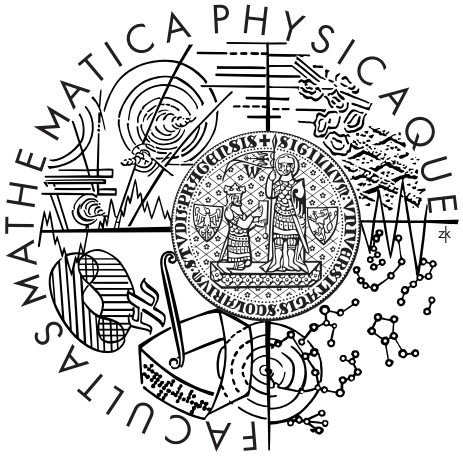
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Tomáš Petráš

Řešení optimalizačních úloh pomocí různých softwarů

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: RNDr. Ing. Miloš Kopa Ph.D.

Studijní program: Matematika

Studijní obor: obecná matematika

Praha 2011

Ďakujem RNDr. Ing. Milošovi Kopovi Ph.D. za konzultácie, zapožičanie literatúry a poskytnutie počítača k výpočtom, Anke a Libuši za pomoc s angličtinou a gramatickú korektúru, rodine za podporu.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Řešení optimalizačních úloh pomocí různých softwarů

Autor: Tomáš Petráš

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: RNDr. Ing. Miloš Kopa Ph.D.

Abstrakt: Táto práca sa zaoberá praktickými problémami pri riešení úloh lineárneho programovania v matematických softvéroch Mathematica, GAMS, R a Matlab. Popisuje základné vlastnosti riešičov ("solverov"), balíkov ("packageov") a optimalizačných funkcií daných softvérov. Hlavným cieľom práce je ich porovnanie na základe časovej efektívnosti riešenia úloh rôznej veľkosti v rámci jednotlivých sofvérov, ale aj medzi softvérmi samotnými. Časi riešenia uvažujeme s i bez vplyvu načítania vstupných údajov. Na záver zo získaných údajov vyberieme pre jednotlivé veľkosti úloh najvhodnejší softvér a dáme doporučenia pre praktické využitie týchto softvérov pri riešení úloh lineárneho programovania.

Klíčová slova: lineárne programovanie, optimalizačné úlohy, softvérové spracovanie

Title: Optimization problems solving using various software

Author: Tomáš Petráš

Department: Department of Probability and Mathematical Statistics

Supervisor: RNDr. Ing. Miloš Kopa Ph.D.

Abstract: This thesis deals with practical problems when solving tasks of linear programming using mathematical software Mathematica, Gams, R and Matlab. It describes the basic properties of solvers, packages and optimizing functions of these programs. The aim of the thesis is to compare computer times needed to solve problems of different sizes using the considered programs. We consider these time efficiencies with and without the time consumed by loading the input data. At the end of the thesis we find the most suitable program for each of the problems sizes and we give some recomendations for the practical use of these products when solving problems of linear programming.

Keywords: linear programming, optimizing problems, software support

Obsah

Kapitola 1

Úvod

Lineárne programovanie je najjednoduchším odvetvím optimalizácie. Jeho hlavným zámerom je hľadanie maxima alebo minima lineárnej funkcie na zadanej polyedrickej množine. Začiatky lineárneho programovania siahajú do roku 1939 kedy jeho princípy prvýkrát použil ruský matematik, Leonid Kantorovič pri plánovaní zníženia nákladov ruskej armády a zvýšení strát nepriateľa počas druhej svetovej vojny. Ďalšími priekopníkmi v tejto oblasti boli George Bernard Dantzig, ktorý v roku 1947 vyvinul simplexový algoritmus a John von Neumann, autor princípu duality. Od tejto doby si lineárne programovanie našlo uplatnenie v mnohých odvetviach ako napr. v makroekonómii, v plánovaní výroby a dopravy podniku alebo pri optimalizovaní toku dát v počítačových sietiach [7].
Cieľom našej práce bolo zhrnúť základné poznatky o lineárnom programovaní a popísať najdôležitejšie metódy slúžiace na riešenie optimalizačných úloh akými sú simplexová alebo duálna simplexová metóda. V empirickej časti porovnávame časovú efektivitu riešenia matematických softvérov Mathematica, GAMS, R a Matlab pomocou metód opísaných v teoretickej časti na úlohách rozdielnej veľkosti.
Práca je členená takto: po prvej kapitole nasleduje stručné oboznámenie sa s problematikou lineárneho programovania, hlavnými používanými definíciami, vetami a taktiež je bližšie opísaný jeden z najdôležitejších princípov lineárneho programovania, princíp duality. V tretej kapitole sú spomenuté základné metódy a algoritmy riešenia optimalizačných úloh s dôrazom na lineárne programovanie. V štvrtej kapitole približujeme rôzne možnosti optimalizácie v dnes najpoužívanejších matematických softvéroch. Predposlednú, piatu kapitolu tvorí numerická štúdia, ktorá je vlastným prínosom tejto práce. V nej je opísaný postup, podľa ktorého sme porovnávali časovú efektivitu spomenutých softvérov a jeho výsledky. Na základe nich sme podľa veľkosti úlohy určili najvhodnejší softvér pre jej riešenie. Po tejto kapitole nasleduje záverečná kapitola, ktorá zhrňuje výsledky tejto práce.

Kapitola 2

Teória lineárneho programovania

Všetky definície, vety a značenia použité v 2. a 3. kapitole sú prebraté zo skrípt [1] .

V lineárnom programovaní sa zaoberáme hľadaním riešenia optimalizačných úloh v tvare

$$\begin{aligned} min\{c^T x: \quad & A_{I_1\times J}x \geq b_{I_1}, \\ & A_{I_2\times J}x \leq b_{I_2}, \\ & A_{I_3\times J}x = b_{I_3}, \\ & x_{J_1} \geq 0, x_{J_2} \leq 0, x \in \mathbb{R}^J\} \\ max\{c^T x: \quad & A_{I_1\times J}x \geq b_{I_1}, \\ & A_{I_2\times J}x \leq b_{I_2}, \\ & A_{I_3\times J}x = b_{I_3}, \\ & x_{J_1} \geq 0, x_{J_2} \leq 0, x \in \mathbb{R}^J\} \end{aligned} \tag{2.1}$$

kde $c \in \mathbb{R}^J$, $b \in \mathbb{R}^I$, $A \in \mathbb{R}^{I\times J}$, $card(I) = m$, $I_1, I_2, I_3 \subset I$ sú po dvoch disjunktné a $I_1 \cup I_2 \cup I_3 = I$, $card(J) = n$, $J_1, J_2, J_3 \subset J$ sú po dvoch disjunktné a $J_1 \cup J_2 \cup J_3 = J$.

Hovoríme, že úloha lineárneho programovania je zadaná v:

štandardnom tvare ak $I_1 = \emptyset$, $I_2 = \emptyset$, $J_2 = \emptyset$, $J_3 = \emptyset$

$$\begin{aligned} min\{c^T x: Ax = b, x \geq 0\} \\ max\{c^T x: Ax = b, x \geq 0\} \end{aligned} \tag{2.2}$$

tvare nerovnosti ak $I_2 = \emptyset$, $I_3 = \emptyset$, $J_2 = \emptyset$, $J_3 = \emptyset$

$$\begin{aligned} &min\{c^T x : Ax \geq b, x \geq 0\} \\ &max\{c^T x : Ax \geq b, x \geq 0\} \end{aligned} \tag{2.3}$$

alebo $I_1 = \emptyset$, $I_3 = \emptyset$, $J_2 = \emptyset$, $J_3 = \emptyset$

$$\begin{aligned} &min\{c^T x : Ax \leq b, x \geq 0\} \\ &max\{c^T x : Ax \leq b, x \geq 0\} \end{aligned} \tag{2.4}$$

zmiešanom tvrare v ostatných prípadoch.

