMPORT.

Pro zpracování výsledku využijeme schopnost GAMSu vypisovat výsledky do textového souboru, které budeme dále zpracovávat.

GAMS je možno zdarma stáhnout ze stránek www.gams.com a používat bez licenčních souborů s určitými omezeními na počty proměnných, omezujících podmínek atd. V naší úloze však tato omezení nesplňujeme, a tak pracujeme v licencované verzi softwaru.

Více o práci se softwarem GAMS uvádí [9].



Obr. 4.1. Ukázka prostředí GAMS

4.1. Řešiče

Důležitou součásti řešení úlohy je volba správného řešiče, který vybíráme podle typu úlohy. Volbu zapíšeme do GAMSu pomocí klíčového slova `option` (např. `option nlp = minos`). Seznam řešičů, které GAMS využívá, lze najít na [10], kde je ke každému z nich podrobná dokumentace obsahující popis metod a algoritmů, které daný řešič používá pro řešení úlohy.

V našich modelech využíváme řešič CPLEX, který si blíže představíme.

CPLEX

CPLEX je velmi stabilní řešič pro úlohy lineárního programování. Obsahuje v sobě několik algoritmů, jejichž základem je *simplexová metoda*, případně *duální simplexová metod*a. Jeho velkou předností je rychlost a schopnost vyřešit naprostou většinu úloh LP bez nutnosti uživatele zasahovat do základního nastavení řešiče. CPLEX s metodami řešení je podrobně rozebrán v [11].

4.2. Propojení s MS Excel

Jak jsme uvedli v úvodu této kapitoly, je pro rozsáhlejší úlohy vhodné využít importu dat z prostředí MS Excel než data zadávat přímo do GAMSu. Takto činíme z důvodu usnadnění práce – není zde nutnost ručního přepisování dat do kódu, lze snadněji upravovat data během používání modelu a je zajištěna lepší přehlednost kódu a nižší pravděpodobnost chyby při opětovné manipulaci s daty.

Dodejme, že stejně jako vstupy z prostředí MS Excel, umí GAMS exportovat i výstupní data do tohoto prostředí. To však v úloze nevyužíváme, a tak je zde rozebrána pouze etapa vstupu dat z MS Excel do softwaru GAMS. Čerpáno z [12].

Nástrojů, které umožňují propojení MS Excel a GAMSu, je více, my využijeme nástroj GDXXRW, který umožňuje načítání i zapisování dat z a do MS Excel. Tento nástroj vyžaduje nainstalovaný Microsoft Excel na počítači, na kterém je používán. Jeho užití si vysvětlíme na příkladu:

4.1. Příklad. Ukázka použití nástroje GDXXRW při načítání dvourozměrné proměnné:

```
A(i,j) incidencni matice
$CALL GDXXRW.EXE incid.xlsx par=A rng=mat
$GDXIN incid.gdx
$LOAD A
$GDXIN
```

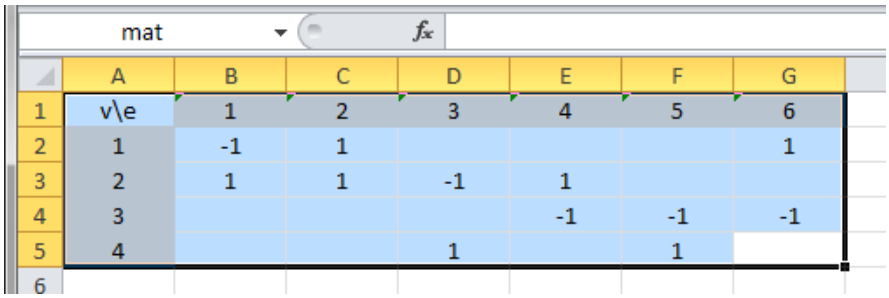
Předpokládáme, že máme v GAMSu definovánu množinu indexů i a j. Poté vytvoříme parametr `A(i,j)`, který pojmenujeme `incidencni matice`.

Řádkem `$CALL GDXXRW.EXE incid.xlsx par=A rng=mat` pak spustíme nástroj GDXXRW, který přečte oblast označenou *mat* (tu v souboru označíme včetně indexů, které se musí názvem shodovat s našimi definovanými indexy), v souboru *incid.xlsx* tím vytvoříme GDX (GAMS Data Exchange) soubor *incid.gdx*, do kterého se uloží data. Příkazem `$GDXIN incid.gdx` pak říkáme, že data budou čtena z tohoto souboru. K jejich načtení slouží příkaz `$LOAD`. Pro ukončení načítání zadáme závěrečný příkaz `$GDXIN`.

Na Obr. 4.2. vidíme soubor v MS Excel, který je možno takto načíst. Opět jsme pro ukázku zvolili incidenční matici z příkladu 2.2. Aby načtení proběhlo úspěšně, musíme v tomto případě mít v GAMSu nadefinovanou množinu indexů:

```
set
i uzly /1, 2, 3, 4/
j hrany /1, 2, 3, 4, 5, 6/
```

Všimněme si především oblasti *mat*, kterou označíme včetně indexů.



Obr. 4.2. Ukázka označení dat v MS Excel

5. Popis modelu

Neboť už máme nachystaný matematický aparát, který budeme pro sestavení modelu a jeho řešení používat, můžeme se pustit do rozboru prvků, ze kterých se model skládá. Model je představován orientovaným grafem, jehož prvky si nyní rozebereme.

5.1. Uzly

Uzly grafu v našem modelu značíme indexem i. V modelu mají obce, EVO, skládky i místa určená ke zpracování LF své vlastní uzly (na rozdíl od MBÚ, které samostatné uzly netvoří, „jsou umístěny“ v obcích, více v 5.1.3.). Jejich chování v modelu se liší podle hodnoty parametrů, které jsou uvedeny níže.

5.1.1. Obce

Z důvodu ušetření výpočtové práce neuvažujeme všechny obce ČR. Vybereme obce s rozšířenou působností (ORP), což jsou správní jednotky, které pokrývají všechny obce v ČR. Nachází-li se v území ORP město nad 10 000 obyvatel, bude v modelu uvažováno samostatně. Další odlišení oproti správnímu dělení na ORP je, že uzemí hlavního města Prahy je z důvodu reálné situace svozu SKO rozděleno na dvě části – levobřežní a pravobřežní. Takto vybraných obcí celkem uvažujeme 213.

Protože udávat název obce jako index i je v modelu dlouhé, nepřehledné a z důvodu české diakritiky při práci v různých programech obtížné, zavedeme si vlastní označení, a to jednoduše číslem. Pro obce tedy $i \in \{1, 2, ..., 213\}$ (např. pro Brno: $i = 22$).

Hlavním parametrem, který nás u obcí zajímá, je roční produkce SKO. Ten označíme o s příslušným indexem, tedy $o_i\,[\mathrm{t/rok}]$ je produkce SKO v uzlu i.

5.1.2. EVO

V modelu uvažujeme EVO samozřejmě v místech, kde už jsou postaveny: Brno, Praha (pravobřežní), Liberec a dále v lokalitách, které jsou vytipovány pro potenciální výstavbu nových zařízení EVO (tento výběr je proveden na základě konzultace s ÚPEI): Praha – Řeporyje (levobřežní), České Budějovice, Hradec Králové (Opatovice), Karviná, Ostrava, Přerov, Plzeň (Chotíkov), Mělník, Otrokovice, Most × Ústí nad Labem, Jihlava × Žďár nad Sázavou.

Obec v závorce znamená potenciální výstavbu v reálné situaci. V modelu však tuto obec nemáme zahrnutou, a proto uvažujeme nejbližší obec, s kterou v modelu pracujeme. Znak × znamená, že EVO se plánuje nejvýše v jedné z těchto obcí.

Ačkoliv v grafu mají EVO své vlastní uzly, vždy existuje obec v modelu, ke které

EVO přísluší. Uzly EVO označíme *s, kde * je index obce, ve které EVO uvažujeme (např. EVO v Brně: $i = 22s$).

