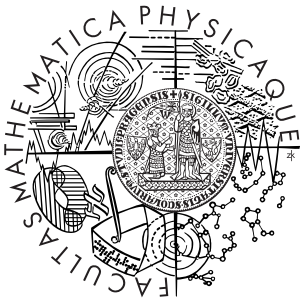
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ladislav Štěpánek

Řešení velkých úloh kvadratického programování v GAMSu

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: RNDr. Ing. Miloš Kopa, Ph.D.
Studijní program: matematika, obecná matematika

2010

Děkuji RNDr. Ing. Milošovi Kopovi, Ph.D. za konzultace, připomínky k práci a poskytnutí počítače k výpočtům, Lucii za korekturu a podporu při psaní a také Radkovi, Jirkovi a přátelům za odreagování v průběhu psaní.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 3.8.2010 Ladislav Štěpánek

Obsah

Název práce: Řešení velkých úloh kvadratického programování v GAMSu
Autor: Ladislav Štěpánek
Katedra (ústav): Katedra pravděpodobnosti a matematické statistiky
Vedoucí bakalářské práce: RNDr. Ing. Miloš Kopa, Ph.D.
e-mail vedoucího: kopa@karlin.mff.cuni.cz

Abstrakt: Program GAMS umožňuje využít několik solverů pro řešení úlohy kvadratického programovaní. V předložené práci studujeme rychlost řešení optimalizační úlohy kvadratického programování různými solvery programu GAMS, které využívají různé metody řešení. V práci dále zkoumáme vliv tvaru účelové funkce a tvaru omezení na rychlost řešení úlohy kvadratického programování různými algoritmy, které jsou ve zkoumaných solverech implementovány. Cílem práce je doporučit vhodný solver či algoritmus pro řešení různých typů úlohy kvadratického programování.

Klíčová slova: kvadratické programování, GAMS, optimalizace

Title: Solving large quadratic problems in GAMS
Author: Ladislav Štěpánek
Department: Department of Prebability and Mathematical Statistics
Supervisor: RNDr. Ing. Miloš Kopa, Ph.D.
Supervisor's e-mail address: kopa@karlin.mff.cuni.cz

Abstract: The program GAMS allows to use several of the solvers for solving quadratic programming problem. In the present work we study solving time of quadratic programming optimization problems in different solvers of program GAMS, which are using different methods. In this work we also explore the influence of shape of objective function and shape of constraints to solving time of quadratic programming problem by different algorithms, which are implemented in surveyed solvers. The goal of this work is to recommend appropriate solver or algorithm for solving different types of quadratic programming problem.

Keywords: quadratic programming, GAMS, optimization

Kapitola 1

Úvod

Kvadratické programování je speciálním případem matematické optimalizace. Podrobně jsou základní problémy optimalizace popsány ve skriptech [8], ze kterých je převzata velká část teorie použité v této práci.

V úloze kvadratického programování minimalizujeme či maximalizujeme hodnotu kvadratické účelové funkce za platnosti lineárních omezení. Tato úloha má široké uplatnění např. ve financích (výběr akciového portfólia pomocí Markowitzova modelu - více v [17]), pro odhad parametrů regresních modelů, ve kterých se minimalizuje čtvercová chyba (OLS-odhady) a také při aproximaci složitějších úloh.

Pro řešení úlohy lze používat několik metod. Tyto metody jsou podrobněji popsány ve třetí kapitole - metody řešení úloh kvadratického programování. Tyto metody se mohou lišit rychlostí a požadavky na systémové prostředky (například paměťové nároky). Jeden z uvažovaných způsobů řešení je pomocí metod nelineárního programování (NLP). Tyto metody mají tu výhodu, že nevyžadují pozitivně semidefinitní matici generující účelovou funkci.

V této práci budeme pozorovat rychlost řešení předem definovaných úloh jednotlivými solvery programu GAMS (Generic Algebraic Modeling System). GAMS umožňuje formulaci různých typů optimalizačních úloh (lineární, nelineární, celočíselné a další), které pak předává předem zvolenému solveru, který úlohu řeší. Více o tomto procesu se lze dočíst v [2].

Pro testování efektivity solverů budeme používat náhodně vygenerované úlohy (vždy je nutno vygenerovat novou úlohu, neboť GAMS si pamatuje řešení posledních úloh a další solvery by úlohu vůbec neřešili). Do času řešení se nepočítá doba nutná pro generování úlohy v prostředí GAMS, ale

pouze čas řešení úlohy daným solverem (neboť v praxi se úloha dodává již pevně zadaná) - tedy od zavolání solveru do vyřešení úlohy. Každý typ úlohy bude každý solver řešit padesetkrát, aby výsledky bylo možné statisticky zpracovat. Cílem je určit vhodný solver pro každý typ úlohy a dále určit, zda a jaký rozdíl je mezi použitím solveru v módu řešení kvadratických (QCP) a nelineárních úloh (NLP) a zda jsou solvery adaptivní a vždy zvolí nejlepší postup. Podrobný popis generovaných úloh a použitých solverů bude uveden dále v práci společně s popisem metod používaných pro řešení úlohy kvadratického programování.

Kapitola 2

Základní definice a věty

2.1 Množiny

Množina přípustných řešení úlohy kvadratického programování je buďto přímo prostor $\mathbb{R}^n$ nebo nějaká jeho podmnožina $\mathbf{M}$. Některé algoritmy mají dodatečné požadavky na tvar množiny přípustných řešení. Zde uvádíme některé typy množin a pojmy, které jsou dále zmiňovány v této práci. Definice a věty jsou převzaty z [8], kde lze také nalézt důkazy zde uváděných tvrzení.

Definice 1. *Bod* $\mathbf{x} \in \mathbf{M}$ *nazveme* **vnitřním bodem** *množiny* $\mathbf{M}$*, jestliže extistuje okolí* $O(\mathbf{x})$ *bodu* $\mathbf{x}$ *takové, že* $O(\mathbf{x}) \subset \mathbf{M}$*.*

Definice 2. *Množina* $\mathbf{M}$ *se nazývá* **otevřená***, jestliže každý bod je vnitřním bodem množiny* $\mathbf{M}$*.*

Definice 3. *Množina* $\mathbf{M}$ *se nazývá* **uzavřená***, jestliže její doplněk je otevřená množina.*

Definice 4. *Sjednocení všech otevřených podmnožin* $\mathbf{M}$ *nazveme* **vnitřkem** *množiny* $\mathbf{M}$ *a značíme* $\mathbf{M}^O$*.*

$$\mathbf{M}^O = \bigcup \{\mathbf{U} \subseteq \mathbb{R}^n : \mathbf{U} \textit{ je otevřená} \wedge \mathbf{U} \subseteq \mathbf{M}\}$$

Definice 5. *Průnik všech uzavřených množin, které obsahují* $\mathbf{M}$ *jako svou podmnožinu, nazveme* **uzávěrem** *množiny* $\mathbf{M}$*, značíme* $\overline{\mathbf{M}}$*.*

$$\overline{\mathbf{M}} = \bigcap \{\mathbf{U} \subseteq \mathbb{R}^n : \mathbf{U} \textit{ je uzavřená} \wedge \mathbf{M} \subseteq \mathbf{U}\}$$

Definice 6. *Bod* $\mathbf{x}$ *se nazývá* **hraničním bodem** *množiny* $\mathbf{M}$*, jestliže každé okolí bodu* $\mathbf{x}$ *obsahuje alespoň 1 bod ležící v množině* $\mathbf{M}$ *a současně alespoň 1 bod neležící v množině* $\mathbf{M}$*. Množina všech hraničních bodů se nazývá* **hranice** *množiny* $\mathbf{M}$ *a označuje se* $\partial\mathbf{M}$*.*

Definice 7. *Řekněme, že množina* $\mathbf{M} \subseteq \mathbb{R}^n$ *je* **konvexní***, jestliže pro každé 2 body* $\mathbf{x}$*,* $\mathbf{y} \in \mathbf{M}$ *a každé* $0 < \lambda < 1$ *platí, že* $\lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in \mathbf{M}$*.*

Definice 8. *Řekněme, že množina* $\mathbf{M} \subseteq \mathbb{R}^n$ *je* **konvexní polyedrická** *množina, jestliže je průnikem konečně mnoha uzavřených poloprostorů.*

2.2 Matice

V této práci jsou použity některé speciální typy matic s jejichž pomocí se pak definují účelová funkce a také množina omezení. Proto zde uvádíme několik poznatků a definic týkajících se matic. Věta o generování pozitivně semidefinitních matic byla v článku [9] uvedena pouze jako tvrzení, tudíž zde provádíme také její důkaz. Definice jsou převzaty z [9].