Jednoduchými transformáciami môžeme prevádzať maximalizačnú úlohu na minimalizačnú, jeden tvar úlohy na iný tvar úlohy.

Vynásobením účelovej funkcie koeficientom (-1) prevádzame minimalizačnú úlohu na maximalizačnú a opačne.

Vynásobením nerovnovnosti koeficientom (-1) môžeme otáčať nerovnosti.

Vynásobením premennej koeficientom (-1) zmeníme jej nekladnosť na nezápornosť a opačne.

Nerovnosť $\sum_{j \in J} A_{i,j} x_j \geq b_i$ (resp. $\sum_{j \in J} A_{i,j} x_j \leq b_i$) zmeníme na rovnosť pridaním tvz. **sklzových premenných** : $\sum_{j \in J} A_{i,j} x_j - v = b_i$ (resp. $\sum_{j \in J} A_{i,j} x_j + v = b_i$), $v \geq 0$. V účelovej funkcii priradíme týmto novým premenným nulový koeficient.

Rovnosť $\sum_{j \in J} A_{i,j} x_j = b_i$ môžme ekvivalentne vyjadriť ako $\sum_{j \in J} A_{i,j} x_j \geq b_i$ & $\sum_{j \in J} A_{i,j} x_j \leq b_i$.

Každú premennú x_j môžeme nahradiť jej rozdelením na kladnú a zápornú časť: $x_j = w^+ - w^-$, $w^+ \geq 0$, $w^- \geq 0$.

Z predchádzajúcich krokov vyplýva, že každý typ úlohy vieme zapísať v štandardnom tvare.

Definícia 2.1. *Množinu*

$$\{x : A_{I_1 \times J} x \geq b_{I_1}, A_{I_2 \times J} x \leq b_{I_2}, A_{I_3 \times J} x = b_{I_3}, x_{J_1} \geq 0, x_{J_2} \leq 0, x \in \mathbb{R}^J\} \quad (2.5)$$

budeme nazývať **množinou prípustných riešení** *úlohy (2.1).*

2.1 Štruktúra množiny prípustných riešení

Keďže každá úloha lineárneho programovania sa dá previesť do štandardného tvaru, tak sa budeme zaoberať iba vyšetrovaním štruktúry množiny prípustných riešení úloh v štandardnom tvare

$$\mathbf{M} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\} \quad (2.6)$$

kde $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$.

Definícia 2.2. *Povieme, že množina $A \subset \mathbb{R}^n$ je* **konvexná**, *ak pre každé dva body $x, y \in A$ a $0 < \lambda < 1$ platí $\lambda x + (1 - \lambda)y \in A$.*

Definícia 2.3. *Množina $A \subset \mathbb{R}^n$ sa nazýva* **konvexná polyedrická množina**, *ak existuje konečný počet uzavretých polopriestorov $H_1, H_2, ..., H_k$ tak, že $A = \bigcap_{i=1}^{k} H_i$.*

Veta 2.4. *Množina (2.6) je konvexná polyedrická množina. Špeciálne je konvexná a uzavretá.*

Definícia 2.5. *Povieme, že $s \in \mathbb{R}^n$ je* **smerom** *neprázdnej množiny $A \subset \mathbb{R}^n$, ak existuje bod $x \in A$ taký, že pre každé $\alpha > 0$ je $x + \alpha s \in A$. Množinu všetkých smerov množiny A budeme označovať $direct(A)$.*

Definícia 2.6. *Nech $A \subset \mathbb{R}^n$ je konvexná množina. Povieme, že bod $s \in A$ je* **krajným bodom** *A, ak neexistujú body $x, y \in A$, $x \neq y$ a $0 < \lambda < 1$ také, aby $s = \lambda x + (1-\lambda)y$. Množinu všetkých krajných bodov množiny A budeme označovať $ext(A)$.*

Definícia 2.7. *Nech $A \subset \mathbb{R}^n$ je neprázdna množina, potom definujeme* **nezáporný obal** *resp.* **konvexný obal** *množiny A, ktorý označujeme $pos(A)$ resp. $conv(A)$ ako:*

$$\begin{aligned} pos(A) &= \{\textstyle\sum_{s\in I} \lambda(s)s : \lambda(s) \geq 0 \ \forall\ s \in I,\ I \subset A \text{ konečná}\} \\ conv(A) &= \{\textstyle\sum_{s\in I} \lambda(s)s : \lambda(s) \geq 0 \ \forall\ s \in I,\ \sum_{s\in I} \lambda(s) = 1\ ,\ I \subset A \text{ konečná}\} \end{aligned}$$

Pre prázdnu množinu definíciu rozširujeme nasledovne:

$$pos(\emptyset) = \{\mathbf{0}\},\ conv(\emptyset) = \emptyset.$$

Definícia 2.8. *Nech $A \subset \mathbb{R}^n$ je konvexná množina. Povieme, že bod $s \in direct(A)$ je* **krajným smerom** *A, ak $\|s\| = 1$ a neexistujú body $x, y \in direct(A)$, $x, y \notin pos(\{s\})$ a $\lambda > 0, \varphi > 0$ také, aby $s = \lambda x + \varphi y$. Množinu všetkých krajných smerov množiny A budeme označovať $extd(A)$.*

Definícia 2.9. *Množina $A \subset \mathbb{R}^n$ sa nazýva :*

- **kužeľ** *(s vrcholom v počiatku), ak $\mathbf{0} \in A$ a pre $\forall\ s\ \in A$ a $\alpha > 0$ je $\alpha s \in A$*
- **konvexný polyedrický kužeľ**, *ak existuje konečná množina $S \subset \mathbb{R}^n$ taká, že $A = pos(S)$*

Veta 2.10. *Pre množinu (2.6) platí, že*

$$direct(\mathbf{M}) = \{x \in \mathbb{R}^n : Ax = 0, x \geq 0\} \tag{2.7}$$

je konvexný polyedrický kužeľ, ktorý je generovaný množinou svojich krajných smerov $extd(\mathbf{M})$*, čiže platí* $direct(\mathbf{M}) = pos(extd(\mathbf{M}))$*.*

Definícia 2.11. Dno množiny M *označujeme ako* $btt(\mathbf{M})$ *a definujeme ho ako*

$$btt(\mathbf{M}) = \{x \in \mathbf{M} : \nexists\, y \in direct(\mathbf{M}), y \neq 0, y \leq x\}. \tag{2.8}$$

Veta 2.12. *Pre množinu (2.6) platí* $btt(\mathbf{M}) \subset conv(ext(\mathbf{M}))$*.*

Veta 2.13. *Pre množinu (2.6) platí* $\mathbf{M} = btt(\mathbf{M}) + direct(\mathbf{M})$