Hlavním parametrem zařízení EVO je zpracovatelská kapacita, což je množství odpadu v tunách, které je EVO schopna ročně zpracovat. Značíme ji C^{EVO} s příslušným indexem i uzlu, tedy C_i^{EVO} [t/rok]. Druhým podstatným parametrem je tzv. cena na bráně, což je poplatek, který zaplatíme EVO za zpracování 1 tuny SKO. Značíme podobně jako u kapacity p_i^{EVO} [Kč·rok/t], i je index uzlu.

V implementaci modelu do prostředí GAMS využijeme pomocný parametr δ^{EVO}, který nabývá hodnoty 1, je-li uzel i EVO, 0 v ostatních případech. Slouží především k jednoduché formulaci podmínek, které nám zajistí práci pouze s uzly typu EVO.

5.1.3. Skládky

Skládky značíme *k, kde * je opět index obce, ke které skládka náleží. Podobně jako u EVO sledujeme u skládek dva parametry, a to kapacitu C_i^{SKL} [t/rok] a poplatek p_i^{SKL} [Kč·rok/t], který zaplatíme za uložení 1 tuny SKO na skládce i.

Posledním pomocným parametrem je δ_i^{SKL} , který nabývá hodnoty 1, jestliže uzel i je skládkou, 0 v ostatních případech.

5.1.4. MBÚ

Pro sestavení modelu je důležité vědět, že LF tvoří průměrně přibližně 40% hmotnosti SKO přivezeného do jednotky MBÚ. Zbylých 60% (TF, PF a kovy) nazveme souhrnně jen těžká frakce (dále TF), i když se tento pojem úplně neshoduje s tím, co značíme jako těžká frakce v průmyslové praxi (vysvětleno v kapitole 1.1) a budeme požadovat její uložení na skládky. Z tohoto důvodu je nutnou podmínkou, aby k obci, ve které uvažujeme MBÚ, příslušela skládka.

V našem modelu jednotky MBÚ na rozdíl od EVO nebo skládek nemají vlastní uzly. „Jsou umístěny“ do určitých obcí. Parametr, který určuje, jestli v dané obci uvažujeme MBÚ nebo ne, nazveme δ_i^{MBU} a nabývá hodnot 1, když v uzlu (obci) i uvažujeme MBÚ, 0 v opačném případě. Protože nevíme, kde budou MBÚ vystavěny, nastavíme zpočátku hodnotu δ_i^{MBU} na 1 u všech obcí, které mají skládku. Model I (viz 7.1) poté provádí podle výpočtů redukci MBÚ pouze na některé lokality přenastavením tohoto parametru na 0.

Abychom zajistili uložení TF na požadované skládce, sestrojíme další incidenční matici $B = (b_{ij})$, která umožňuje transport TF pouze po hranách vedoucích z obcí, kde se nachází MBÚ, do skládek příslušným těmto obcím ($b_{ij} = 1$ pouze, když j je typu *–*k a $i =$ *k, pro $i =$ * je $b_{ij} = -1$).

Další parametr je zpracovatelská kapacita MBÚ, značíme C_i^{MBU} [t/rok], která udává množství SKO, které je za rok jednotka MBÚ schopna zpracovat. Jednotky MBÚ jsou dimenzovány z důvodu technicko-ekonomického modelu [1] na 60 kt/rok.

Rovněž nás zajímá poplatek za zpracování 1 tuny SKO p_i^{MBU} [Kč·rok/t].

5.1.5. Zpracování lehké frakce (ZLF)

LF z MBÚ je možné využít v cementárnách, popřípadě v energetických jednotkách k tomu určených (stávající teplárny a elektrárny). Pro tato místa máme v modelu opět samostatné uzly souhrnně označené jako ZLF. Značíme je *z, kde * je index obce, ke které ZLF přísluší. V těchto místech máme taktéž kapacitní omezení C_i^{ZLF} [t/rok], což je maximální množství LF, které je možno v uzlu i ročně zpracovat. Neuvažujeme už však žádnou finanční bilanci za zpracování, ta je zahrnuta v poplatku MBÚ.

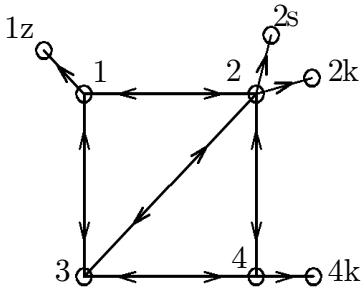
5.2. Hrany

Uvedené uzly jsou spojené hranami. Ty představují dopravní propojení, po kterých se pohybuje odpad (SKO nebo LF).

Jediným parametrem hran je v_j [km], což představuje délku hrany j neboli vzdálenost mezi uzly, které hrana spojuje. Vzdálenost v_j mezi obcemi vychází z reálné dopravní situace ČR, kdy se jedná o nejkratší cestu mezi obcemi, které hrana spojuje. V modelu uvažujeme orientovaný graf, ten tedy rozlišuje počáteční a koncový uzel hrany. Pro obce platí, že existuje-li hrana, která vede z jedné obce do druhé, existuje i hrana, která vede z druhé do první, přičemž hodnotu parametru v mají stejnou.

Jestliže hrana nespojuje dvě obce, ale obci s jiným typem uzlu, pro příklad uvažujme EVO (stejné je to pro skládku i ZLF), je v modelu použito triku, kdy hrana vede pouze z příslušné obce, ve kterém je EVO (skládkou, ZLF) uvažováno, do této jednotky EVO (skládky, ZLF). Délka hrany je přitom vždy nastavena na 1 km, což představuje zanedbatelné náklady. Tento trik zajišťuje lepší přehlednost ve výsledcích a zároveň odstraňuje nutnost použití dalších omezujících podmínek v modelu než by tomu bylo v případě, kdy bychom neuvažovali EVO, skládky a ZLF v samostatných uzlech. Popsaná situace je snadno pochopitelná z obrázku 5.1.

Hrany značíme a–b, kde a je index počátečního vrcholu, z kterého hrana vede, a b index koncového vrcholu hrany.



Obr. 5.1. Architektura sítě

6. Příprava dat

Jak je uvedeno ve 3. kapitole, data, s kterými budeme pracovat v GAMSu, budeme načítat z prostředí MS Excel, kde je možno s nimi lépe pracovat. Jedná se o parametry uvedené v předchozí kapitole (o_i , C_i^{EVO}, p_i^{EVO}, C_i^{SKL}, p_i^{SKL}, C_i^{MBU}, p_i^{MBU}, δ_i^{MBU}, δ_i^{EVO}, δ_i^{SKL}, C_i^{ZLF}) a o incidenční matici grafu, kterým popisujeme situaci. Ovšem ne všechna tato data máme v podobě, ve které je potřebujeme, a proto si je musíme extrahovat z různých dokumentů a dat, které nám poskytlo pracoviště ÚPEI.

Protože by ruční zpracovávání takto rozsáhlého souboru dat zabralo příliš mnoho času a bylo by velice snadné při něm udělat nějakou chybu, využijeme možnosti tvorby maker v MS Excel, což znamená vytvořit v programovacím jazyce VBA (Visual Basic for Applications) kód, který bude požadovanou činnost dělat za nás. Makra spouštíme na kartě Zobrazení/Makra ve verzi MS Excel 2010.

6.1. Hrany

K reprezentaci hran v modelu potřebujeme mít incidenční matici grafu a vzdálenosti jednotlivých hran. Máme k dispozici *tabulku vzdáleností* [viz obr. 6.1. a)], z které si tyto parametry musíme vytvořit. Přejdeme k systému označení uzlů, jak je popsáno v kapitole 5.1.

Poté postupujeme následovně:

- nejdříve symetricky překlopíme hodnoty (abychom vytvořili obousměrné cesty) pomocí makra Symetrie() (viz přílohu A). Tím vznikne tabulka z obr. 6.1. b).
- rozšíříme tabulku o uzly skládek, EVO a ZLF (ručně podle umístění) – taktéž do políčka, kde se kříží sloupec EVO, skládky či ZLF s řádkem obce, ke které přísluší, doplníme 1 (vzdálenost).
- Nyní už spustíme makro Incid() (viz přílohu A), které projde celou tabulku a pokud narazí na nenulovou položku, vytvoří příslušnou hranu a na další list sešitu, v němž pracuje, zaznamená délku (vzdálenost) této hrany.