Definice 9. *Matice* $\mathbf{A}$ *se nazývá* **symetrická***, jestliže* $\mathbf{A} = \mathbf{A^T}$*.*

Definice 10. *Symetrická matice* $\mathbf{A}$ *rozměru* $n \times n$ *se nazývá* **pozitivně semidefinitní***, jestliže pro každý vektor* $\mathbf{x}$ $n \times 1$ *platí:*

$$\mathbf{x^T A x} \geq 0$$

Věta 1. (o generování pozitivně semidefinitních matic) *Nechť* $\mathbf{Z}$ *je libovolná matice rozměrů* $n \times n$*, potom* $\mathbf{C} = \mathbf{ZZ^T}$ *je pozitivně semidefinitní matice rozměru* $n \times n$*.*

Důkaz. Matice $\mathbf{C}$ je symetrická neboť:

$$c_{i,j} = \sum_{k=1}^{n} z_{i,k} z_{j,k} = \sum_{k=1}^{n} z_{j,k} z_{i,k} = c_{j,i}$$

Dále nechť $\mathbf{x}$ je libovolný vektor rozměrů $1 \times$n a označme $\mathbf{y} = \mathbf{x^T Z}$ potom:

$$\mathbf{x^T C x} = \mathbf{x^T Z Z^T x} = \mathbf{y y^T} = \sum_{k=1}^{n} y_i^2 \geq 0$$

Tedy matice $\mathbf{C}$ je pozitivně semidefinitní. □

2.3 Funkce

Vzhledem k tomu, že účelem této práce není zevrubný popis vlastností funkcí, tak zde uvedeme pouze nejdůležitější definici a větu týkající se účelové funkce kvadratického programování. Informace jsou převzaty z [8].

Definice 11. *Pro funkci* $f : \mathbb{R}^n \to \mathbb{R}^*$ *definujeme její* **epigraf** *jako:*

$$epi(f) = \{(\mathbf{x}, \eta) : f(\mathbf{x}) \leq \eta, \mathbf{x} \in \mathbb{R}^n, \eta \in \mathbb{R}\}$$

Definice 12. *Řekněme, že* **funkce** $f : \mathbb{R}^n \to \mathbb{R}^*$ **je konvexní**, *jestliže epi(f) je konvexní množina.*

Věta 2. (vlastnosti konvexní funkcí) *Nechť* $\mathbf{D} \subset \mathbb{R}^n$ *je konvexní množina a* $f : \mathbf{D} \to \mathbb{R}^n$ *má spojité druhé parciální derivace na* $\mathbf{D}^O$. *Pak f je konvexní právě tehdy, když* $\nabla^2_{x,x} f(\mathbf{x}) = (\frac{\partial^n f}{\partial x_i \partial x_j}(\mathbf{x}))^{n,n}_{i=1,j=1}$ *je pozitivně semidefinitní pro každé* $\mathbf{x} \in \mathbf{D}^O$.

Důkaz. Lze nalézt v [8]. □

2.4 Úloha kvadratického programování

Definice 13. Úlohou kvadratického programování *rozumíme:*

$$\min\{\mathbf{p^T x} + \frac{1}{2}\mathbf{x^T C x} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \geq 0\} \tag{2.1}$$

kde $\mathbf{C}$ *je pozitivně semidefinitní matice rozměrů* $n \times n$, *tedy je speciálně symetrická. Vektor* $\mathbf{p}$ *má rozměr* $n \times 1$ *a určuje lineární členy účelové funkce. Matice* $\mathbf{A}$ *rozměrů* $n \times m$ *a vektor* $\mathbf{b}$ *rozměrů* $m \times 1$ *určují množinu přípustných řešení úlohy. n je počet proměnných a m počet omezení.*

Množinu přípustných řešení značíme:

$$\mathbf{M} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \geq 0\} \tag{2.2}$$

Vidíme tedy, že množina $\mathbf{M}$ je konvexní polyedrická množina. Tedy speciálně konvexní a uzavřená. Některé algoritmy konvexnost množiny $\mathbf{M}$ vyžadují pro správnou funkčnost.

Dále funkce f je diferencovatelná na celém $\mathbb{R}^n$ a platí:

$$f(\mathbf{x}) = \mathbf{p}^{\mathbf{T}}\mathbf{x} + \frac{1}{2}\mathbf{x}^{\mathbf{T}}\mathbf{C}\mathbf{x} \tag{2.3}$$

$$\nabla_x f(\mathbf{x}) = \mathbf{p} + \mathbf{C}\mathbf{x} \tag{2.4}$$

$$\nabla^2_{x,x} f(\mathbf{x}) = \mathbf{C} \tag{2.5}$$

Při diferencování je třeba si uvědomit, že matice $\mathbf{C}$ je symetrická.

Důsledek 1. *Vzhledem k pozitivní semidefinitnosti matice* $\mathbf{C}$ *na* $\mathbb{R}^n$ *je podle věty o vlastnostech konvexních funkcí účelová funkce* f *konvexní na* $\mathbf{M}$.

Kapitola 3

Metody řešení úloh kvadratického programování

3.1 Gradientní metoda

Je určena pro řešení úlohy typu:

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}\} \tag{3.1}$$

kde funkce f je konvexní a má spojité parciální defivace a $\mathbf{M}$ je uzavřená konvexní množina. Jedná se o jednu z metodu vnitřního bodu. Uvědome si, že pro obecný směr $\mathbf{s} \in \mathbb{R}^n$ platí následující. Definujeme funkci:

$$\varphi(\lambda) = f(\mathbf{x} + \lambda\mathbf{s}) \text{ pro } \lambda \geq 0 \tag{3.2}$$

Vzhledem k předpokladům na f je funkce φ diferencovatelná a platí:

$$\varphi'(\lambda) = \mathbf{s}^{\mathbf{T}} \nabla f(\mathbf{x} + \lambda\mathbf{s}) \tag{3.3}$$

A speciálně v bodě 0:

$$\varphi(0) = \mathbf{s}^{\mathbf{T}} \nabla f(\mathbf{x}) \tag{3.4}$$

Tedy funkce φ klesá v bodě 0, pokud $\varphi(0) = \mathbf{s^T}\nabla f(\mathbf{x}) < 0$. To znamená, že funkce f klesá ve směru $\mathbf{s}$ v bodě $\mathbf{x}$.

Definice 14. *Řekněme, že* $\mathbf{s} \in \mathbb{R}^n$ *je* **přípustný směr** *z bodu* $\tilde{\mathbf{x}} \in \mathbf{M}$ *pro úlohu* $f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}$*, jestliže platí:*

1. $\varphi(0) = \mathbf{s^T}\nabla f(\tilde{\mathbf{x}}) < 0$

2. $\exists \epsilon$ *¿ 0:* $\tilde{\mathbf{x}} + \lambda\mathbf{s} \in \mathbf{M}$ *pro* $0 \leq \lambda \leq \epsilon$

První podmínka zaručuje pokles funkce f při pohybu z bodu $\tilde{\mathbf{x}}$ ve směru $\mathbf{s}$ a druhá podmínka, že při malém pohybu v tomto směru zůstáváme v množině $\mathbf{M}$. Pro nalezení minima použijeme následující algoritmus:

KROK 1: Zvolíme výchozí řešení $\mathbf{x_0} \in \mathbf{M}$, položíme t=0

KROK 2: Určíme přípustný směr $\mathbf{s_t}$ z bodu $\mathbf{x_t}$.