Veta 2.14. *Množina (2.6) je súčtom konvexného polyedru a konvexného polyedrického kužeľa. Špeciálne* $\mathbf{M} = conv(ext(\mathbf{M})) + pos(extd(\mathbf{M}))$*.*

Veta 2.15. *Pre minimalizačnú úlohu lineárneho programovania zadanú v tvare (2.2) nastáva iba jedna z týchto troch možností:*

a) $\mathbf{M} = \emptyset$

b) $\mathbf{M} \neq \emptyset$ *a* $inf\{c^T x : x \in \mathbf{M}\} = -\infty$

c) $\mathbf{M}^* \neq \emptyset$

kde symbolom $\mathbf{M}^*$ *označujeme množinu všetkých optimálnych riešení. Naviac platí:*

- *ak* $\mathbf{M} \neq \emptyset$ *potom existuje krajný bod* $\mathbf{M}$
- *ak* $\mathbf{M}^* \neq \emptyset$ *potom existuje krajný bod* $\mathbf{M}$*, ktorý je optimálnym riešením úlohy*

Z vety 2.15 vyplývajú nasledujúce vlastnosti pre prípustné a optimálne riešenia úlohy (2.2):

- Pokiaľ neexistuje žiadny krajný bod, potom úloha nemá žiadne prípustné riešenie.

- Ak existuje krajný bod a účelová funkcia má pre nejaký smer zápornú hodnotu, potom minimalizačná úloha nemá optimálne riešenie. Účelová funkcia neobmedzene klesá, napríklad v smere krajného smeru, pre ktorý nadobúda zápornú hodnotu.

- Ak existuje krajný bod a účelová funkcia má pre každý krajný smer nezápornú hodnotu, potom úloha má optimálne riešenie. Jedným z optimálnych riešení je, napríklad, krajné riešenie s najmenšou hodnotou účelovej funkcie.

2.2 Princíp duality v lineárnom programovaní

V lineárnom programovaní využívame princíp duality pri riešení jednej z ***dvojice duálnych úloh*** lineárneho programovania

$$\begin{aligned} min\{c^T x: \quad & A_{I_1\times J}x \geq b_{I_1}, \\ & A_{I_2\times J}x \leq b_{I_2}, \\ & A_{I_3\times J}x = b_{I_3}, \\ & x_{J_1} \geq 0, x_{J_2} \leq 0, x \in \mathbb{R}^J\} \end{aligned} \tag{2.9}$$

$$\begin{aligned} max\{b^T y: \quad & (A_{I\times J_1})^T y \leq c_{J_1}, \\ & (A_{I\times J_2})^T y \geq c_{J_2}, \\ & (A_{I\times J_3})^T y = c_{J_3}, \\ & y_{I_1} \geq 0, y_{I_2} \leq 0, y \in \mathbb{R}^I\} \end{aligned} \tag{2.10}$$

kde $c \in \mathbb{R}^J$, $b \in \mathbb{R}^I$, $A \in \mathbb{R}^{I\times J}$, $card(I) = m$, $I_1, I_2, I_3 \subset I$ sú po dvoch disjunktné a $I_1 \cup I_2 \cup I_3 = I$, $card(J) = n$, $J_1, J_2, J_3 \subset J$ sú po dvoch disjunktné a $J_1 \cup J_2 \cup J_3 = J$.

Ku každej úlohe lineárneho programovania je jednoznačne priradená úloha, ktorá spolu s ňou tvorí dvojicu duálnych úloh. Najdôležitejšou vlastnosťou je, že pri vyriešení jednej úlohy z nich, nájdeme taktiež riešenie druhej úlohy z tohto páru, pomocou podmienky komplementarity, viď. veta 2.16.

Označme

$$\mathbf{M} := \{x \in \mathbb{R}^J : A_{I_1 \times J}x \geq b_{I_1}, A_{I_2 \times J}x \leq b_{I_2}, A_{I_3 \times J}x = b_{I_3}, x_{J_1} \geq 0, x_{J_2} \leq 0\}$$
$$\mathbf{N} := \{y \in \mathbb{R}^I : (A_{I \times J_1})^T y \leq c_{J_1}, (A_{I \times J_2})^T y \geq c_{J_2}, (A_{I \times J_3})^T y = c_{J_3}, y_{I_1} \geq 0, y_{I_2} \leq 0\}$$
$$\gamma^* := inf\{c^T x : x \in \mathbf{M}\}$$
$$\delta^* := sup\{b^T x : y \in \mathbf{N}\}$$

Veta 2.16. *Pre dvojicu úloh (2.9), (2.10) a ich prípustné riešenia $x \in \mathbf{M}$ a $y \in \mathbf{N}$ platí $c^T x \geq b^T y$. Pričom rovnosť nastáva iba vtedy, keď sú splnené* ***podmienky komplementarity***

$$\forall j \in J : (A^T y - c)_j = 0 \ \textit{alebo} \ x_j = 0,$$

$$\forall i \in I : (Ax - b)_i = 0 \ \textit{alebo} \ y_i = 0.$$

Veta 2.17. *Ak $\mathbf{M} \neq \emptyset$ a $\mathbf{N} \neq \emptyset$, potom majú obe úlohy (2.9) a (2.10) optimálne riešenie.*

Veta 2.18. *Úloha (2.9) má optimálne riešenie práve vtedy, keď úloha (2.10) má optimálne riešenie. Pokiaľ jedna z týchto úloh má optimálne riešenie, potom platí rovnosť $\gamma^* = \delta^*$.*

Kapitola 3

Metódy riešenia úloh lineárneho programovania

3.1 Simplexová metóda

Simplexová metóda nám umožňuje riešiť úlohy lineárneho programovania zadané v štandardnom tvare

$$min\{c^T x : Ax = b, x \geq 0, x \in \mathbb{R}_J\} \tag{3.1}$$

kde $c \in \mathbb{R}_J, A \in \mathbb{R}_{I \times J}, b \in \mathbb{R}_I$, pričom matica A musí mať plnú riadkovú hodnosť. Simplexová metóda je založená na princípe duality, umožňuje nám efektívne numerické riešenie úloh lineárneho programovania.

Definícia 3.1. *L sa nazýva **báza** úlohy (3.1), ak matica $A_{I \times L}$ je regulárna.*

Definícia 3.2. *Nech L je báza úlohy (3.1), potom vektor $x(L) \in \mathbb{R}^J$ spĺňajúci $\mathcal{N}(x(L)) = J \backslash L$ [1], $A_{I \times L} x(L)_L = b$ nazveme **bázicky primárnym riešením** úlohy (3.1) príslušnej bázy L.*

Definícia 3.3. *Ak L je báza a $x(L)$ je prípustným riešením úlohy (3.1), tj. $x(L) \geq 0$, potom hovoríme, že L je **primárne prípustná báza**.*

[1] kde $\mathcal{N}$ symbolizuje množinu nulových súradníc $\mathcal{N}(x) = \{i \in I : x_i = 0\}$

Definícia 3.4. *Ak L je báza a $x(L)$ je optimálnym riešením úlohy (3.1), potom hovoríme, že L je* ***optimálna báza****.*

Definícia 3.5. *Krajný bod množiny prípustných riešení úlohy v štandardnom tvare nazveme* **nedegenerovaný***, ak má práve $h(A)$ nenulových zložiek.*
Úloha lineárneho programovania v štandardnom tvare sa nazýva **nedegenerovaná***, ak každý krajný bod jej množiny prípustných riešení je nedegenerovaný.*

Simplexový algoritmus

Krok 0: Nájdeme $L_0 \subset J$ primárne prípustnú bázu úlohy (3.1) a ideme na **Krok 1**.
Pokiaľ žiadna prípustná báza úlohy (3.1) neexistuje, potom ideme na **End 1**.