Takto vzniklá incidenční matice, která nám reprezentuje dopravní infrastrukturu ČR, je typu 355×2094.

6.2. Uzly

Každý z parametrů, které nás zajímají u uzlů, má svůj excelovský sešit, kde ve sloupci A je označení uzlu, ve sloupci B pak hodnota příslušného parametru. Dodejme, že tyto parametry jsou definovány pro všechny uzly, tedy i ty, ke kterým

daný parametr nepřísluší (např. o_i u skládky). V takovém uzlu nabývá hodnoty 0.

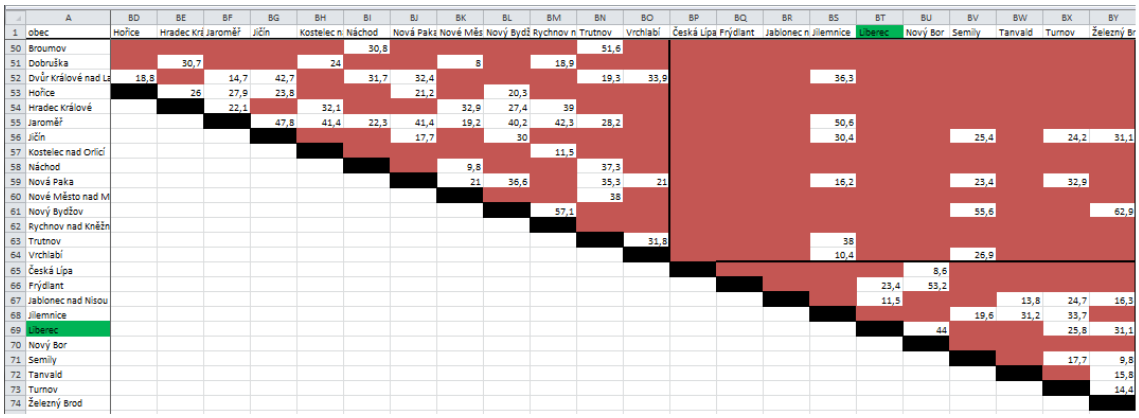
6.2.1. Produkce SKO o_i

Tyto údaje byly poskytnuty pro 207 ORP (spolu s údaji o počtu obyvatel těchto obcí), proto i zde musíme udělat mírnou úpravu, a to doplnit údaje k obcím, které jsou ze správních jednotek ORP vyňaty. U těchto obcí určíme produkci SKO jako počet obyvatel této obce vynásobený průměrnou produkcí celé ORP, do které patří. Samozřejmě musíme upravit produkci ORP, z které obec byla vyjmuta tak, aby celková produkce zůstala stejná.

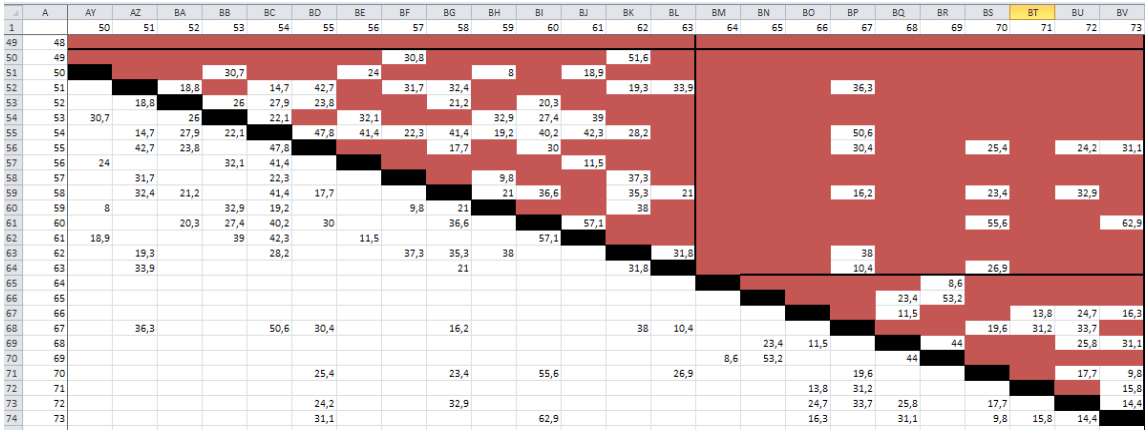
6.2.2. Kapacita skládek C_i^{SKL}

U tohoto parametru je problém, že některé skládky se nacházejí v lokalitě, kterou v modelu neuvažujeme. Kapacitu takovéto skládky přičteme ke kapacitě skládky, která přísluší obci z modelu, která je nejblíže.

Model počítá s údaji vztaženými na 1 rok. Dále pro jednoduchost předpokládáme, že skládka je projektovaná na cca 25 let, proto tuto projektovou kapacitu vydělíme. (viz [13])



a)



b)

Obr. 6.1. Ukázka vlastního označení indexů a použití makra Symetrie()

7. Model

Nyní už máme připravena všechna data a nachystaný matematický aparát, abychom mohli sestavit optimalizační úlohu. V této kapitole si ukážeme dva modely, z nichž každý má svůj odlišný účel.

7.1. Model I

Cílem tohoto modelu je odhadnout kapacity EVO a MBÚ, které dosud nejsou postaveny.

Na situaci pohlížíme, jako bychom byli „starostou všech obcí“, a za minimálních nákladů na rozvoz a zpracování odpadu určíme cesty a množství odpadu po nich převážených. Poté vyhodnocujeme informace, kolik odpadu se do které EVO či MBÚ, a následně lehké frakce do ZLF, odveze.

Proměnné, jejichž hodnoty hledáme tak, abychom minimalizovali účelovou funkci, jsou tyto:

x_j – množství SKO v tunách převážené přes hranu j

l_j – množství LF v tunách převážené přes hranu j

t_j – množství TF v tunách převážené přes hranu j

Nyní formulujeme optimalizační úlohu, tj. účelovou funkci + omezující podmínky, a provedeme její rozbor.

$$\min z = \sum_j d\, v_j\, x_j + \sum_i \sum_j a_{ij} x_j (p_i^{EVO} + p_i^{SKL}) + \sum_j e\, v_j\, l_j - \sum_i \sum_j 2{,}5\, a_{ij}\, l_j\, p_i^{MBU}$$

za podmínek, že pro každý uzel i platí:

1) $\sum_j a_{ij} x_j + o_i + \delta_i^{MBU} (\sum_j a_{ij} l_j + \sum_j b_{ij} t_j) \leq C_i^{EVO} + C_i^{SKL}$
2) $\sum_j a_{ij} l_j \leq C_i^{ZLF}$
3) $-2{,}5 \sum_j a_{ij} l_j \leq C_i^{MBU}$
4) $\delta_i^{MBU1}, 5 \sum_j a_{ij} l_j = \delta_i^{MBU} \sum_j b_{ij} t_j$
5) $\delta_i^{SKL} (\sum_j a_{ij} x_j + \sum_j b_{ij} t_j) \leq C_i^{SKL}$

Seznam symbolů a jejich význam najdeme na str. 36.

V modelu se často vyskytuje člen $\sum_j a_{ij}\ x_j$ nebo podobný (např. místo x_j je t_j nebo l_j, místo a_{ij} může být b_{ij}) – tento člen vždy představuje rozdíl přivezeného a odvezeného množství x (t, l) do a z uzlu i.

účelová funkce:

$\sum_j\ d\ v_j\ x_j$ představuje celkové náklady na rozvoz SKO, d je cena za přepravu 1 tuny SKO na vzdálenost 1 km [Kč/t·km], v_j pak vzdálenost hrany j

$\sum_i\sum_j\ a_{ij}\ x_j\ (p_i^{EVO} + p_i^{SKL})$ vyjadřuje náklady na zpracování SKO v EVO nebo jejich skládkování. Protože z uzlů *s nebo *k (viz obr. 5.1.) nevede žádná hrana (a_{ij} je vždy kladné), nemůže tento člen být záporný a pouze v těchto uzlech p_i^{EVO} nebo p_i^{SKL} jsou nenulové.