- Jestli přípustný směr neexistuje, pak jdeme na **KROK 4**
- Existuje-li, pak jdeme na **KROK 3**

KROK 3: Nalezneme nové řešení $\mathbf{x_{t+1}} = \mathbf{x_t} + \lambda_t\mathbf{s_t}$ jako optimální řešení jednorozměrné optimalizační úlohy:

$$\min\{f(\mathbf{x_t} + \lambda\mathbf{s_t}) : \mathbf{x_t} + \lambda\mathbf{s_t} \in \mathbf{M}, \lambda \geq 0\} \tag{3.5}$$

Položíme t:= t + 1 a jdeme na **KROK 2**.

KROK 4: $\mathbf{x_t}$ je optimální řešení.

Pro úlohu konvexního programování (speciálně i pro kvadratické programování) platí, že když se algoritmus zastaví, tak jsme nalezli optimální řešení.

3.2 Wolfeho algoritmus

Frank-Wolfeho algoritmus poprvé navrhnuli Marquerite Frank a Phil Wolfe v roce 1956 v publikaci [12] jako způsob řešení úlohy kvadratického programování. V každém kroku je účelová funkce linearizována a potom je

proveden krok směrem k optimálnímu řešení. Tento krok snižuje hodnotu účelové funkce a přitom se pohybuje v rámci daných omezení. Wolfeho algoritmus je zobecnění simplexového algoritmu (více o simplexovém algoritmu lze nalézt např. v [8]) pro řešení lineárních úloh. Místo simplexového algoritmu je možné použít i duální simplexový algoritmus, kterého využívá právě námi zkoumaný solver CPLEX. Popis duálního simplexového algoritmu lze opět najít v [8] na stranách 70-72.

Wolfeho algoritmus většinou velmi rychle konverguje k hledanému minimu během prvních několika iterací, ale pak se velmi často stává, že se rychlost konvergence značně zpomalí. Tudíž nejlepším využitím tohoto algoritmu jsou přibližná řešení. Obecné informace o Wolfeho algoritmu jsou převzaty z [11] a popis algoritmu z [8].

Pro pochopení fungování algoritmu je nutná znalost základních definic a vět nelineárního programování. Tyto definice lze nalézt opět v publikaci [8]. Zde uvedeme pouze větu a důsledek přímo se týkající Wolfeho algoritmu.

Věta 3. *Nechť* $\mathbf{C}$ *je pozitivně semidefinitní matice. Pak optimální řešení úlohy kvadratického programování (2.1) existuje tehdy a jen tehdy, když existuje* $\mathbf{y}^*$ *tak, že* $(\mathbf{x}^*, \mathbf{y}^*)$ *splňuje:*

$$\mathbf{p} + \mathbf{C}\mathbf{x}^* + \mathbf{A}^\mathbf{T}\mathbf{y}^* \geq 0, \mathbf{x}^* \geq 0, (\mathbf{x}^*)^\mathbf{T}(\mathbf{p} + \mathbf{C}\mathbf{x}^* + \mathbf{A}^\mathbf{T}\mathbf{y}^*) = 0 \tag{3.6}$$

$$\mathbf{A}\mathbf{x}^* - \mathbf{b} \leq 0, \mathbf{y}^* \geq 0, (\mathbf{y}^*)^\mathbf{T}(\mathbf{A}\mathbf{x}^* - \mathbf{b}) = 0 \tag{3.7}$$

Důkaz. Plyne z teorie nelineárního programování a lze jej nalézt v [8] na straně 99. □

Důsledek 2. *Zavedeme-li do vztahů (3.6) a (3.7) skluzové proměnné* $\mathbf{w}$, $\mathbf{v}$ *tak, abychom získali rovnosti:*

$$\mathbf{A}\mathbf{x} + \mathbf{w} = \mathbf{b}, \qquad \mathbf{w} \geq 0 \tag{3.8}$$

$$\mathbf{C}\mathbf{x} + \mathbf{A}^\mathbf{T}\mathbf{y} - \mathbf{v} = -\mathbf{p}, \quad \mathbf{v} \geq 0 \tag{3.9}$$

Pak podmínky komplementarity z podmínek (3.6) a (3.7) přejdou na rovnosti:

$$\mathbf{y}^\mathbf{T}\mathbf{w} = 0, \mathbf{x}^\mathbf{T}\mathbf{v} = 0 \tag{3.10}$$

KROK 0: Nejprve najdeme přípustnou bázi $\mathrm{B} \subset x_1, x_2, ..., x_n, w_1, w_2, ..., w_m$ dimenze m pro nezáporné řešení soustavy $\mathbf{Ax + w = b}$. Když taková primárně přípustná báze neexistuje, pak jdeme na KONEC 1.

KROK 1: Sestavíme pomocnou úlohu:

$$\min \left\{ \sum_{k=1}^{n} z_k : \quad \begin{array}{c} \mathbf{Ax} + \mathbf{w} = \mathbf{b}, \mathbf{Cx} + \mathbf{A^T y} - \mathbf{v} + \mathbf{Dz} = -\mathbf{p} \\ \mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{w} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}, \mathbf{z} \geq \mathbf{0} \end{array} \right\} \tag{3.11}$$

kde **D** je diagonální matice, kterou definujeme následovně:

- je-li $\sum_j c_{k,j}\ x(B)_j > -p_k$ pak $d_{k,k} = -1$,
- je-li $\sum_j c_{k,j}\ x(B)_j \leq -p_k$ pak $d_{k,k} = +1$,

Položíme $L_0 = \mathrm{B} \cup \{z_1, z_2, \ldots, z_n\}$
Pak L_0 je přípustná báze modifikované úlohy (3.11).

KROK 2: Řešíme modifikovanou úlohu (3.11) pomocí simplexového algoritmu s dodatečnými podmínkami pro zařazení proměnných do báze:

- je-li x_k zařazeno v bázi, nesmíme do báze zařadit složku v_k,
- je-li v_k zařazeno v bázi, nesmíme do báze zařadit složku x_k,
- je-li w_j zařazeno v bázi, nesmíme do báze zařadit složku y_j,
- je-li y_j zařazeno v bázi, nesmíme do báze zařadit složku w_j.

Těmito podmínkami je ošetřena komplementarita, která není součástí modifikované úlohy (3.11). Pak je zajištěno, že řešení splňuje podmínky 3.10.

Jako počáteční bázi pro nastartovaní simplexového algoritmu použijeme L_0 z kroku 1.

Takto upravený simplexový algoritmus se vždy zastaví. Označme si konečnou nalezenou bázi $L_{posledni}$.
Když $\sum_{j=1}^{m} y(L_{posledni})_j = 0$, pak jdeme na KONEC 2,
když $\sum_{j=1}^{m} y(L_{posledni})_j > 0$, pak jdeme na KONEC 3.

KONEC 1: Modifikovaná úloha (3.11) nemá přípustné řešení

KONEC 2: Komponenta bazického řešení x($L_{posledni}$) je optimálním řešením modifikované úlohy (3.11)

KONEC 3: Algoritmus nenašel optimální řešení modifikované úlohy (3.11)

Wolfeho algoritmus se sice vždy po konečně krocích zastaví, ale nemusí vždy najít optimální řešení modifikované úlohy (3.11). Nicméně za dodatečné podmínky již vede k optimálnímů řešení:

Věta 4. *Nechť* $\mathbf{C}$ *je pozitivně semidefinitní a* $h(\mathbf{C}|\mathbf{p}) = h(\mathbf{C})$*, pak Wolfeho algoritmus bud' zjistí, že neexistuje přípustné řešení nebo nalezne optimální řešení modifikované úlohy (3.11).*

Důkaz. *Je možné nalézt v článku [10].*

Pro naše úlohy toto bude pravděpodobně splněno neboť generované matice $\mathbf{C}$ jsou s velmi vysokou pravděpodobností plné hodnosti. V případě, že by se ve výsledcích vyskytla chyba plynoucí z tohoto požadavku, bude tento pokus opakován na jiné matici.

3.3 Bariérová metoda

Je určena pro řešení úlohy typu:

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathbf{M}\} \tag{3.12}$$

kde $\mathbf{M} \subset \mathbb{R}^n$ je uzavřená množina s neprázdným vnitřkem. Dále potřebujeme vhodnou bariérovou funkci $\Phi : \mathbf{M} \to \mathbb{R}^*$, která roste, když se přibližujeme hranici množiny $\mathbf{M}$ a splňuje $\Phi(\mathbf{x}) = +\infty$ pro každé $\mathbf{x} \in \partial(\mathbf{M})$. Algoritmus při hledání optimálního řešení prochází vnitřní body množiny $\mathbf{M}$. Jedná se tedy o metodu vnitřního bodu.