Krok 1: Majme primárne prípustnú bázu $L_k \subset J$.
Spočítame $\delta^T = (c_{L_k})^T (A_{I \times L_k})^{-1} A - c^T \in \mathbb{R}^J$.
Ak $\delta \leq 0$, potom choď na **End 2**, inak choď na **Krok 2**.

Krok 2: Nájdeme index $j \in J$ taký, že $\delta > 0$ a spočítame vektor $\rho = (A_{I \times L_k})^{-1} A_{I \times \{j\}}$.
Ak $\rho \leq 0$, potom prejdeme na **End 3**, inak pokročíme na **Krok 3**.

Krok 3: Nájdeme index $i \in L_k, \rho_i > 0$ taký, že

$$\frac{x(L_k)_i}{\rho_i} = min\left\{ \frac{x(L_k)_u}{\rho_u} : \rho_u > 0, u \in L_k \right\} \tag{3.2}$$

Prevedieme zmenu bázy $L_{k+1} = L_k \cup \{j\} \backslash \{i\}$. Potom L_{k+1} je primárne prípustná báza. Zvýšime $k := k + 1$ a prejdeme na **Krok 1**.

End 1: Úloha (3.1) nemá žiadne prípustné riešenie.

End 2: Vektor $x(L_k)$ je optimálnym riešením úlohy (3.1).

End 3: Účelová funkcia úlohy (3.1) neobmedzene klesá na polpriamke $x(L_k) + t\Delta$, $t \geq 0$, kde

$$\begin{aligned} \Delta_u &= -\rho_u \quad pre \quad u \in L_k \\ &= 1 \quad pre \quad u = j, \\ &= 0 \quad pre \quad u \in J \backslash (L_k \cup \{j\}). \end{aligned}$$

Veta 3.6. *Ak je úloha (3.1) nedegenerovaná, potom sa simplexový algoritmus po konečnom počte krokov zastaví. Buď zistí, že úloha (3.1) nemá prípustné riešenie, alebo nájde optimálnu bázu úlohy (3.1), prípadne nájde smer, v ktorom účelová funkcia úlohy (3.1) neobmedzene klesá.*

3.2 Duálna simplexová metóda

Pomocou duálnej simplexovej metódy sme schopní riešiť úlohu (3.1) hľadaním optimálneho riešenia príslušnej duálnej úlohy k úlohe (3.1), tj.:

$$max\{b^T y : A^T y \leq c, y \in \mathbb{R}\} \tag{3.3}$$

Definícia 3.7. *Ak L je báza úlohy (3.1) a vektor $y(L) \in \mathbb{R}^I$ je prípustným riešením úlohy (3.3), tj. $A^T y \leq c$, potom hovoríme, že L je* ***duálne prípustná báza****.*

Duálny simplexový algoritmus

Krok 0: Nájdeme $L_0 \subset J$ duálne prípustnú bázu úlohy (3.1) a ideme na krok **Krok 1**. Pokiaľ žiadna prípustná báza úlohy (3.1) neexistuje, potom ideme na **End 1**.

Krok 1: Majme duálne prípustnú bázu $L_k \subset J$.
Spočítame $x(L_k)$.
Ak $x(L_k) \leq 0$, potom choď na **End 2**, inak choď na **Krok 2**.

Krok 2: Nájdeme index $i \in L_k$ taký, že $x(L_k)_i < 0$ a spočítame vektor
$\tau^T = ((A_{I \times L_k})^{-1} A)_{\{i\} \times J}$.
Ak $\tau \geq 0$, prejdeme na **End 3**, inak spočítame $\delta^T = (c_{L_k})^T (A_{I \times L_k})^{-1} A - c^T \in \mathbb{R}^J$ a pokročíme na **Krok 3**.

Krok 3: Nájdeme index $j \in J \backslash L_k, \tau_j < 0$ taký, že

$$\frac{\delta_j}{\tau_j} = min\left\{ \frac{\delta_u}{\tau_u} : \tau_u < 0, u \in J \backslash L_k \right\} \tag{3.4}$$

Prevedieme zmenu bázy $L_{k+1} = L_k \cup \{j\} \backslash \{i\}$. Potom je L_{k+1} duálne prípustná báza. Zvýšime $k := k + 1$ a prejdeme na **Krok 1**.

End 1: Úloha (3.1) nemá optimálne riešenie. Nevieme však, či je to preto, že nemá žiadne prípustné riešenie, alebo preto, že jej infimum je $-\infty$.

End 2: Vektor $x(L_k)$ je optimálnym riešením úlohy (3.1).

End 3: Úloha (3.1) nemá žiadne prípustné riešenie a úloha (3.3) má neobmedzený extrém.

Definícia 3.8. *Úloha (3.1) je* ***duálne nedegenerovaná****, pokiaľ pre každú duálne prípustnú bázu L platí:*

$$\begin{aligned} A^T y(L)_j - c_j &= 0 \, pre \, j \in L \\ &< 0 \, pre \, j \in J \backslash L \end{aligned}$$

Veta 3.9. *Ak je úloha (3.1) duálne nedegenerovaná, potom sa duálný simplexový algoritmus po konečnom počte krokov zastaví. Buď nájde optimálnu bázu úlohy (3.1) alebo zistí, že úloha (3.1) nemá optimálne riešenie, poprípade nemá žiadne prípustné riešenie.*

3.3 Metóda zovšeobecnených redukovaných gradientov

Gradientné metódy sú metódy vnútorného bodu určené pre úlohy v tvare $min\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}\}$, kde f má spojité parciálne derivácie a $\mathbf{M} = clo(int(\mathbf{M}))$.

Definujme si funkciu $\varphi(\lambda) = f(\mathbf{x}+\lambda\mathbf{s})$ pre $\lambda \geq 0$, kde $\mathbf{s} \in \mathbb{R}^n$. Funkcia je diferencovateľná a platí $\varphi'(\lambda) = \mathbf{s}^T\nabla f(\mathbf{x}+\lambda\mathbf{s})$, špeciálne pre $\lambda = 0$ platí $\varphi'(0) = \mathbf{s}^T\nabla f(\mathbf{x})$. Teda funkcia φ klesá v bode 0 pokiaľ $\mathbf{s}^T\nabla f(\mathbf{x}) \leq 0$. To znamená, že funkcia f klesá v smere $\mathbf{s}$.

Definícia 3.10. *Povieme, že* $\mathbf{s} \in \mathbb{R}$ *je* ***prípustný smer*** *z bodu* $\tilde{\mathbf{x}} \in \mathbf{M}$ *pre úlohu* $min\{f(x) : x \in \mathbf{M}\}$*, ak platí*

1. $\mathbf{s}^T\nabla f(\mathbf{x}) \leq 0$
2. $\exists\, \varepsilon > 0 : \mathbf{x} + \lambda\mathbf{s} \in \mathbf{M} \quad pre\ 0 \leq \lambda \leq \varepsilon$

Prvá podmienka zaručuje pokles funkcie f pri pohybe z bodu $\mathbf{x}$ v smere $\mathbf{s}$ a druhá, že pri malom pohybe v tomto smere zostávame v množine $\mathbf{M}$. Pre nájdenie minima používame nasledujúci algoritmus.