$\sum_j\ e\ v_j\ l_j$ obdobně jako v případě SKO takto vyjádříme celkovou cenu na rozvoz LF, kde e je cena za přepravu 1 tuny LF na vzdálenost 1 km, v_j pak vzdálenost hrany j

$-\sum_i\sum_j 2{,}5\ a_{ij}\ l_j\ p_i^{MBU}$ představuje náklady na zpracování v MBÚ. Množství SKO, které se přiveze do MBÚ ke zpracování je rovno 2,5násobku LF (LF je obsaženo přibližně 40% v SKO), které se z MBÚ odveze (proto znaménko –).

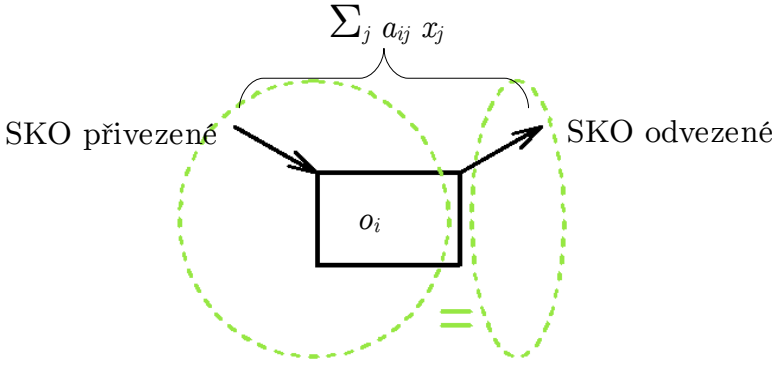
omezující podmínky:

Podmínka 1) nám zajišťuje mnohé. U obcí, aby v nich žádný SKO nezůstal, u EVO a skládek, aby nebyla překročena kapacita, a u MBÚ bylo možné SKO rozdělit na LF a TF. Podívejme se, jak bude tato podmínka vypadat v jednotlivých typech uzlů po zanedbání nulových členů.

Obce bez MBÚ ($\delta_i^{MBU} = 0,\ C_i^{EVO} = C_i^{SKL} = 0$) :

$$\sum_j a_{ij}\ x_j + o_i \leq 0$$

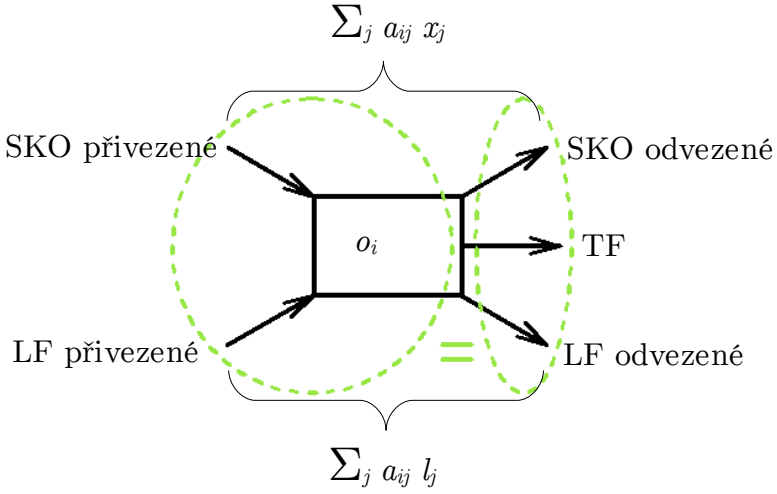
a tedy množství SKO, které se přiveze + množství SKO, které se v obci i produkuje, musí být menší nebo rovno (v tomto případě nastává rovnost) než množství odpadu, které se odveze. Žádný odpad se nemůže v obci i „hromadit“.



Obce s MBU ($C_i^{EVO} = C_i^{SKL} = 0$, δ_i^{MBU}=1)

$\sum_j a_{ij}\ x_j + o_i + \sum_j a_{ij}\ l_j + \sum_j b_{ij}\ t_j \leq 0$

Tato rovnice je podobná té předchozí s tím rozdílem, že do bilance zahrneme ještě možnost „přeměnit“ SKO na LF a TF [správný poměr třídění zajišťuje Podmínka 4)]. Tato podmínka zajišťuje platnost zákona zachování hmoty.



EVO a skládky (δ_i^{MBU}= 0, $o_i = 0$)

$\sum_j a_{ij}\ x_j \leq C_i^*$, kde C_i^* je buď C_i^{EVO} nebo C_i^{SKL}

Rovnice říká, že kapacita EVO nebo skládky musí být větší než množství SKO, které do ní dovezeme ke zpracování.

ZLF (δ_i^{MBU}= 0, $o_i = 0$, $C_i^{EVO} = C_i^{SKL} = 0$)

$\sum_j a_{ij}\ x_j \leq 0$

Z této rovnice vyplývá, že do ZLF se žádný SKO dovézt nemůže, musel by se v tom případě i odvézt, to ale není možné, protože ze ZLF nevede žádná hrana.

Podmínka 2) nám zajišťuje „základní rozvoz“ LF. Říká nám, že do obce se nesmí přivézt více LF, než se odveze. Je-li uzel ZLF, pak podmínka zajišťuje nepřekročení kapacity ZLF.

Podmínka 3) nám omezuje kapacitu MBÚ. Tu máme danou jako maximální množství SKO, které je MBÚ schopno ročně zpracovat. Proto množství LF, které z MBÚ může být získáno, je 2,5krát menší než množství SKO dovezeného ke zpracování.

Podmínka 4) pak zajišťuje, aby se odpad v MBÚ „roztřídil“ ve správném poměru, tj. LF : TF je 2:3 (40% : 60%).

Podmínka 5) hlídá, aby nebyla překročena kapacita skládek, kde je ukládán SKO a TF.

Máme formulován optimalizační model, který implementujeme do GAMSu. Jeho vyřešení nám mimo jiné dává informaci, jaké množství SKO se do které EVO, skládky či MBÚ doveze ke zpracování. Dalším požadavkem na model však je, abychom uvažovali jen ty MBÚ, které naplní danou minimální kapacitu, kterou si uživatel modelu navolí.

Kdybychom úlohu spočítali a pak pouze „vyškrtnuli“ nevyhovující MBÚ (nenaplnění požadované kapacity), dopustili bychom se tak chyby, protože je možné, že MBÚ, které v úloze nenaplní požadovanou kapacitu, ji naplní po „vyřazení“ jiných MBÚ.

Proto je v modelu naprogramována část kódu, která funguje následovně: Spočítáme základní úlohu s MBÚ uvažovanými ve všech obcích, které mají skládku. Najdeme MBÚ, do kterého se přiveze nejméně SKO ke zpracování (`minmbuvmodelu` v GAMSu). Je-li tato hodnota menší než požadovaná minimální kapacita (`minMBU` v GAMSu), dané MBÚ zrušíme (tj. nastavíme proměnnou δ_i^{MBU} této obce na 0) a úlohu znovu vyřešíme. Takto postupujeme stále dokola, dokud kapacita nejmenšího MBÚ nepřekročí naši požadovanou minimální hodnotu.

7.2. Model II

Dalším úkolem je sestavit model, který bude schopen určit maximální výšku poplatku vybrané EVO tak, aby se naplnila požadovaná kapacita této EVO, kterou bude uživatel moci zadávat. Tento maximální poplatek označíme p^{MAX}.