Vhodná volba bariérové funkce je například $\Phi\ (\mathbf{x}) = \sum_{j=1}^{m} \phi_j(g_j(\mathbf{x}))$, kde $g_j(\mathbf{x})$ označuje vzdálenost j-té podmínky od uzávěru množiny $\mathbf{M}$ a ϕ je spojitá, nejlépe konvexní funkce, která roste do nekonečna na (-∞ , 0). Vhodné jsou například $-\frac{1}{y}$ *nebo* $-\ log(-y)$.

Algoritmus je založený na aproximaci zadané úlohy úlohou $\min\{f(\mathbf{x}) + \nu\Phi(\mathbf{x}) : \mathbf{x} \in \mathbf{M}^O\}$ Velikostí ν pak volíme vliv bariérové funkce. Jelikož bariérová funkce roste do nekonečna, musí ν tlumit její růst, aby celkový součin zůstal u nuly. Funkce $\Phi(\mathbf{x})$ vytváří bariéru, přes kterou se v algoritmu nelze dostat.

KROK 1: Volíme toleranci metody $\epsilon > 0, \mathbf{x}^1 \in \mathbf{M}^O, 0 < \beta < 1, k := 1, \nu_1 := \nu$.

KROK 2: Nalezneme optimální řešení $\mathbf{x}^{k+1}$ úlohy:

$$\min\{f(\mathbf{x}) + \nu_k \Phi(\mathbf{x}) : \mathbf{x} \in \mathbf{M}^O\} \tag{3.13}$$

KROK 3: Pokud už jsme dosáhli tolerance:

$$\nu_k \Phi(\mathbf{x}^{k+1}) < \epsilon \tag{3.14}$$

pak algoritmus končí. V opačném případě klademe $\nu_{k+1} = \beta\nu_k, k := k + 1$ a opakujeme **KROK 2**.

3.4 Metoda zobecněných redukovaných gradientů (GRG - Generalized Reduced Gradient)

Tato metoda je zobecněním výše popsané gradientní metody pro řešení obecných úloh nelineárního programování. Tudíž pro řešení úlohy kvadratického programování si vystačíme s popisem gradientní metody. Více informací o této metodě se lze dočíst například v [16].

Kapitola 4

Solvery programu GAMS

Program GAMS jako takový úlohu neřeší, pouze ji připraví do předem daného formátu a předá ji zvolenému solveru, který ji následně řeší a předá zpátky výsledek zpracovávané úlohy.

Vybraný solver následně úlohu řeší implementovanými algoritmy. Výsledek je pak předán zpátky do prostředí GAMS, kde je zobrazen uživateli. U většiny solverů je možné nastavit různé parametry pro zpracování úlohy, nicméně pro potřeby této práce jsme ponechali výchozí nastavení.

V této kapitole si ve stručnosti představíme v naší práci používáné solvery a také uvedeme, jaké využívají metody pro řešení úlohy kvadratického programování.

4.1 CPLEX

CPLEX je primárně zaměřen na řešení velkých úloh lineárního a kvadratického programování a nedávno byla dodána i podpora konvexního programování. CPLEX byl vyvinut Robertem Bixbym a distribuován společností ILOG, kterou v roce 2009 převzala IBM.

Pro řešení úloh kvadratického programování může CPLEX využívat libovolnou ze svých metod (Simplexová, Duální simplexová nebo bariérová). Jako výchozí metodu CPLEX používá duální simplexovou metodu. CPLEX může také běžet v paralelním módu (nutná speciální licence), při kterém se daná úloha zpracovává všemi metodami a první, která dokončí vrátí výsledek. Nicméně pro naše účely jsme ponechali výchozí nastavení - duální simplexovou metodu. Více o CPLEXu se lze dočíst v [4].

4.2 COINIPOPT

COIN-IPOPT je solver určený pro velké úlohy nelineárního programování. Zkratka IPOPT (**I**nterior **P**oint **OPT**imizer) skrývá název používané metody a sice metodu vnitřního bodu. COINIPOPT byl naprogramován Carlem Lairdem a Andreasem Wächterem v prostředí C++ a je distribuován pod licencí CPL (Common Public Licence). Tento solver vyžaduje, aby byly všechny funkce nejméně jednou spojitě diferencovatelné (v ideálním případě dvakrát), což naše úloha splňuje. Více o COINIPOPT se lze dočíst na stránkách projektu [5]

4.3 CONOPT

CONOPT je solver určený pro velké úlohy nelineárního programování se speciálním zaměřením na signifikantně nelineární modely a modely s velkým počtem bazických proměnných. Program má verze CONOPT1 - CONOPT3 (vše součástí jednoho licenčního balíčku), přičemž pro drtivou většinu aplikací je doporučeno používat verzi CONOPT3 a tedy ji použijeme i pro řešení úloh v této práci. CONOPT je vyvíjen společností ARKI Consulting and Development a obsahuje mnoho pomocných metod, které doplňují hlavní používanou metodu zobecněných redukovaných gradientů (GRG - Generalized Reduced Gradient).

Některé pomocné metody implementované v solveru CONOPT:

- pomocná metoda klesajících kroků (konverguje rychle ve velké vzdálenosti od optimální hodnoty)
- kvazi-Newtonova metoda
- SLP (Sequential Linear Programming) - využívá lineární aproximace nelineárních úloh
- SQP metoda využívající redukovaný Hessián (při malém množství bazických proměnných) nebo konjugované gradienty (pro velké množství bazických proměnných)

Pomocné metody jsou voleny dynamicky v závislosti na vlastnostech modelu a měření výkonosti jednotlivých metod. Více o tomto solveru je možné najít v [6].

Kapitola 5

Metodika pokusů - účelová funkce

Pro testování efektivity solverů bylo vytvořeno celkem 18 typů úloh. Úlohy byly generovány přímo v prostředí systému GAMS, který obsahuje funkce pro tvorbu náhodných veličin. Při generování úlohy je nutné nastavit takzvaný "seed"ze kterého se pak odvozují veškerá náhodně generovaná čísla. V případě, že by se nenastavil pokaždé na jinou hodnotu, tak by všechny "náhodně"vygenerované úlohy stejného typu byly totožné. Proto je v každé úloze nastaven "seed"na 1 + počet milisekund v aktuálním datumu (číslo 0-99). Tedy je možné, aby se vygenerovaly 2 stejné úlohy za sebou což by se mohlo projevit tím, že by úloha byla vyřešena v neúměrně krátkém času. Tedy bude zajímavé zjišťovat, jestli GAMS i takovéto 2 náhodně generované úlohy rozpozná jako shodné. Více o generování náhodných čísel v GAMS se lze dočíst v [13].

Pro názornost si znovu připomeneme tvar generované úlohy:

$$\min\{\mathbf{p^T x} + \frac{1}{2}\mathbf{x^T C x} : \mathbf{Ax} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \geq 0\} \tag{5.1}$$

Každá úloha kvadratického programování se skládá z pozitivně semidefinitní matice $\mathbf{C}$ určující účelovou funkci, vektoru $\mathbf{p}$ určujícího lineární členy účelové funkce a soustavy nerovnic, které jsou určeny maticí $\mathbf{A}$ a vektorem pravých stran $\mathbf{b}$. V této práci používáme 3 různé typy matice účelové funkce $\mathbf{C}$ a 6 typů matic omezení $\mathbf{A}$. Lineární členy generované vektorem $\mathbf{p}$ úloha obsahuje vždy. Úlohy jsou formulovány tak, aby matice omezení $\mathbf{A}$ měla

vždy 1 000 000 prvků. Tyto prvky jsou náhodně vybírány z rovnoměrného rozdělení na (0,1), dále jen U(0,1) - není nutné volit větší interval nebot' mezi ním a reálnými čísly existuje bijekce. Dále pro zajištení existence přípustného řešení jsme při generování dopočítávali pravou stranu omezení $\mathbf{b}$ tak, aby náhodně vygenerovaný vektor splňoval všechny rovnosti.