Gradientná metóda

Krok 1: Zvoľ $x_0 \in \mathbf{M}$.

Krok 2: Urči prípustný smer $\mathbf{s}_t$ z bodu $\mathbf{x}_t$.
Ak prípustný smer neexistuje, potom choď na **Krok 4**. Ak existuje, choď na **Krok 3**.

Krok 3: Generuj nové riešenie $\mathbf{x}_{t+1} = \mathbf{x}_t + \lambda_t \mathbf{s}_t$, kde λ_t (napr.) rieši jednorozmernú úlohu $min\{f(\mathbf{x}_t + \lambda \mathbf{s}_t) : \mathbf{x}_t + \lambda \mathbf{s}_t \in \mathbf{M}, \lambda \geq 0\}$. Polož $t := t + 1$ a choď na **Krok 2**.

Krok 4: Ak je funkcia konvexná a $\mathbf{M}$ je konvexná množina, potom je $\mathbf{x}_t$ optimálne riešenie.

Pre úlohu konvexného programovania platí, že keď sa algoritmus zastaví, potom našiel optimálne riešenie.

3.4 Bariérová metóda

Bariérová metóda je určená pre úlohy v tvare $min\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}\}$, kde $\mathbf{M} \in \mathbb{R}^n$ je uzavretá množina a $\mathbf{M} = clo(int(\mathbf{M}))$. Ďalej máme k dispozícii vhodnú bariérovú funkciu $\Phi : \mathbf{M} \to \mathbb{R}^*$, ktorá spĺňa $\Phi(\mathbf{x}) = +\infty$ pre každé $\mathbf{x}$ z hranice množiny $\mathbf{M}$ a rastie tým, čím bližšie je k hranici množiny $\mathbf{M}$.
Jedná sa o metódu vnútorného bodu. Metóda vyžaduje, aby množina prípustných riešení úlohy mala neprázdne vnútro. Hodí sa preto iba pre nerovnosti $g_j \leq 0$, $\forall j$ a nie je vhodná pre úlohy s rovnicami. Množina prípustných riešení úlohy s rovnosťou má totiž prázdne vnútro.
Metóda je založená na aproximácii zadanej úlohy úlohou

$$min\{f(\mathbf{x}) + \nu\Phi(\mathbf{x}) : \mathbf{x} \in int(\mathbf{M})$$

Veľkosťou ν určujeme vplyv bariérovej funkcie. Ak $\Phi \nearrow \infty$, potom cez ν tlmíme jej nárast, aby ich súčin ostal blízky nule. Funkcia Φ vytvára bariéru, cez ktorú sa nedostaneme - ide o metódu vnútorného bodu.

Bariérová metóda

Krok 1: $\varepsilon > 0$, $\mathbf{x}^1 \in int(\mathbf{M})$ [2], (inak by algoritmus nefungoval), $0 < \beta < 1$, $k := 1$.

Krok 2: Hlavný krok: Nájdeme $\mathbf{x}^{k+1}$ optimálne riešenie úlohy

$$min\{f(\mathbf{x}) + \nu_k \Phi(\mathbf{x}) : \mathbf{x} \in int(\mathbf{M})\}$$

Krok 3: Pokiaľ platí tolerancia $\nu_k \Phi(\mathbf{x}^{k+1}) < \varepsilon$ tak algoritmus končí. Inak položíme $\nu_{k+1} = \beta \nu_k$, $k := k + 1$ a ideme na **Krok 2**.

3.5 Penalizačná metóda

Penalizačná metóda je určená pre úlohy v tvare $min\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}\}$, kde $\mathbf{M} \subset \mathbf{N} \subset \mathbb{R}^n$, $\mathbf{M}$ je uzavretá a $\mathbf{N}$ je otvorená množina. Ďalej máme k dispozícii vhodnú penalizačnú funkciu $\Psi : \mathbf{N} \to \mathbb{R}$, ktorá spĺňa $\Psi(\mathbf{x}) = 0$ pre každé $\mathbf{x} \in \mathbf{M}$ a rastie so vzdialenosťou od množiny $\mathbf{M}$.
Metóda je založená na aproximácii zadanej úlohy úlohou $min\{f(\mathbf{x}) + \mu \mathbf{\Psi}(\mathbf{x}) : \mathbf{x} \in \in \mathbf{N}\}$. Jedná sa o metódu vonkajšieho bodu.

[2] $\mathbf{x}^i$ symbolizuje vektor $\mathbf{x}$ s indexom i označujúcim číslo iterácie, nie mocninu

Penalizačná metóda

Krok 1: $\varepsilon > 0$, $\mathbf{x}^1$ ľubovoľné, $\mu_1 > 0$, $\beta > 1$, $k := 1$.

Krok 2: Hlavný krok: Nájdeme $\mathbf{x}^{k+1}$ optimálne riešenie úlohy

$$min\{f(\mathbf{x}) + \mu_{\mathbf{k}}\mathbf{\Psi}(\mathbf{x}) : \mathbf{x} \in \mathbf{N}\}$$

Krok 3: Pokiaľ platí $\mu_k \Psi(\mathbf{x}^{k+1}) < \varepsilon$ tak algoritmus končí. Inak $\mu_{k+1} = \beta\mu_k$, $k := k + 1$ a ideme na **Krok 2**.

Kapitola 4

Optimalizačný softvér

Na riešenie optimalizačných úloh môžeme využiť rôzne druhy matematického softvéru. V tejto práci sme skúmali vlastnosti dnes v praxi najpoužívanejších softvérov ako sú Mathematica, GAMS, R a Matlab v súvislosti s riešením úlohy lineárneho programovania.

4.1 Mathematica

Programovací jazyk Mathematica obsahuje bohatú ponuku optimalizačných funkcií, ktorých výber záleží na druhu a tvare zápisu úlohy. Pre lineárne programovanie vo vektorovom tvare je najvhodnejšia funkcia *LinearProgramming* [8]. Táto funkcia nám umožňuje zvoliť si druh metódy, pomocou ktorej chceme riešiť úlohu lineárneho programovania. Na výber máme z nasledujúcich metód:

Method $\rightarrow$ "*Simplex*"

Method $\rightarrow$ "*RevisedSimplex*"

Method $\rightarrow$ "*InteriorPoint*"

Ako už názov napovedá, metódy "*Simplex*" a "*RevisedSimplex*" sú založené na simplexovej a upravenej simplexovej metóde. Mathematica odporúča používať tieto metódy pre menšie úlohy zadávané zo vstupu, kedže čas riešenia rastie v závislosti na veľkosti úlohy exponenciálne [8]. Metóda "*InteriorPoint*" je založená na princípe vnútorného bodu a je odporúčaná pre väčšie úlohy, rýchlosť riešenia je lineárna [8].

4.2 GAMS

Program GAMS - The General Algebraic Modeling System [6], je matematický softvér špecializovaný hlavne pre potreby optimalizácie a matematického programovania. GAMS úlohy priamo nerieši, iba spracuváva zadané údaje. Na riešenie úloh používa takzvané riešiče ("solvery"), ktoré sú voliteľné užívateľom podľa druhu úlohy. Tie riešia úlohu predpracovanú programom GAMS pomocou rôznych metód. Pri výbere balíčkov vhodných pre riešenie úloh lineárneho programovania sme čerpali informácie z bakalárskej práce [5], ktorá sa hlbšie zaoberá problematikou lineárneho programovania v jazyku GAMS.