Výběr určité EVO v modelu provedeme pomocí parametru δ_i^{ZK}, který nastavíme na 1 u zařízení EVO, jehož hodnotu maximálního možného poplatku budeme hledat. Protože poplatek u této EVO nyní představuje p^{MAX}, je třeba nastavit p^{EVO} u tohoto EVO na 0, což provedeme jednoduchou pomínkou:

```
loop(i, if(binzkoumana(i) = 1, p_EVO(i) = 0));
```

Protože stále vycházíme z předpokladu celkových minimálních nákladů obcí na zpracování SKO, účelová funkce a omezující podmínky vypadají téměř stejně a mají i stejný význam, jako v případě Modelu I:

$$\min \; z = \sum_j d\, v_j x_j + \sum_i \sum_j a_{ij} x_j (\delta_i^{ZK} p^{MAX} + p_i^{EVO} + p_i^{SKL}) +$$
$$+ \sum_j e\, v_j l_j - \sum_i \sum_j 2{,}5\, a_{ij} l_j p_i^{MBU}$$

za podmínek, že pro každý uzel i platí:

1) $\sum_j a_{ij} x_j + o_i + \delta_i^{MBU} (\sum_j a_{ij} l_j + \sum_j b_{ij} t_j) \leq C_i^{EVO} + C_i^{SKL}$
2) $\sum_j a_{ij} l_j \leq C_i^{ZLF}$
3) $-2{,}5 \sum_j a_{ij} l_j \leq C_i^{MBU}$
4) $\delta_i^{MBU} 1{,}5 \sum_j a_{ij} l_j = \delta_i^{MBU} \sum_j b_{ij} t_j$
5) $\delta_i^{SKL} (\sum_j a_{ij} x_j + \sum_j b_{ij} t_j) \leq C_i^{SKL}$

Rozdíl oproti Modelu I je v účelové funkci, kde výraz $\delta_i^{ZK} p^{MAX}$ představuje poplatek vybraného zařízení EVO, který se snažíme najít co největší.

Toto hledání realizujeme metodou bisekce. Zavedeme dva odhady, shora a zdola (v modelu `horni` resp. `dolni`), které k sobě v cyklu while přibližujeme na základě podmínky o naplnění požadované kapacity tak dlouho, až je jejich rozdíl libovolně malý. Tento rozdíl představuje přesnost, s jakou jsme poplatek našli.

Počáteční odhady nastavíme tak, abychom kapacitu určitě naplnili v případě dolního odhadu, určitě nenaplnili v případě horního. Tento cyklus vypadá v GAMSu takto:

```
horni = 5000; dolni = 1;

while ((horni - dolni) > 1,
     p_max = (horni + dolni)/2;
     Solve transport using lp minimizing z;
     loop(i, if (binzkoumana(i) = 1,
          if (sum(j, A(i,j)*x.l(j)) < pozadovanakapacita,
          horni = (horni + dolni)/2;
          else dolni = (horni + dolni)/2)));
);
```

Tímto získáme p^{MAX} s přesností 1 jednotky. Tento výsledek si můžeme ověřit např. za pomoci Modelu I.

8. Zpracování výsledků

Model I implementujeme do GAMSu (viz přílohu B) a necháme spočítat s konkrétními vstupními daty, která byla stanovena na základě konzultace s ÚPEI.

Řešením tohoto modelu je množství SKO a LF (x_j, l_j), které převážíme po každé hraně tak, abychom dosáhli celkových minimálních nákladů. Pohled na výstup z GAMSu [viz obr. 8.1. a)] nám však sám o sobě příliš užitečných informací neposkytuje, proto musíme výsledky dále zpracovávat. V této kapitole máme za cíl:

- zjistit, která obec kam sváží SKO
- zjistit odhadnuté kapacity, tzn. kolik SKO se do kterého zařízení EVO či MBÚ sváží ke zpracování
- vizualizovat výsledky v podobě mapy

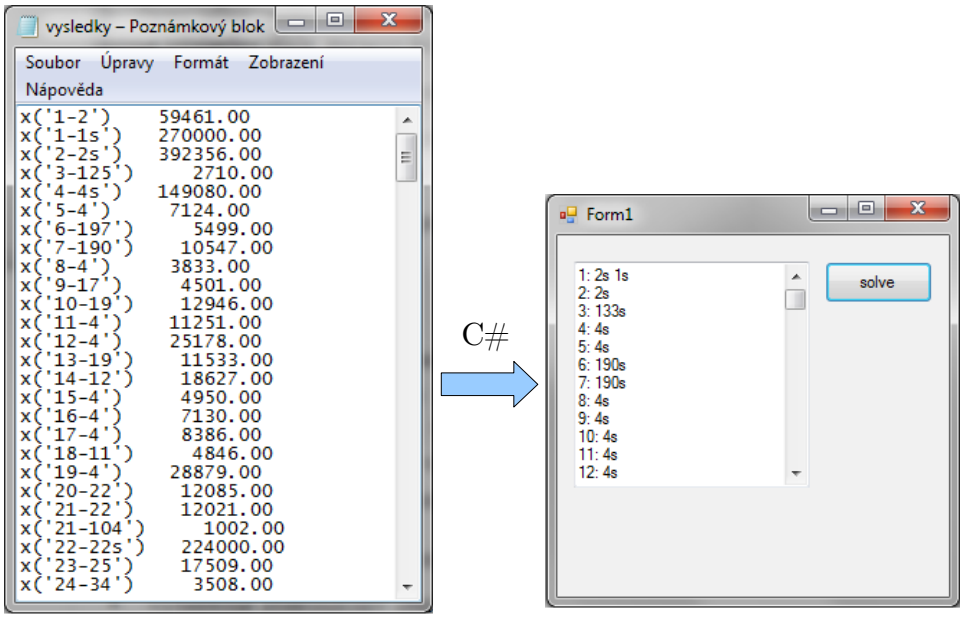
8.1. Svoz

Chceme zjistit, které obce kam sváží SKO. Nejprve využijeme možnosti GAMSu vypsat výsledky do textového souboru [viz obr 8.1. a)]. Poté vytvoříme program v jazyce C#, který je schopen tyto výsledky načíst a pomocí rekurzivní funkce dohledat, odkud kam se SKO vozí. Výstupem z programu je pak seznam obcí a k nim cílová místa, kde SKO skončí.

Postup algoritmu, který koncová místa dohledává, je tento:

- vybereme kteroukoliv obec a
- najdeme všechny hrany, u kterých je tato obec počátečním vrcholem
- zkontrolujeme, jestli je koncový vrchol skládka nebo EVO
- jestliže ano, můžeme tuto skládku nebo EVO přidat do cílových míst obce
- jestliže ne [tzn. koncový vrchol hrany je jiná obec b (tj. hrana a–b)], rekurzivně zavoláme tuto funkci s tím, že vstupním argumentem bude tato druhá obec b

Když tímto algoritmem projdeme všechny obce z modelu, získáme ke všem informaci, ve které EVO nebo skládce SKO skončí [viz obr. 8.1 b)].



Obr. 8.1. a) výstup z GAMSu b) ukázka výpisu programu v C#

8.2. Odhadnutí kapacit

Dále chceme zjistit, kolik SKO se přiveze, do které EVO či MBÚ. Pro zjištění této informace využijeme možnost programovat v GAMSu jednoduché cykly a podmínky. Po vyřešení úlohy projdeme všechny uzly i a jestliže se jedná o EVO ($\delta_i^{EVO} = 1$), vypíšeme do textového souboru (příkaz `put`) informace o množství SKO přivezeného do tohoto zařízení EVO: $\sum_j a_{ij}\, x_j$

Celý příkaz tedy vypadá:

```
loop(i, if (binspalovny(i)>0, put i.tl;
put sum(j, A(i,j)*x.l(j))));
```

Podobně pro množství SKO zpracovaného v MBÚ:

```
loop(i, if (binmbu(i)>0, put i.tl;
put (-2.5*sum(j, A(i,j)*lf.l(j)))));
```

Zde opět využíváme přepočtu množství dovezeného SKO ke zpracování na množství odvezené LF. Kdybychom chtěli počítat pomocí bilance x_j, museli bychom do ní započítat i produkci SKO uzlu i.