Vygenerovaná úloha byla zpracována programem GAMS a předána solveru. Čas se měří od předání úlohy solveru do jejího vyřešení. Toto se řeší standartním GAMSovým výstupním souborem, ve kterém je uveden nejen čas řešení, ale i použitý solver, hodnota řešení, počet iterací, počet prvků a další užitečné informace. Podrobný popis výstupního souboru lze nalézt v [14]. Navíc soubor má formát CSV souboru, takže jej lze velmi jednoduše importovat do dalších programů.

Jednotlivé úlohy jsou popsány níže popsány včetně výsledků dosažených jednotlivými solvery při jejich řešení.

5.1 Typy matic účelových funkcí

V naší práci jsou použity 3 typy matic účelových funkcí $\mathbf{C}$. Jedná se o:

1. obecná matice - $\mathbf{C}$ má (téměř) všechny prvky nenulové
2. řídká matice - $\mathbf{C}$ obsahuje průměrně pouze 10% nenulových prvků
3. blokově diagonální matice - $\mathbf{C}$ je tvořena bloky o náhodné velikosti 1-10 na diagonále, ostatní prvky jsou nulové

Generování matice účelové funkce $\mathbf{C}$ probíhá ve 2 krocích. V prvním kroku je vygenerována pomocná matice s příslušnými vlastnosmi, označme ji $\mathbf{Z}$. Tato matice obsahuje prvky z U(0,1). Ve druhém kroku je vytvořená matice $\mathbf{C}$ dle věty o generování pozitivně semidefinitních matic jako $\mathbf{C} = \mathbf{Z}\mathbf{Z}^\mathbf{T}$. Tímto je zaručena pozitivní semidefinitnost matice $\mathbf{C}$, která je potřebná pro úlohu kvadratického programování. Dále náhodně vygenerujeme vektor $\mathbf{p}$ a získáme tak účelovou funkci tvaru:

$$\mathbf{p}^\mathbf{T}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\mathbf{C}\mathbf{x}, \tag{5.2}$$

kterou budeme minimalizovat.

V dalším kroku vygenerujeme matici omezení $\mathbf{A}$ požadovaného tvaru a pro každý typ matice $\mathbf{A}$ otestujeme všechny matice užitkové funkce $\mathbf{C}$ tak, jak jsou popsány výše.

5.2 Obecná matice účelové funkce

Generování obecné matice účelové funkce, jak je výše uvedeno, probíhá ve 2 krocích. Pro vygenerování obecné matice stačí náhodně vygenerovat matici $\mathbf{Z}$, která má všechny prvky z U(0,1). Matice účelové funkce $\mathbf{C} = \mathbf{Z}\mathbf{Z}^{\mathbf{T}}$ je pak obecná matice, ve které jsou téměř všechny prvky nenulové. Výsledky jednotlivých solverů pro tento typ matice a různé matice omezení jsou uvedeny v tabulce 5.2.

5.3 Řídká matice účelové funkce

Vzhledem k tomu, že potřebujeme pozitivně semidefinitní matici, tak je opět nejprve potřebné vygenerovat matici $\mathbf{Z}$ a následně dle vzorce $\mathbf{C} = \mathbf{Z}\mathbf{Z}^{\mathbf{T}}$ získáme matici $\mathbf{C}$. Zde ovšem vyvstává otázka, jakým způsobem vygenerovat matici $\mathbf{Z}$ tak, aby matice $\mathbf{C}$ měla průměrně právě 10% nenulových prvků. Abychom správně určili počet nenulových prvků v matici $\mathbf{Z}$, je nutné její počet prvků odvodit pomocí teorie pravděpodobnosti. Nenulové prvky matice $\mathbf{Z}$ budeme vybírat z U(0,1).

Nechť p je pravděpodobnost, že prvek matice $\mathbf{Z}$ bude nulový a q=1-p, že prvek bude nenulový. Potom prvek matice $\mathbf{C}$ bude vzhledem k nezápornosti prvků $z_{i,j}$ nulový právě tehdy, když každý člen sumy:

$$c_{i,j} = \sum_{k=0}^{n} z_{i,k} z_{j,k} \tag{5.3}$$

bude nulový. n značí rozměr matice. Nyní musíme rozlišit 2 případy a to i=j a i≠j.

Pro i=j platí:

$$P(c_{i,i} = 0) = P(\textstyle\sum_{k=1}^{n} z_{i,k} z_{i,k} = 0) = P(z_{i,k} = 0, k = 1, 2, \ldots, n) = \\ = \textstyle\prod_{k=1}^{n} P(z_{i,k} = 0) = \prod_{k=1}^{n} p = p^n$$

Třetí rovnost plyne z nezávislosti pravděpodobností, že 2 různé prvky matice $\mathbf{Z}$ jsou nulové. Tedy střední počet nenulových prvků na diagonále matice $\mathbf{C}$ bude n(1-p^n).

Pro $i \neq j$ platí:

$$P(c_{i,j} = 0) = P(\sum_{k=1}^{n} z_{i,k} z_{j,k} = 0) = P(z_{i,k} z_{j,k} = 0, k = 1, 2..n) =$$
$$= \prod_{k=1}^{n} P(z_{i,k} z_{j,k} = 0) = \prod_{k=1}^{n} 1 - P(z_{i,k} z_{j,k} \neq 0) =$$
$$= \prod_{k=1}^{n} 1 - (P(z_{i,k} \neq 0) P(z_{j,k} \neq 0)) = (1 - q^2)^n$$

Zde jsme opět využili nezávislosti pravděpodobností. Tedy střední počet nenulových prvků mimo diagonálu bude (n^2 - n) [1-$(1 - q^2)^n$].

Celkový střední počet nenulových prvků matice **C** tedy bude:

$$n(1 - p^n) + (n^2 - n)[1 - (1 - q^2)^n] \tag{5.4}$$

Nutný počet nenulových prvků v matici **Z** je také závislý na velikosti matice. Tedy musíme pomocí numerického výpočtu určit počty nenulových prvků pro n=500, n=1 000 a n=2 000. Výsledky jsou uvedeny v následující tabulce:

n	p	prvků **C**	nenulových na diag. **C**	nenulových mimo diag. **C**
500	0.01438	250 000	499.64	24 509
1 000	0.0102154	1 000 000	999.96	98 999
2 000	0.00724	4 000 000	1 999.99	397 919

Tabulka 5.1: *Hustota generující matice v závislosti na rozměru matice*

Vidíme tedy, že i při velmi řídké matici **Z** (0.7 - 1.4% nenulových prvků v závislosti na rozměru matice) vznikne matice **C** účelové funkce, která bude mít průměrně 10% nenulových prvků. Výsledky výpočtů pro tento typ matice účelové funkce jsou uvedeny v tabulce 5.3.

5.4 Blokově diagonální matice účelové funkce

Při generování blokově diagonální pozitivně semidefinitní matice pomocí vzorce $\mathbf{C} = \mathbf{Z}\mathbf{Z}^\mathbf{T}$ si stačí uvědomit, že matice **C** bude mít přesně stejné uspořádání bloků jako matice **Z**. Velikost bloků volíme z diskrétního rovnoměrného rozdělení na množině {1,2,...,10}, vygenerujeme matici příslušného

rozměru a tu vložíme na diagonálu matice **Z**. Takto postupujeme, dokud není zaplněna celá diagonála. Prvky matice **Z** jsou z U(0,1). Násobením získáme požadovanou matici **C**. Výsledky jsou uvedeny v tabulce 5.4.

5.5 Výsledky dle matice účelové funkce

Zde uvádíme shrnutí výsledků pokusů rozdělených dle matice účelové funkce do jednotlivých tabulek. V tabulkách je uveden také tvar matice omezení **A**.