COINIPOPT

Riešič je odporučený pre riešenie úloh nelineárneho programovania avšak, zvláda aj riešenie úloh lineárneho programovania. K nájdeniu riešenia používa metódu "Primal - Dual Interior point Filter Line Search Algorithm", čo je metóda vnútorného bodu.

COINCBC

Riešič COINCBC využíva pre riešenie úloh lineárneho programovania metódu vetvenia a rezu, čo je taktiež jedna z metód vnútorného bodu, obzvlášť výhodná pre úlohy celočíselného programovania.

COINGLPK

Slúži pre riešenie zmiešaných úloh lineárneho programovania, v ktorých sú niektoré premenné celočíselného typu. Pri riešení využíva simplexovú metódu, primárne - duálnu metódu vnútorného bodu a metódu vetvenia a rezu.

CPLEX

CPLEX je primárne určený pre riešenie úloh lineárneho programovania. Je schopný používať viacero metód, ako napr. bariérovú či simplexovú metódu. Prednastavenou metódou je duálna simplexová metóda.

CONOPT

Hlavnou úlohou CONOPTu je riešenie veľkých nelineárnych úloh, pri ktorých využíva metódu zovšeobecnených redukovaných gradientov.

4.3 R

Programovací jazyk R využíva pre riešenie úloh lineárneho programovania rôzne funkcie. Tie sú obsiahnuté vo voľne stiahnuteľných balíčkoch ("packageoch"). Ich výber je závislý, rovnako ako u predchádzajúcich jazykov, na type úlohy a tvare jej zápisu. Pre riešenie úloh lineárneho programovania zapísaných vo vektorovom tvare sú najvhodnejšie balíčky *limSolve* [4] a *lpSolve* [2] s funkciami *linp* a *lp*. Funkcia *linp* je založená na simplexovej metóde a funckia *lp* využíva pri riešení modifikovanú simplexovú metódu.

4.4 Matlab

Základným nástrojom pre optimalizáciu úloh lineárneho programovania softvéru Matlab sú optimalizačné funcie. Tie su rozdelené do skupín pre lineárne, kvadratické, nelineárne a iné špecifické druhy programovania. Pre lineárne programovanie je určená funkcia *linprog* [3]. Tá má prednastavenú metódu riešenia založenú na princípe "Mehrotra's predictor-corrector algorithm", čo je jedna z metód vnútorného bodu. Avšak existuje možnosť voľby aj iných metód. Tie volíme pomocou nastavenia parametru *options* funkcie *linprog* nasledujúcimi príkazmy:

a) $options = optimset('LargeScale','off','Simplex','on')$

b) $options = optimset('LargeScale','off','Simplex','off')$

c) $options = optimset('LargeScale','on','Simplex','off')$

Nastavenie v možnosti *a*) využíva pri riešení simplexovú metódu a je odporučené pre úlohy malého a stredného rozsahu, čo je zrejmé aj z parametru nastavenia $'LargeScale','off'$. Metóda typu *b*) je taktiež odporučená pre malé a stredne veľké úlohy a je založená na princípe aktívnych obmedzení ("Active set"). V prípade *c*) máme možnosť navrátu k už spomínanej prednastavenej metóde vnútorného bodu, ktorá najlepšie vyhovuje riešeniu úloh väčšieho rozsahu.

Kapitola 5

Empirické pozorovanie

5.1 Postup pozorovania

Pri určovaní efektívnosti riešenia jednotlivých softvérov spomenutých v 3. kapitole sme postupovali nasledujúcim spôsobom:

I. Nagenerovali sme vstupné údaje úlohy

II. Previedli sme riešenie úlohy vo všetkých softvéroch

III. Spracovali a porovnali sme získané údaje

I. Generovanie úlohy

Pre meranie časovej efektívnosti jednotlivých softvérov pre úlohu lineárneho programovania sme použili úlohy v štandardnom tvare, čiže v tvare:

$$min\{c^T x : Ax = b, x \geq 0\}$$

Generovali sme štyrikrát po 100 úloh pre rozmery matice 100×100, 500×500, 1000×1000 a 2500×2500 v programovacom jazyku Pascal. Vektor b a maticu A sme generovali tak, aby úloha mala aspoň jedno optimálne riešenie. To sme zaručili pomocou volenia kladných koeficientov v matici A a vo vektore b, čím sme zaručili kompaktnosť množiny prípustných riešení a tým aj existenciu optimálneho

riešenia. Tieto koeficienty sme volili v intervale $(0, 1)$ ktorý je izomorfný množine reálnych čísel. Hodnoty koeficientov účelovej funkcie sme volili v intervale $(-1, 1)$. Dáta sme zapisovali do textového súboru.

II. Riešenie úlohy

Riešenie úloh pre jednotlivé rozmery sme prevádzali v programoch Mathematica, R a Matlab v cykle. V každom cykle sme načítali parametre jednej úlohy z textového súboru a následne previedli meranie časov všetkých dostupných spôsobov riešenia pre daný softvér. Taktiež sme merali aj dĺžku času načítania vstupných parametrov, kedže aj tento údaj ovplyvňuje vhodnosť výberu softvéru pre danú úlohu. V jazyku GAMS nie je možné prevádzať načítavanie vstupných údajov v cykle, iba jednotlivo, v našom prípade pre každú zo 100 úloh samostatne, čo je časovo nesmierne náročné. Preto sme úlohy generovali priamo v jazyku GAMS rovnakým spôsobom ako v jazyku Pascal. Táto zmena nemala dopad na objektivitu merania, kedže ako uvidíme neskôr, rozptyl časov riešenia úloh bol pre každý z piatich rozmerov zanedbateľný, z čoho vyplýva, že vplyv špecifickosti úlohy čo do hodnoty koeficientov vstupných premenných je veľmi malý.

III. Spracovanie údajov

Zo získaných informácií o časovej náročnosti riešenia sme pre jednotlivé programy a im prislúchajúce použité metódy určili maximum, minimum, medián, smerodajnú odchýlku a priemernú hodnotu. Tieto údaje sme navzájom porovnali a vybrali najvhodnejší program a spôsob riešenia pre jednotlivé rozmery úlohy.

Všetky spomenuté operácie v krokoch I., II., a III. sme prevádzali pod operačným systémom Windows XP professional na počítači s procesorom 2.4 GHz Intel Core Duo a veľkosťou operačnej pamäte 2 GB.

5.2 Pozorovanie

V tabuľkách na nasledujúcich stranách sú zobrazené výsledky merania pre jednotlivé softvéry a rozmery úlohy s použitím nasledujúceho štandardného značenia:

$\bar{t}$ - priemerný čas riešenia v sekundách

$\tilde{t}$ - prostredná hodnota (medián) času riešenia v sekundách

$s(t)$ - smerodajná odchýlka času riešenia v sekundách

$min(t)$ - minimálna hodnota času riešenia v sekundách

$max(t)$ - maximálna hodnota času riešenia v sekundách

$\times$ - neschopnosť danej funkcie, riešiča alebo balíčka riešiť úlohu z dôvodu nedostatku alokovanej pamäte pri riešení

dim - rozmer úlohy

$Math.$ - Mathematica

Názov tabuľky označuje rozmer úlohy, na ktorej bolo meranie prevádzané. V poslednom riadku každej tabuľky sú zobrazené informácie o čase načítania údajov do daného programu pri danom rozmere úlohy. V prípade jazyka GAMS sa obmedzíme, z dôvodu spomínaného v časti o riešení úlohy, na údaje o časovej efektivite riešenia, údaje o načítaní vynecháme.