8.3. Vizualizace výsledků

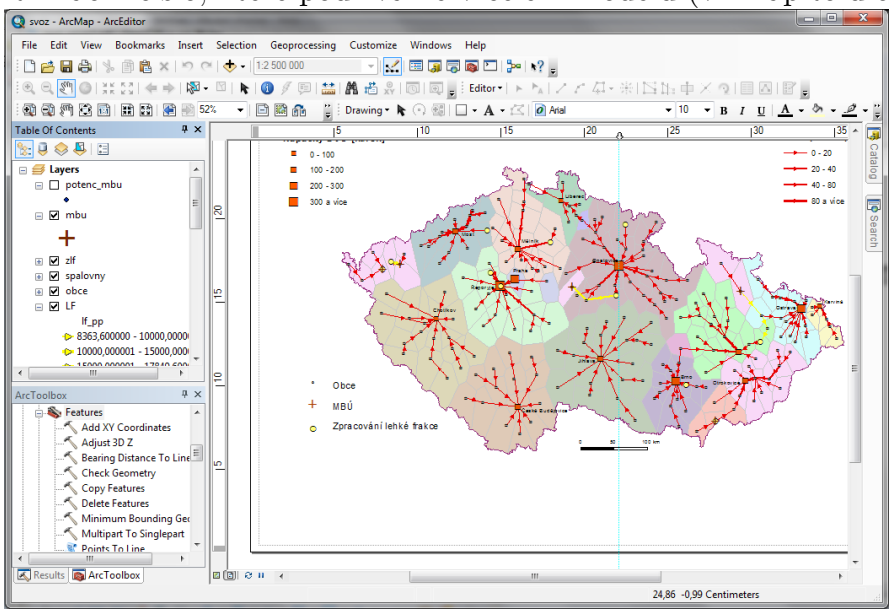
Podstatnou částí práce z pohledu prezentace výsledků je vizualizace pomocí mapy svozu, která přehledně znázorňuje celou situaci. Mapu vytvoříme v programu

ArcGIS, přičemž využíváme výsledků z části 8.1. a 8.2. , které načítáme do prostředí ArcGIS díky možnosti propojení s MS Excel.

ArcGIS

ArcGIS je geografický informační systém (GIS) od firmy Esri, který slouží pro práci s mapami a geografickými informacemi. Obsahuje řadu vzájemně propojených aplikací, z nichž využijeme ArcMap pro tvorbu mapového výstupu. Tento program umožňuje propojení s MS Excel, odkud je možno načítat data. Geografická data České republiky nám byla pro účel řešení práce poskytnuta Katedrou geoinformatiky Univerzity Palackého v Olomouci.

ArcMap umožňuje synchronizaci těchto dat s daty v MS Excel, která chceme znázornit, na základě specifického identifikačního čísla. Pomocí toho už můžeme provést výběr pouze těch obcí, které v modelu uvažujeme, a také k nim přidat vlastní identifikační číslo, které používáme v celém modelu (viz kapitolu 5.1.).



Obr. 8.2. Ukázka prostředí aplikace ArcMap 10

Dalším úkolem je znázornit dopravní síť České republiky, kterou máme zadanou pomocí incidenční matice. ArcMap obsahuje nástroj *XY To Line*, pomocí něhož znázorníme dopravní propojení jako úsečky mezi jednotlivými obcemi. Tím vzniknou nová data typu *line*, kterým je možno přiřazovat potřebné parametry - údaje o přepravě SKO a LF.

Chceme-li znázornit svozové oblasti, je třeba vytvořit data typu *polygon*, které pokryjí celou ČR a budou představovat území jednotlivých obcí. Dané plochy vytvoříme jako Voroneho buňky kolem jednotlivých obcí. Ačkoliv tedy tato území neodpovídají reálnému stavu, pro vizualizační potřeby je toto plně dostačující.

Vykreslenou mapu si je možno prohlédnout v příloze D.

9. Závěr

Cílem práce bylo sestavení optimalizačních modelů dopravní úlohy, které slouží pro analýzu současného problému nakládání se směsným komunálním odpadem na území ČR.

Po seznámení se základními pojmy a nastínění problému jsme si ve druhé a třetí kapitole představili matematický aparát, který pro řešení úlohy využíváme. Další kapitola se pak věnuje optimalizačnímu softwaru GAMS a práci s ním. Podrobněji jsme se zaměřili na importování dat z prostředí MS Excel. Kapitoly 5, 6, 7 už se zabývají vlastním modelům. Nejdříve popisu jejich prvků, následně přípravě dat, která importujeme do prostředí GAMS, a poté formulací modelů.

Sestavili jsme dva modely, z nichž každý má jiný účel, ale oba vychází ze stejného předpokladu, a sice z minimalizace nákladů obcí na zpracování odpadu. Model I slouží k určení svozových oblastí a odhadu kapacit plánovaných zařízení, Model II k určení maximálního poplatku jednoho vybraného zařízení.

V osmé kapitole jsme provedli ukázku zpracování výsledků na poskytnutých datech a jejich vizualizaci do mapy svozu.

Samozřejmě se nabízí otázka na další možný rozvoj modelů a jejich vylepšování. Protože jsme oba modely řešili jako úlohu lineárního programování, museli jsme stanovit cenu za přepravu 1 tuny odpadu na 1 km jako konstantní hodnotu. V reálné situaci však tato hodnota bude funkcí jak převáženého množství, tak přepravní vzdálenosti. Zahrnutí tohoto požadavku by však vyžadovalo jinou strukturu grafu reprezentujícího dopravní sít a samozřejmě použití nelineárního programování.

Literatura

[1] UCEKAJ, V.: *Analýza možností nakládání s komunálními odpady v rámci mikroregionu*, disertační práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2010. 153 s.

[2] Zákon č. 185/2001 Sb., o odpadech a o změně některých dalších zákonů [online][cit 2012-5-4]. Dostupný z: http://business.center.cz/business/pravo/zakony/odpady/cast1.aspx

[3] Svaz měst a obcí ČR, Asociace krajů ČR: *Aktualizace strategie rozvoje nakládání s odpady v obcích a městech ČR*, květen 2011.

[4] BAZARAA, M.S., JARVIS, J.J.: *Linear Programming and Network Flows*, first edition, John Wiley & Sons, Inc., New York, 1977, ISBN 0-471-06015-1.

[5] KLAPKA, J., DVOŘÁK, J., POPELA, P.: *Metody operačního výzkumu*, druhé vydání, Nakladatelství VUTIUM, Brno, 2001

[6] DEMEL, J.: *Grafy a jejich aplikace*, první vydání, Nakladatelství Academia, Praha, 2002

[7] HRABEC, D.: *Optimalizace inženýrského návrhu*, bakalářská práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2009, 42 s.

[8] KUREŠ, M.: *Matematická analýza II* [online]. ÚM FSI VUT v Brně, poslední revize 17. leden 2008 http://mathonline.fme.vutbr.cz/Matematicka-analyzanbspII/sc-1226/default.aspx

[9] MOLLIKOVÁ, E. *Modelovací jazyky v optimalizaci*, bakalářská práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010, 54 s.

[10] GAMS Solvers [online] [cit 2012-5-8]. Dostupný z http://www.gams.com/solvers/solvers.htm

[11] CPLEX Solver Manual [online] [cit 2012-5-8]. Dostupný z http://www.gams.com/dd/docs/solvers/cplex.pdf

[12] GAMS Data Exchange with Excel [online] [cit 2012-5-8]. Dostupný z http://interfaces.gams-software.com/doku.php?id=excel:excel

[13] BARTÁČKOVÁ, L.: Atlas zařízení pro nakládní s odpady, Výzkumný ústav vodohospodářský T.G.Masaryka, Praha, 2010