Solver	Mód	Matice **A**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	45.23	2.355	44.60	38.98	52.00
CONOPT	NLP	obecná	168.9	13.486	170.2	142.3	200.3
CONOPT	QCP	obecná	172.0	12.645	174.3	141.8	195.4
COINIPOPT	NLP	obecná	896.5	259.440	807.4	802.6	1 813.6
COINIPOPT	QCP	obecná	818.8	23.355	809.7	807.8	908.4
CPLEX	QCP	řídká	13.05	1.802	12.06	11.75	19.05
CONOPT	NLP	řídká	56.47	3.877	56.08	48.59	72.56
CONOPT	QCP	řídká	57.61	2.819	57.95	51.98	63.98
COINIPOPT	NLP	řídká	826.0	112.562	807.2	806.0	1 602.2
COINIPOPT	QCP	řídká	812.2	19.837	807.0	804.6	938.2
CPLEX	QCP	blok. diag.	8.84	0.514	9.00	8.04	9.84
CONOPT	NLP	blok. diag.	4.69	0.189	4.64	4.61	5.59
CONOPT	QCP	blok. diag.	4.72	0.039	4.71	4.69	4.89
COINIPOPT	NLP	blok. diag.	1.649	0.279	1.516	1.485	2.563
COINIPOPT	QCP	blok. diag.	1.685	0.221	1.579	1.562	2.188
CPLEX	QCP	záv. řádky	80.95	4.077	80.19	75.59	95.89
CONOPT	NLP	záv. řádky	127.4	5.042	128.0	113.6	137.5
CONOPT	QCP	záv. řádky	133.4	13.149	131.4	115.0	201.7
COINIPOPT	NLP	záv. řádky	1 615	18.502	1 510	1 608	1 701
COINIPOPT	QCP	záv. řádky	1 009	5.795	1 009	1 000	1 021
CPLEX	QCP	více řádků	35.35	0.583	35.45	32.53	35.64
CONOPT	NLP	více řádků	38.70	2.077	39.42	35.53	42.75
CONOPT	QCP	více řádků	39.12	1.594	39.57	35.66	42.67
COINIPOPT	NLP	více řádků	NA	NA	NA	NA	NA
COINIPOPT	QCP	více řádků	NA	NA	NA	NA	NA
CPLEX	QCP	více sloupců	180.2	5.824	184.8	173.0	184.9
CONOPT	NLP	více sloupců	632.6	115.452	603.5	463.0	876.7
CONOPT	QCP	více sloupců	491.4	26.06	483.1	456.2	526.7
COINIPOPT	NLP	více sloupců	12 733	174.744	12 651	12 611	13 268
COINIPOPT	QCP	více sloupců	12 763	129.547	12 703	12 691	13 019

Tabulka 5.2: *Výsledky pro obecnou matici účelové funkce*

Solver	Mód	Matice **A**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	38.41	4.304	37.60	33.09	48.81
CONOPT	NLP	obecná	170.6	10.158	171.3	146.0	194.3
CONOPT	QCP	obecná	171.4	11.143	171.4	150.1	208.3
COINIPOPT	NLP	obecná	22.07	4.494	20.23	19.08	34.66
COINIPOPT	QCP	obecná	20.31	1.348	19.75	18.95	25.28
CPLEX	QCP	řídká	12.16	0.297	12.15	11.64	12.70
CONOPT	NLP	řídká	53.40	2.768	53.32	47.44	59.91
CONOPT	QCP	řídká	53.80	2.686	53.32	49.92	59.83
COINIPOPT	NLP	řídká	16.92	3.064	16.17	15.00	28.70
COINIPOPT	QCP	řídká	16.42	1.515	15.98	14.98	21.42
CPLEX	QCP	blok. diag.	8.139	0.173	8.164	7.734	8.500
CONOPT	NLP	blok. diag.	1.119	0.025	1.125	1.070	1.164
CONOPT	QCP	blok. diag.	1.121	0.022	1.125	1.078	1.156
COINIPOPT	NLP	blok. diag.	16.37	4.477	14.80	12.23	27.42
COINIPOPT	QCP	blok. diag.	15.07	1.157	14.97	13.11	18.92
CPLEX	QCP	záv. řádky	63.42	4.331	62.68	56.06	72.08
CONOPT	NLP	záv. řádky	130.3	5.569	130.8	119.4	141.5
CONOPT	QCP	záv. řádky	129.2	5.065	128.3	116.1	145.4
COINIPOPT	NLP	záv. řádky	63.92	15.092	68.66	26.28	124.83
COINIPOPT	QCP	záv. řádky	51.20	9.263	52.88	39.27	74.77
CPLEX	QCP	více řádků	30.90	2.212	30.82	27.25	34.98
CONOPT	NLP	více řádků	37.89	1.825	38.45	34.97	40.92
CONOPT	QCP	více řádků	38.34	1.971	39.07	34.64	44.00
COINIPOPT	NLP	více řádků	NA	NA	NA	NA	NA
COINIPOPT	QCP	více řádků	NA	NA	NA	NA	NA
CPLEX	QCP	více sloupců	133.1	10.514	133.3	113.6	154.3
CONOPT	NLP	více sloupců	451.0	37.807	442.3	407.6	652.0
CONOPT	QCP	více sloupců	450.8	38.670	444.5	396.9	648.8
COINIPOPT	NLP	více sloupců	1085	29.548	1085	1001	1138
COINIPOPT	QCP	více sloupců	334.9	14.948	330.9	314.3	388.9

Tabulka 5.3: *Výsledky pro řídkou matici účelové funkce*

Solver	Mód	Matice **A**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	29.62	0.343	29.57	29.09	30.45
CONOPT	NLP	obecná	167.2	7.764	166.8	152.0	184.2
CONOPT	QCP	obecná	167.2	8.194	165.1	148.4	185.3
COINIPOPT	NLP	obecná	9.85	1.129	9.484	9.48	13.28
COINIPOPT	QCP	obecná	11.21	0.983	10.94	10.67	14.58
CPLEX	QCP	řídká	2.31	0.025	2.31	2.25	2.39
CONOPT	NLP	řídká	21.52	1.882	21.71	17.09	26.36
CONOPT	QCP	řídká	22.22	1.558	22.22	17.08	26.39
COINIPOPT	NLP	řídká	6.41	0.915	6.23	5.76	9.50
COINIPOPT	QCP	řídká	6.71	1.141	6.26	5.843	9.56
CPLEX	QCP	blok. diag.	0.058	0.007	0.062	0.046	0.062
CONOPT	NLP	blok. diag.	0.769	0.014	0.766	0.734	0.797
CONOPT	QCP	blok. diag.	0.756	0.017	0.766	0.719	0.797
COINIPOPT	NLP	blok. diag.	0.111	0.019	0.109	0.093	0.188
COINIPOPT	QCP	blok. diag.	0.110	0.009	0.109	0.093	0.141
CPLEX	QCP	záv. řádky	37.25	1.510	36.42	34.55	41.75
CONOPT	NLP	záv. řádky	137.3	12.683	141.7	111.4	159.7
CONOPT	QCP	záv. řádky	111.4	5.630	112.2	95.08	125.6
COINIPOPT	NLP	záv. řádky	33.89	13.261	31.19	18.50	73.28
COINIPOPT	QCP	záv. řádky	35.76	10.601	31.25	18.55	74.19
CPLEX	QCP	více řádků	31.44	1.972	32.41	28.14	33.75
CONOPT	NLP	více řádků	40.81	1.777	40.36	36.22	48.67
CONOPT	QCP	více řádků	37.13	2.743	36.22	34.58	41.03
COINIPOPT	NLP	více řádků	NA	NA	NA	NA	NA
COINIPOPT	QCP	více řádků	NA	NA	NA	NA	NA
CPLEX	QCP	více sloupců	20.88	0.834	20.38	20.14	23.48
CONOPT	NLP	více sloupců	407.9	99.089	410.5	289.1	663.6
CONOPT	QCP	více sloupců	133.5	4.491	132.8	126.4	143.8
COINIPOPT	NLP	více sloupců	1 009.3	7.677	1 009.9	972.6	1 024.3
COINIPOPT	QCP	více sloupců	166.5	8.106	165.7	144.8	181.3