Mathematica

100

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
LinearProgramming	*Simplex*	0.00	0.00	0.01	0.02	0.00
LinearProgramming	*RevisedSimplex*	0.05	0.05	0.01	0.09	0.02
LinearProgramming	*InteriorPoint*	0.10	0.09	0.02	0.16	0.05
spracovanie dát		0.10	0.09	0.02	0.14	0.05

500

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
LinearProgramming	*Simplex*	0.11	0.11	0.02	0.16	0.06
LinearProgramming	*RevisedSimplex*	0.40	0.40	0.10	0.63	0.22
LinearProgramming	*InteriorPoint*	27.46	26.97	6.27	46.61	16.97
spracovanie dát		1.51	1.48	0.17	1.83	1.22

1000

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
LinearProgramming	*Simplex*	0.47	0.45	0.07	0.63	0.34
LinearProgramming	*RevisedSimplex*	1.08	1.08	0.23	1.67	0.63
LinearProgramming	*InteriorPoint*	267.55	260.85	75.17	477.33	128.81
spracovanie dát		5.59	5.59	0.56	7.06	4.55

2500

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
LinearProgramming	*Simplex*	3.82	3.65	0.74	6.55	2.86
LinearProgramming	*RevisedSimplex*	4.73	4.59	0.86	7.56	3.16
LinearProgramming	*InteriorPoint*	×	×	×	×	×
spracovanie dát		35.98	35.56	2.09	42.23	32.13

R

100

Funkcia	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linp	0.04	0.05	0.01	0.07	0.03
lp	0.03	0.03	0.01	0.05	0.01
spracovanie dát	0.17	0.17	0.03	0.24	0.12

500

Funkcia	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linp	0.61	0.59	0.04	0.80	0.56
lp	0.68	0.65	0.07	0.96	0.62
spracovanie dát	3.45	3.39	0.18	4.19	3.26

1000

Funkcia	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linp	2.31	2.28	0.14	2.75	2.11
lp	3.65	3.61	0.15	4.09	3.41
spracovanie dát	18.11	17.98	0.59	20.03	17.07

2500

Funkcia	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linp	12.50	12.39	0.47	13.96	11.63
lp	36.95	36.75	1.00	40.77	35.10
spracovanie dát	129.66	128.16	6.04	152.66	120.62

Matlab

100

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linprog	*Simplex*	0.17	0.17	0.07	0.83	0.11
linprog	*Active set*	0.30	0.31	0.09	1.05	0.20
linprog	*Interior point*	0.28	0.28	0.07	0.84	0.19
spracovanie dát		0.14	0.14	0.03	0.27	0.09

500

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linprog	*Simplex*	3.19	3.14	0.28	3.91	2.69
linprog	*Active set*	17.40	17.36	0.34	18.37	16.77
linprog	*Interior point*	15.75	15.39	2.22	25.53	12.50
spracovanie dát		2.77	2.75	0.26	3.42	2.28

1000

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linprog	*Simplex*	12.063	12.47	1.24	14.55	9.78
linprog	*Active set*	113.34	110.74	7.50	130.97	102.97
linprog	*Interior point*	83.514	67.93	42.76	224.70	23.59
spracovanie dát		10.59	10.91	1.126	12.58	8.58

2500

Funkcia	**Metóda**	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s(t)**	**max(t)**	**min(t)**
linprog	*Simplex*	67.53	64.85	6.10	83.31	61.86
linprog	*Active set*	1167.59	1146.15	54.11	1383.80	1107.10
linprog	*Interior point*	×	×	×	×	×
spracovanie dát		59.23	56.80	5.527	73.16	54.09

GAMS

100

Solver	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s**(**t**)	**max**(**t**)	**min**(**t**)
$COINCBC$	0.02	0.02	0.01	0.05	0.00
$COINGLPK$	0.01	0.02	0.01	0.03	0.00
$COINIPOPT$	0.02	0.02	0.01	0.08	0.00
$CONOPT$	0.01	0.02	0.01	0.03	0.00
$CPLEX$	0.03	0.03	0.01	0.06	0.00

500

Solver	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s**(**t**)	**max**(**t**)	**min**(**t**)
$COINCBC$	0.21	0.21	0.04	0.28	0.17
$COINGLPK$	0.22	0.23	0.03	0.28	0.17
$COINIPOPT$	0.21	0.20	0.03	0.28	0.17
$CONOPT$	0.22	0.22	0.04	0.27	0.17
$CPLEX$	0.21	0.20	0.04	0.28	0.17

1000

Solver	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s**(**t**)	**max**(**t**)	**min**(**t**)
$COINCBC$	0.77	0.77	0.08	1.00	0.67
$COINGLPK$	0.76	0.76	0.07	1.02	0.67
$COINIPOPT$	0.76	0.74	0.07	1.00	0.67
$CONOPT$	0.74	0.70	0.07	1.05	0.67
$CPLEX$	0.76	0.73	0.08	1.00	0.67

2500

Solver	$\bar{\mathbf{t}}$	$\tilde{\mathbf{t}}$	**s**(**t**)	**max**(**t**)	**min**(**t**)
COINCBC	5.87	5.86	0.27	6.70	5.28
COINGLPK	5.95	5.92	0.30	7.19	5.38
COINIPOPT	5.91	5.88	0.34	7.95	5.30
CONOPT	5.92	5.89	0.26	6.63	5.40
CPLEX	5.92	5.89	0.24	6.72	5.36

5.3 Výsledok pozorovania

Zhrnutie hodnotenia efektívnosti riešenia úloh lineárneho programovania môžeme rozdeliť do dvoch kategórií, a to do porovnania jednotlivých optimalizačných riešičov, respektíve funkcií v rámci matematických softvérov a do celkového porovnania matematických softvérov medzi sebou v závislosti na hodnote $\tilde{t}$.

I. Čiastkové hodnotenie

Výsledky porovnania sú znázornené pre jednotlivé softvéry pomocou dvoch skupín tabuliek. V prvej skupine je pre jednotlivé rozmery úlohy uvedený názov najrýchlejšieho spôsobu riešenia v rámci daného softvéru, čas riešenia, časový rozdiel medzi najrýchlejším a druhým najrýchlejším spôsobom riešenia, názov druhého najrýchlejšieho spôsobu riešenia a jeho čas riešenia. Názov tabuliek charakterizuje softvér, ktorého sa uvádzané údaje týkajú.