Seznam použitých symbolů a zkratek

název	význam	GAMS	jednotka
SKO	směsný komunální odpad		
LF	lehká frakce		
TF	těžká frakce		
MBÚ	mechanicko-biologická úprava		
EVO	zařízení na energetické využití odpadu		
i	indexová množina uzlů		
j	indexová množina hran		
A = (a_{ij})	incidenční matice pro rozvoz SKO a LF	`A(i,j)`	
B = (b_{ij})	incidenční matice pro rozvoz TF	`B(i,j)`	
x_j	množství SKO převážené po hraně j	`x(j)`	[t/rok]
l_j	množství LF převážené po hraně j	`lf(j)`	[t/rok]
t_j	množství TF převážené po hraně j	`tf(j)`	[t/rok]
o_i	produkce SKO uzlu i	`odp(i)`	[t/rok]
C_i^{EVO}	kapacita zařízení EVO v uzlu i	`C_EVO(i)`	[t/rok]
C_i^{SKL}	kapacita skládky v uzlu i	`C_SKL(i)`	[t/rok]
C_i^{MBU}	kapacita MBÚ	`C_MBU(i)`	[t/rok]
C_i^{ZLF}	kapacita ZLF	`C_ZLF(i)`	[t/rok]
p_i^{EVO}	poplatek za zpracování 1t SKO u EVO v uzlu i	`p_EVO(i)`	[Kčt/t·rok]
p_i^{SKL}	popltatek za uložení 1t SKO v uzlu i	`p_SKL(i)`	[Kčt/trok]
p_i^{MBU}	poplatek za zpracování 1t SKO technologií MBÚ	`p_MBU(i)`	[Kč/t·rok]
p^{MAX}	hledaný maximální poplatek	`p_max`	[Kč/t·rok]
v_j	délka hrany j (vzdálenost uzlů, které spojuje)	`v(j)`	[km]
d	cena za přepravu 1t SKO	`d`	[Kč/t·km]
e	cena za přepravu 1t LF	`e`	[Kč/t·km]
δ_i^{MBU}	= 1, když uzel i typu obec s MBÚ, 0 jinak	`binmbu(i)`	
δ_i^{EVO}	= 1, když uzel i typu EVO, 0 jinak	`binspalovny(i)`	
δ_i^{SKL}	= 1, když uzel i typu skládka, 0 jinak	`binskladky(i)`	
δ_i^{ZK}	= 1 u EVO, u níž hledáme p^{MAX} v Modelu II	`binzkoumana(i)`	
	požadovaná minimální kapacita EVO	`pozadovanakapacita`	[t/rok]
	minimalní kapacita MBÚ, kterou požadujeme	`minMBU`	[t/rok]
	hodnota kapacity nejmenšího MBÚ v Modelu I	`minmbuvmodelu`	[t/rok]

Příloha A
Makra použitá k přípravě dat

```
Sub Symetrie()
' doplni tabulku na symetrickou
    i = 1
    Do While (i < 400)
    j = i
        Do While (j < 400)
            If (Cells(i + 1, j + 1).Value > 0) Then
            Cells(j + 1, i + 1).Value = Cells(i + 1, j + 1).Value
            End If
            j = j + 1
        Loop
    i = i + 1
    Loop
End Sub

Sub Incid()
' vytvori na druhem liste incid. matici a ohodnoti hrany na tretim listu
 Dim a As String
 Dim b As String
 sloupec = 2
 i = 1
    Do While (i < 400)
    j = 1
    Sheets("List2").Cells(i, 1).Value = Sheets("List1").Cells(i, 1).Value
        Do While (j < 400)
            If (Sheets("List1").Cells(i + 1, j + 1).Value > 0) Then
            a = Sheets("List1").Cells(i + 1, 1).Value
            b = Sheets("List1").Cells(1, j + 1).Value
            c = a + "-" + b
            ' vytvori incidencni matici na druhem listu
            Sheets("List2").Cells(1, sloupec).Value = c
            Sheets("List2").Cells(i + 1, sloupec).Value = -1
            Sheets("List2").Cells(j + 1, sloupec).Value = 1

            ' ohodnoti hrany na tretim listu
            Sheets("List3").Cells(sloupec - 1, 1).Value = c
            Sheets("List3").Cells(sloupec - 1, 2).Value =
      Sheets("List1").Cells(i + 1, j + 1).Value
            sloupec = sloupec + 1
            End If
            j = j + 1
        Loop
    i = i + 1
    Loop
End Sub
```

Příloha B
Zdrojový kód GAMSu Modelu I

```
scalar
deltaZdar    /0/
deltaJihlava /1/
deltaUsti    /0/
deltaMost    /1/

d naklad na tunu*km (x) /4/

e naklad na tunu*km (lf) /2/

*nastavime minimalni kapacitu MBU, kterou jsme ochotni pripustit
minMBU  minimalni hodnota kapacity MBU
/20000/

minmbuvmodelu

set
i uzly /1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
j hrany /1-2, 1-140, 1-141, 1-142, 1-144, 1-145, 1-146, 1-148, 1-150,...

parameters
$INCLUDE cr_data.gms

variable
z celkove naklady

positive variable
x(j) odpadu pres hranu j
lf(j) lehke frakce pres hranu j
tf(j) tezka frakce

equations
ucelfce, omez1(i), omez2(i), omez3(i), omez4(i), omez5(i);

ucelfce.. z =e= sum(j, v(j)*d*x(j)) + sum(i,sum(j, A(i,j)*x(j))*(p_EVO(i)
          + p_SKL(i))) + sum(j, v(j)*e*lf(j))
          - sum(i,binmbu(i)*sum(j, A(i,j)*2.5*lf(j))*(p_MBU(i)));
omez1(i)..  sum(j, A(i,j)*x(j)) - C_EVO(i)+odp(i) - C_SKL(i) + binmbu(i)
            *(sum(j, A(i,j)*lf(j))+sum(j,B(i,j)*tf(j))) =l= 0;
omez2(i).. sum(j, A(i,j)*lf(j)) - C_ZLF(i) =l= 0;
omez3(i)..  -2.5*sum(j, A(i,j)*lf(j))- C_MBU(i) =l= 0;
omez4(i).. binmbu(i)*1.5*sum(j, A(i,j)*lf(j)) =e=
           binmbu(i)*sum(j, B(i,j)*tf(j));
omez5(i).. binskladky(i)*(sum(j, A(i,j)*x(j)) + sum(j, B(i,j)*tf(j))) -
C_SKL(i) =l= 0;

file vysledky /vysledky_model1.txt/;
put vysledky;
model transport /all/;

if(deltaZdar = 0, C_EVO('200s')=0);
if(deltaJihlava = 0, C_EVO('190s')=0);
if(deltaUsti = 0, C_EVO('180s')=0);
```

```
if(deltaMost = 0, C_EVO('175s')=0);

*zde zacina "vyhazovani" mbu, ktere nedosahnou pozadovane kapacity
solve transport using lp minimizing z;
*prvni iterace, zjistime kapacitu nejmensiho MBU:
minmbuvmodelu = 60000;
loop(i, if (binmbu(i)>0, if((-2.5*sum(j, A(i,j)*lf.l(j)))>0,
      if((-2.5*sum(j, A(i,j)*lf.l(j)))< minmbuvmodelu, minmbuvmodelu =
      -2.5*sum(j, A(i,j)*lf.l(j))))));
*cyklus "vyhazovani"
while (minmbuvmodelu<minMBU,
      minmbuvmodelu=60000;
      loop(i, if (binmbu(i)>0,if((-2.5*sum(j, A(i,j)*lf.l(j)))>0,
            if((-2.5*sum(j, A(i,j)*lf.l(j)))< minmbuvmodelu,
            minmbuvmodelu = -2.5*sum(j, A(i,j)*lf.l(j))))));
      loop(i, if((-2.5*sum(j, A(i,j)*lf.l(j)))= minmbuvmodelu,
      binmbu(i) = 0));

      solve transport using lp minimizing z;

      minmbuvmodelu=60000;
      loop(i, if (binmbu(i)>0, if((-2.5*sum(j, A(i,j)*lf.l(j)))>0,
      if((-2.5*sum(j, A(i,j)*lf.l(j)))< minmbuvmodelu, minmbuvmodelu =
      -2.5*sum(j, A(i,j)*lf.l(j))))));
);
*zde konci "vyhazovani" mbu, ktere nedosahnou pozadovane kapacity

*vypis dovozu:
loop(i, if (binspalovny(i)>0, put i.tl; put sum(j, A(i,j)*x.l(j));
put'      '; put/;));
loop(i, if (binmbu(i)>0, put i.tl; put (-2.5*sum(j, A(i,j)*lf.l(j)));
put'      '; put/;));

**vypis cest:
*put 'celkem    :'; put celkem.l; put /; put /;
*loop(j, if(x.l(j)>0,put x(j); put x.l(j); put /));
*loop(j, if(lf.l(j)>0,put lf(j); put lf.l(j); put /));

display x.l, lf.l, z.l;
```