Tabulka 5.4: *Výsledky pro blokově diagonální matici účelové funkce*

Kapitola 6

Metodika pokusů - tvar omezení

6.1 Typy matic omezení

Na matici omezení nejsou kladeny žádné specifické požadavky jako na matici určující tvar účelové funkce. Tudíž tato matice je generována přímo dle požadavků pro jednotlivé typy úloh. Typy matic omezení jsou stejné nebo podobné jako v bakalářské práci ([7]), aby bylo možné porovnání doby řešení úlohy lineárního a kvadratického programování. Všechny matice omezení jsou stejně jako v práci pana Murgaše generovány tak, aby měli 1 000 000 prvků. Vlastnosti matic jsou shodné s vyjímkou řídké matice omezení. U řídké matice omezení byl zvolen pevný počet nenulových prvků v řádku oproti proměnlivému. V této práci je použito 6 typů matic omezení **A**, z nichž typy 1-4 jsou čtvercové matice a 5-6 obdélníkové. Jedná se o:

1. obecná matice - **A** má (téměř) všechny prvky nenulové

2. řídká matice - **A** obsahuje průměrně pouze 5% nenulových prvků

3. blokově diagonální matice - **A** je tvořena bloky o náhodné velikosti 1×1 až 10×10 na diagonále, ostatní prvky jsou nulové

4. matice se závislými řádky - **A** je generována tak, aby měla hodnost maximálně 500

5. matice s více řádky - **A** má 2 000 řádků a 500 sloupců, úloha má 2 000 proměnných

6. matice s více sloupci - **A** má 500 řádků a 2 000 sloupců, úloha má 500 proměnných

Takto vygenerovaná matice omezení pak bude určovat množinu přípustných řešení **M**:

$$\mathbf{M} = \{\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \geq 0, \mathbf{Ax} = \mathbf{b}\} \tag{6.1}$$

6.2 Obecná matice omezení

Matice **A** má rozměry 1 000×1 000. Její jednotlivé prvky jsou vybírány z U(-1,1). Existenci nejméně jednoho řešení i v případě singulární matice jsme zajistili vhodnou volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.1.

Solver	Mód	Matice **C**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	45.23	2.355	44.60	38.98	52.00
CONOPT	NLP	obecná	168.9	13.486	170.2	142.3	200.3
CONOPT	QCP	obecná	172.0	12.645	174.3	141.8	195.4
COINIPOPT	NLP	obecná	896.5	259.440	807.4	802.6	1 813.6
COINIPOPT	QCP	obecná	818.8	23.355	809.7	807.8	908.4
CPLEX	QCP	řídká	38.41	4.304	37.60	33.09	48.81
CONOPT	NLP	řídká	170.6	10.158	171.3	146.0	194.3
CONOPT	QCP	řídká	171.4	11.143	171.4	150.1	208.3
COINIPOPT	NLP	řídká	22.07	4.494	20.23	19.08	34.66
COINIPOPT	QCP	řídká	20.31	1.348	19.75	18.95	25.28
CPLEX	QCP	blok. diag.	29.62	0.343	29.57	29.09	30.45
CONOPT	NLP	blok. diag.	167.2	7.764	166.8	152.0	184.2
CONOPT	QCP	blok. diag.	167.2	8.194	165.1	148.4	185.3
COINIPOPT	NLP	blok. diag.	9.85	1.129	9.484	9.48	13.28
COINIPOPT	QCP	blok. diag.	11.21	0.983	10.94	10.67	14.58

Tabulka 6.1: Výsledky pro obecnou matici omezení

6.3 Řídká matice omezení

Matice **A** má opět rozměry 1 000×1 000. Její prvky jsou generovány tak, aby měla přibližně pouze 5% nenulových prvků. Tedy nejprve jsme náhodně

vygenerovali číslo h z U(0,1) určující zda bude daný prvek nulový nebo nenulový. Pokud je $h \leq 0.05$, pak na místo prvku dosadíme náhodně vybrané číslo z U(-1,1), v opačném případě dosadíme 0. Tato úloha by měla být snáze řešitelná, neboť úloha obsahuje menši množství omezujících podmínek. Vzhledem k množství prvků je velmi nepravděpodobné, aby matice měla výrazně jiný počet nenulových prvků než právě 5%. Existenci alespoň jednoho řešení je opět zajištěna vhodnou volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.2.

Solver	Mód	Matice **C**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	13.05	1.802	12.06	11.75	19.05
CONOPT	NLP	obecná	56.47	3.877	56.08	48.59	72.56
CONOPT	QCP	obecná	57.61	2.819	57.95	51.98	63.98
COINIPOPT	NLP	obecná	826.0	112.562	807.2	806.0	1 602.2
COINIPOPT	QCP	obecná	812.2	19.837	807.0	804.6	938.2
CPLEX	QCP	řídká	12.16	0.297	12.15	11.64	12.70
CONOPT	NLP	řídká	53.40	2.768	53.32	47.44	59.91
CONOPT	QCP	řídká	53.80	2.686	53.32	49.92	59.83
COINIPOPT	NLP	řídká	16.92	3.064	16.17	15.00	28.70
COINIPOPT	QCP	řídká	16.42	1.515	15.98	14.98	21.42
CPLEX	QCP	blok. diag.	2.31	0.025	2.31	2.25	2.39
CONOPT	NLP	blok. diag.	21.52	1.882	21.71	17.09	26.36
CONOPT	QCP	blok. diag.	22.22	1.558	22.22	17.08	26.39
COINIPOPT	NLP	blok. diag.	6.41	0.915	6.23	5.76	9.50
COINIPOPT	QCP	blok. diag.	6.71	1.141	6.26	5.843	9.56

Tabulka 6.2: Výsledky pro řídkou matici omezení

6.4 Blokově diagonální matice omezení

Matice **A** má opět rozměry 1 000×1 000. Matice má velice specifický tvar, kdy na diagonále jsou umístěny matice o velikosti 1×1 až 10×10. Tato matice má velice výhodný tvar neboť úlohu v podstatě rozkládá na podúlohy o velikosti jednotlivých bloků. Tedy v případě, že by některý solver dokázal využít této vlastnosti, tak by neřešil jednu úlohu o 1 000 proměnných ale řádově několik set úloh o 1 - 10 proměnných. Samotná matice je generována tak, že se nejprve zvolí náhodná velikost bloku z diskrétního rovnoměrného

rozdělení na množině {1,2,. . . ,10}, vygeneruje se matice příslušných rozměrů a ta se vloží na diagonálu matice omezení **A**. Takto se postupuje dokud není zaplněna celá diagonála. Existenci alespoň jednoho řešení je opět zajištěna vhodnou volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.3.

Solver	Mód	Matice **C**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	8.84	0.514	9.00	8.04	9.84
CONOPT	NLP	obecná	4.69	0.189	4.64	4.61	5.59
CONOPT	QCP	obecná	4.72	0.039	4.71	4.69	4.89
COINIPOPT	NLP	obecná	1.649	0.279	1.516	1.485	2.563
COINIPOPT	QCP	obecná	1.685	0.221	1.579	1.562	2.188
CPLEX	QCP	řídká	8.139	0.173	8.164	7.734	8.500
CONOPT	NLP	řídká	1.119	0.025	1.125	1.070	1.164
CONOPT	QCP	řídká	1.121	0.022	1.125	1.078	1.156
COINIPOPT	NLP	řídká	16.37	4.477	14.80	12.23	27.42
COINIPOPT	QCP	řídká	15.07	1.157	14.97	13.11	18.92
CPLEX	QCP	blok. diag.	0.058	0.007	0.062	0.046	0.062
CONOPT	NLP	blok. diag.	0.769	0.014	0.766	0.734	0.797
CONOPT	QCP	blok. diag.	0.756	0.017	0.766	0.719	0.797
COINIPOPT	NLP	blok. diag.	0.111	0.01	0.109	0.093	0.188
COINIPOPT	QCP	blok. diag.	0.110	0.009	0.109	0.093	0.141

Tabulka 6.3: Výsledky pro blokově diagonální matici omezení

6.5 Matice omezení se závislými řádky

Matice **A** má opět rozměry 1 000×1 000. Nicméně obsahuje nejvýše 500 lineárně nezávislých řádků. Matice je generována tak, že se vygenerují 2 matice $\mathbf{M_1}$ a $\mathbf{M_2}$ o rozměrech 1 000×500 a 500×1 000, které opět obsahují náhodné prvky z U(-1,1). Následně položíme $\mathbf{A} = \mathbf{M_1}\mathbf{M_2}$ a tím získáme matici 1 000×1 000 s maximálně 500 lineárně nezávislými řádky (opět poznatek z lineární algebry). Tato úloha bude mít nejméně 500 stupňu volnosti a tudíž existuje více řešení při daných podmínek a budou se moci uplatnit algoritmy využívající metody vnitřního bodu. V každém případě je existence řešení opět zajištěna vhodnou volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.4.