Mathematica

dim	**1.spôsob**	$\tilde{\mathbf{t}}$	**rozdiel**	**2.spôsob**	$\tilde{\mathbf{t}}$
100	*Simplex*	0.00	0.05	*RevisedSimplex*	0.05
500	*Simplex*	0.11	0.29	*RevisedSimplex*	0.40
1000	*Simplex*	0.45	0.63	*RevisedSimplex*	1.08
2500	*Simplex*	3.65	0.94	*RevisedSimplex*	4.59

R

dim	**1.spôsob**	$\tilde{\mathbf{t}}$	**rozdiel**	**2.spôsob**	$\tilde{\mathbf{t}}$
100	*lp*	0.03	0.02	*linp*	0.05
500	*linp*	0.59	0.06	*lp*	0.65
1000	*linp*	2.28	1.33	*lp*	3.61
2500	*linp*	12.39	24.36	*lp*	36.75

Matlab

dim	**1.spôsob**	$\tilde{\mathbf{t}}$	**rozdiel**	**2.spôsob**	$\tilde{\mathbf{t}}$
100	*Simplex*	0.17	0.11	*Interior point*	0.28
500	*Simplex*	3.14	12.25	*Interior point*	15.39
1000	*Simplex*	12.47	55.46	*Interior point*	67.93
2500	*Simplex*	64.85	1081.30	*Active set*	1146.15

GAMS

dim	**1.spôsob**	$\tilde{\mathbf{t}}$	**rozdiel**	**2.spôsob**	$\tilde{\mathbf{t}}$
100	*CONOPT*	0.02	0.00	*COINGLPK*	0.02
500	*COINIPOPT*	0.20	0.00	*CPLEX*	0.20
1000	*CONOPT*	0.70	0.03	*CPLEX*	0.73
2500	*COINCBC*	5.86	0.02	*COINIPOPT*	5.88

V druhej skupine nájdeme zoradené spôsoby riešenia úlohy s rozmerom 2500 od najrýchlejšieho po najpomalší a ich čas riešenia.

Mathematica

spôsob	$\tilde{\mathbf{t}}$
Simplex	3.65
RevisedSimplex	4.59
InteriorPoint	×

R

spôsob	$\tilde{\mathbf{t}}$
Linp	12.59
lp	36.75

Matlab

spôsob	$\tilde{\mathbf{t}}$
Simplex	64.85
Active	1146.15
Interiorpoint	×

GAMS

spôsob	$\tilde{t}$
CPLEX	5.86
COINIPOPT	5.88
CONOPT	5.89
COINCBC	5.89
COINGLPK	5.92

II. Celkové hodnotenie

Pri celkovom hodnotení použijeme obdobný spôsob ako v predchádzajúcej časti, teda výsledky ilustrujeme pomocou tabuľky obsahujúcej pre každý rozmer úlohy názov najrýchlejšieho softvéru a v rámci daného softvéru najvhodnejší riešič ("solver"), balíček ("package") alebo funkciu, čas jeho riešenia, časový rozdiel medzi najrýchlejším a druhým najrýchlejším softvérom a jemu prislúchajúcim spôsobom riešenia a názov druhého najrýchlejšieho softvéru, spôsob riešenia a jeho čas. Porovnanie rozdelíme do dvoch prípadov a to na porovnanie efektívnosti jednotlivých softvérov v rámci času riešenia úlohy a porovnanie softvérov s prihliadnutím aj na čas načítania vstupných údajov.

Čistý čas riešenia

dim	softvér	spôsob	$\tilde{t}$	rozdiel	softvér	spôsob	$\tilde{t}$
100	*Math.*	*Simplex*	0.00	0.02	*GAMS*	*COINCBC*	0.02
500	*Math.*	*Simplex*	0.13	0.06	*GAMS*	*COINCBC*	0.19
1000	*Math.*	*Simplex*	0.43	0.27	*GAMS*	*CPLEX*	0.70
2500	*Math.*	*Simplex*	3.50	3.16	*GAMS*	*CPLEX*	6.66

Celkový čas riešenia s načítaním

dim	**softvér**	**spôsob**	$\tilde{\mathbf{t}}$	**rozdiel**	**softvér**	**spôsob**	$\tilde{\mathbf{t}}$
100	*Math.*	*Simplex*	0.09	0.11	*R*	*lp*	0.20
500	*Math.*	*Simplex*	1.59	2.39	*R*	*linp*	3.98
1000	*Math.*	*Simplex*	6.04	14.22	*R*	*linp*	20.26
2500	*Math.*	*Simplex*	39.21	82.44	*Matlab*	*Simplex*	121.65

Kapitola 6

Záver

V našej práci sme sa zaoberali problematikou lineárneho programovania. Popísali sme základné poznatky a metódy riešenia lineárnych optimalizačných úloh a zmapovali sme možnosti ich riešenia pomocou rôznych počítačových softvérov. Hlavným zámerom práce bolo porovnať časovú efektivitu riešenia úloh v štandardnom tvare v softvéroch Mathematica, GAMS, R a Matlab v závislosti na veľkosti úlohy. Za týmto účelom sme generovali 100 úloh o rozmeroch 100×100, 500×500, 1000×1000 a 2500×2500, v programovacom jazyku Pascal, na ktorých sme skúmali rýchlosť riešenia jednotlivých softvérov pri použití dostupných riešičov ("solverov"), balíčkov ("packageov") a optimalizačných funkcií. Pri hodnotení časovej efektivity jednotlivých programov sme brali do úvahy aj vplyv rýchlosti načítania vstupných parametrov.
Merania ukázali, že najlepšie výsledky dosiahol softvér Mathematica s optimalizačnou funkciou $LinearProgramming$ a prednastavenou metódou $Simplex$, ktorý dokázal vyriešiť všetky úlohy v najkratšom čase s i bez uvažovania hodnoty času načítania vstupných parametrov. Druhým najrýchlejším softvérom sa stal GAMS, ktorý pre riešenie úloh s rozmerom 100×100 a 1000×1000 využíva riešič $CONOPT$, pre úlohy 500×500 riešič $COINIPOPT$ a 2500×2500 riešič $COIN\text{-}CBC$. V prípade softvéru R je na riešenie úlohy o rozmere 100×100 najvhodnejší balíček *lpSolve* s optimalizačnou funkciou *lp*. Balíček *limSolve* s optimalizačnou funkciou *linp* doporučujeme použiť vo zvyšných prípadoch. Najrýchlejšie riešenie úloh lineárneho programovania v softvéri Matlab je možné dosiahnuť s použitím optimalizačnej funkcie *linprog* s prednastavenou simplexovou metódou.

Zoznam použitej literatúry

[1] P. Lachout, *Matematické programování* - pracovní text k prednášce ,,EKN011 Optimalizace I"zo dňa 26.10.2008

[2] lp solve reference guide menu - *http* : *//lpsolve.sourceforge.net*/5.5/ zo dňa 20.7.2011

[3] Matlab User's Guide-
http : *//www.mathworks.com/help/toolbox/optim/ug/linprog.html* zo dňa 21.7.2011

[4] K. Meersche, D. Oevelen, K. Soetaert, Solving Linear Inverse Models -
http : *//cran.r − project.org/web/packages/limSolve/limSolve.pdf* zo dňa 20.7.2011

[5] K. Murgaš, *Řešení velkých úloh lineárního programování v GAMSu*, Bakalářská práce, MFF UK, 2008

[6] R. Rosenthal, GAMS - A User's Guide
http : *//www.gams.com/dd/docs/bigdocs/GAMSUsersGuide.pdf* zo dňa 16.7.2011

[7] Wikipedia - *http* : *//en.wikipedia.org/wiki/Linear_programming* zo dňa 25.7.2011

[8] Wolfram Mathematica documentation center -
http : *//reference.wolfram.com/mathematica/tutorial/ConstrainedOptimizationLinearProgramming.html* zo dňa 25.7.2011