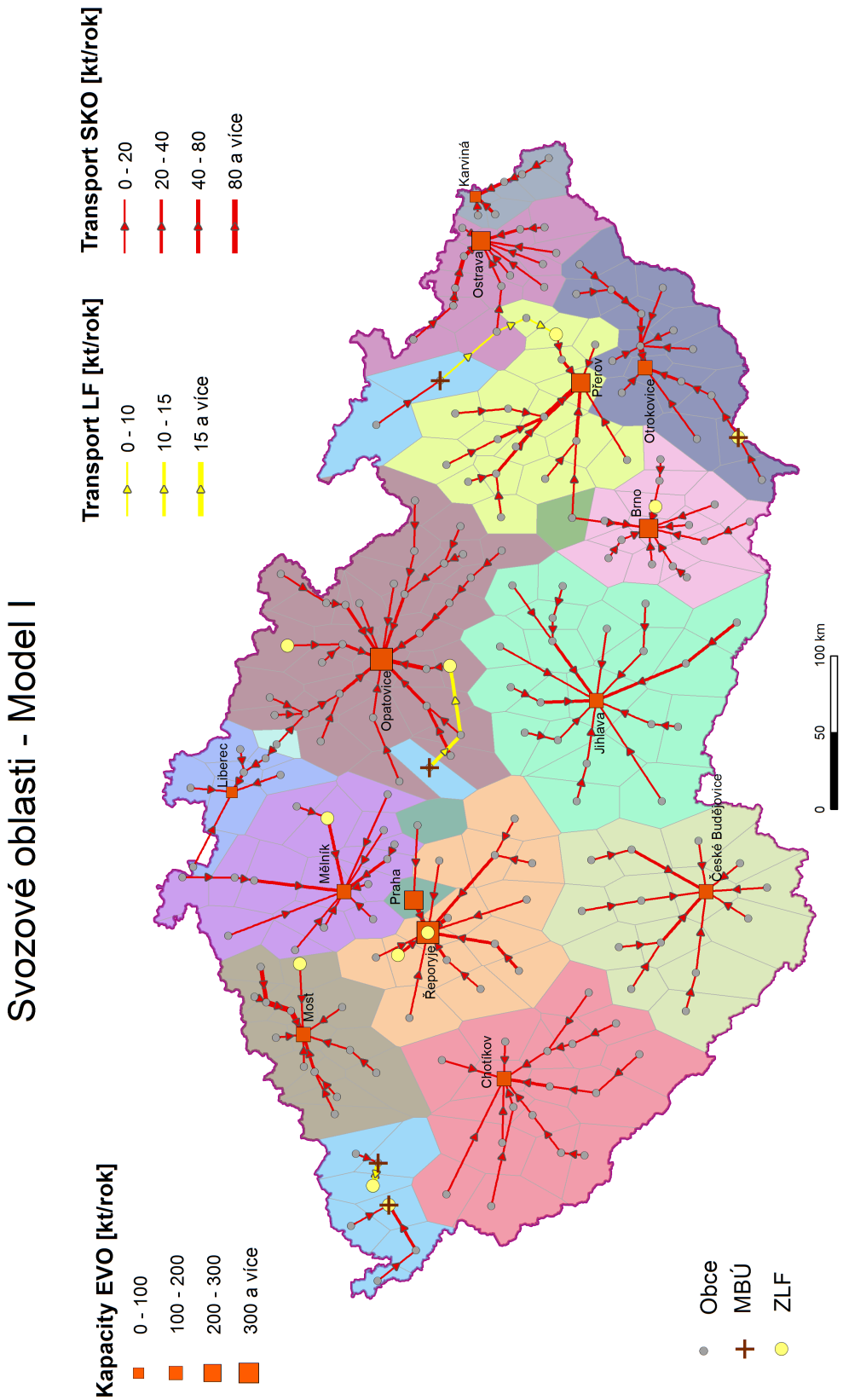
Příloha C
Zdrojový kód GAMSu Modelu II

```
scalar
deltaZdar    /0/
deltaJihlava /1/
deltaUsti    /0/
deltaMost    /1/

d naklad na tunu*km (x) /4/

e naklad na tunu*km (lf) /2/

*nastavime minimalni kapacitu, kterou chceme naplnit:
pozadovanakapacita pozadovana kapacita EVO v tunach
/150000/

p_max, horni, dolni;

set
i uzly /1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,...
j hrany /1-2, 1-140, 1-141, 1-142, 1-144, 1-145, 1-146, 1-148, 1-150,...

parameters
$INCLUDE cr_data.gms

*vybereme EVO, kterou chceme pocitat
binzkoumana(i)
/22s 1/

variable
z celkove naklady

positive variable
x(j) odpadu pres hranu j
lf(j) lehke frakce pres hranu j
tf(j) tezka frakce

equations
ucelfce, omez1(i), omez2(i), omez3(i), omez4(i), omez5(i);

ucelfce.. z =e= sum(j, v(j)*d*x(j))
            + sum(i,sum(j,A(i,j)*x(j))*(binzkoumana(i)*p_max + p_EVO(i)
            + p_SKL(i))) + sum(j, v(j)*e*lf(j))
            - sum(i,binmbu(i)*sum(j, A(i,j)*2.5*lf(j))*(p_MBU(i)));
omez1(i)..  sum(j, A(i,j)*x(j)) - C_EVO(i)+odp(i) - C_SKL(i) + binmbu(i)
            *(sum(j, A(i,j)*lf(j))+sum(j,B(i,j)*tf(j))) =l= 0;
omez2(i)..  sum(j, A(i,j)*lf(j)) - C_ZLF(i) =l= 0;
omez3(i)..  -2.5*sum(j, A(i,j)*lf(j))- C_MBU(i) =l= 0;
omez4(i)..  binmbu(i)*1.5*sum(j, A(i,j)*lf(j)) =e=
            binmbu(i)*sum(j, B(i,j)*tf(j));
omez5(i)..  binskladky(i)*(sum(j, A(i,j)*x(j)) + sum(j, B(i,j)*tf(j))) -
            C_SKL(i) =l= 0;

file vysledky /vysledky_model2.txt/;
```

```
put vysledky;
model transport /all/;

if(deltaZdar = 0, C_EVO('200s')=0);
if(deltaJihlava = 0, C_EVO('190s')=0);
if(deltaUsti = 0, C_EVO('180s')=0);
if(deltaMost = 0, C_EVO('175s')=0);

loop(i, if(binzkoumana(i) = 1, p_EVO(i) = 0));

*bisekce na hledani p_max
horni = 5000; dolni = 1;
while ((horni - dolni)>1,
      p_max = (horni + dolni)/2;
      solve transport using lp minimizing z;
      loop(i, if (binzkoumana(i) = 1,
      if (sum(j, A(i,j)*x.l(j)) < pozadovanakapacita,
      horni = (horni + dolni)/2; else dolni = (horni + dolni)/2)));
);

loop(i, if (binzkoumana(i) = 1,
      put "zkoumana EVO:                        "; put i.tl; put/;
      put "pozadavek na minimalni kapacitu: ";put pozadovanakapacita;put/;
      put "maximalni poplatek:             "; put p_max;));

display x.l, lf.l, z.l, p_max;
```

Příloha D

Mapa svozu SKO podle Modelu I



Příloha E
Seznam příloh na CD

`/modely` obsahuje:

- vstupní data – celkem 15 souborů .xls, které načítáme
- `projekt.gpr`
- `data.gms` soubor zajišťujicí načtení dat z MS Excel do GAMSu
- `model1.gms` implementace Modelu I (viz kapitolu 7.1) do GAMSu
- `model2.gms` implementace Modelu II (viz kapitolu 7.2) do GAMSu

`/mapa` obsahuje mapu svozu a svozových oblastí podle Modelu I (viz přílohu D) ve formátu .png v rozlišení 4959 × 3509 pixelů.