Solver	Mód	Matice **C**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	80.95	4.077	80.19	75.59	95.89
CONOPT	NLP	obecná	127.4	5.042	128.0	113.6	137.5
CONOPT	QCP	obecná	133.4	13.149	131.4	115.0	201.7
COINIPOPT	NLP	obecná	1 615	18.502	1 510	1 608	1 701
COINIPOPT	QCP	obecná	1 009	5.795	1 009	1 000	1 021
CPLEX	QCP	řídká	63.42	4.331	62.68	56.06	72.08
CONOPT	NLP	řídká	130.3	5.569	130.8	119.4	141.5
CONOPT	QCP	řídká	129.2	5.065	128.3	116.1	145.4
COINIPOPT	NLP	řídká	63.92	15.092	68.66	26.28	124.83
COINIPOPT	QCP	řídká	51.20	9.263	52.88	39.27	74.77
CPLEX	QCP	blok. diag.	37.25	1.510	36.42	34.55	41.75
CONOPT	NLP	blok. diag.	137.3	12.683	141.7	111.4	159.7
CONOPT	QCP	blok. diag.	111.4	5.630	112.2	95.08	125.6
COINIPOPT	NLP	blok. diag.	33.89	13.261	31.19	18.50	73.28
COINIPOPT	QCP	blok. diag.	35.76	10.601	31.25	18.55	74.19

Tabulka 6.4: Výsledky pro matici omezení se závislými řádky

6.6 Matice omezení s více řádky než sloupci

Abychom zachovali počet prvků matice omezení **A** na 1 000 000, tak má nyní matice omezení 2 000 řádků a 500 sloupců. S tím ovšem přichází velká změna i v definici účelové funkce, neboť nyní je použito pouze 500 proměnných a tedy matice určující účelovou funkci má rozměry pouze 500×500. Toto úlohu velmi zjednodušuje, ale algoritmus se i tam bude muset vypořádat s 2 000 lineárními omezeními. V této úloze bude opět matice **A** lineárně závislá - matici budeme opět zaplňovat náhodně prvky z U(-1,1) a je velice pravděpodobné, že bude mít hodnost právě 500. Existence přípustného řešení bude opět zajištěna volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.5.

Solver	Mód	Matice **C**	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	35.35	0.583	35.45	32.53	35.64
CONOPT	NLP	obecná	38.70	2.077	39.42	35.53	42.75
CONOPT	QCP	obecná	39.12	1.594	39.57	35.66	42.67
COINIPOPT	NLP	obecná	NA	NA	NA	NA	NA
COINIPOPT	QCP	obecná	NA	NA	NA	NA	NA
CPLEX	QCP	řídká	30.90	2.212	30.82	27.25	34.98
CONOPT	NLP	řídká	37.89	1.825	38.45	34.97	40.92
CONOPT	QCP	řídká	38.34	1.971	39.07	34.64	44.00
COINIPOPT	NLP	řídká	NA	NA	NA	NA	NA
COINIPOPT	QCP	řídká	NA	NA	NA	NA	NA
CPLEX	QCP	blok. diag.	31.44	1.972	32.41	28.14	33.75
CONOPT	NLP	blok. diag.	40.81	1.777	40.36	36.22	48.67
CONOPT	QCP	blok. diag.	37.13	2.743	36.22	34.58	41.03
COINIPOPT	NLP	blok. diag.	NA	NA	NA	NA	NA
COINIPOPT	QCP	blok. diag.	NA	NA	NA	NA	NA

Tabulka 6.5: Výsledky pro matici omezení s více řádky než sloupci

6.7 Matice omezení s více sloupci než řádky

Abychom zachovali počet prvků matice omezení **A** na 1 000 000, tak má nyní matice omezení 500 řádků a 2 000 sloupců. S tím ovšem přichází velká změna i v definici účelové funkce, neboť nyní je použito dokonce 2 000 proměnných a tedy matice určující účelovou funkci má rozměry 2 000×2 000 - tedy 4 000 000 prvků. Účelová funkce má tedy čtyřikrát více prvků a dá se tedy předpokládat, že řešení bude trvat déle než v předchozích případech. Na druhou stranu bude mít matice omezení **A** pouze 500 řádků a tedy bude exitovat více řešení a opět se mohou uplatnit algoritmy využívající metody vnitřního bodu. Existence nejméně jednoho řešení je zajištěna vhodnou volbou vektoru **b**. Výsledky jsou uvedeny v tabulce 6.6.

Solver	Mód	Matice $\mathbf{C}$	$\bar{x}$	s	$\tilde{x}$	min	max
CPLEX	QCP	obecná	180.2	5.824	184.8	173.0	184.9
CONOPT	NLP	obecná	632.6	115.452	603.5	463.0	876.7
CONOPT	QCP	obecná	491.4	26.06	483.1	456.2	526.7
COINIPOPT	NLP	obecná	12 733	174.744	12 651	12 611	13 268
COINIPOPT	QCP	obecná	12 763	129.547	12 703	12 691	13 019
CPLEX	QCP	řídká	133.1	10.514	133.3	113.6	154.3
CONOPT	NLP	řídká	451.0	37.807	442.3	407.6	652.0
CONOPT	QCP	řídká	450.8	38.670	444.5	396.9	648.8
COINIPOPT	NLP	řídká	1085	29.548	1085	1001	1138
COINIPOPT	QCP	řídká	334.9	14.948	330.9	314.3	388.9
CPLEX	QCP	blok. diag.	20.88	0.834	20.38	20.14	23.48
CONOPT	NLP	blok. diag.	407.9	99.089	410.5	289.1	663.6
CONOPT	QCP	blok. diag.	133.5	4.491	132.8	126.4	143.8
COINIPOPT	NLP	blok. diag.	1 009.3	7.677	1 009.9	972.6	1 024.3
COINIPOPT	QCP	blok. diag.	166.5	8.106	165.7	144.8	181.3

Tabulka 6.6: Výsledky pro matici omezení s více slouci než řádky

Kapitola 7

Závěr

Výsledky testů ukázali, že mezi solvery jsou řádové rozdíly (až 70×) a proto je velice důležité vybrat správný solver. Ve většině testů velmi přesvědčivě zvítězil solver CPLEX s duální simplexovou metodou. Pouze v případech řídkých či blokově diagonálních matic omezení či účelové funkce byl lepší solver COINIPOPT s metodou vnitřního bodu. Nicméně zde je třeba postupovat velmi opatrně, neboť pokud nebude mít úloha správný tvar účelové funkce a správný tvar omezení, tak byl COINIPOPT ještě výrazně pomalejší než již tak velmi pomalý CONOPT a nedokáže řešit úlohy s větším počtem omezení než proměnných. Navíc COINIPOPT je rychlejší v případech velmi řídkých matic, které všechny solvery zvládly vyřešit v porovnání s ostatními úlohami velmi rychle (CPLEX cca 8.8s - COINIPOPT cca 1.6s). Naopak při velmi složitých úlohách byl CPLEX mnohem rychlejší (např. pro 2 000 proměnných při plných maticích CPLEX cca 180s - COINIPOPT cca 12 763s!!). Proto bychom obecně doporučili pro řešení úlohy kvadratického programování používat solver CPLEX a pouze ve vyjímečných případech, kdy přesně známe strukturu obou matic, bychom doporučili využívání solveru COINIPOPT.

Solver CONOPT s metodou redukovaných gradientů se většinou umisťoval na 2. místě a je v průměru 2×až 5× pomalejší než nejrychlejší solver pro danou úlohu. Pouze ve výjímečných případech se nejlepšímu solveru vyrovnal či jej dokonce lehce předstihl. Nicméně opět se jednalo o úlohy které se řeší velmi rychle a náskok nikdy nebyl tak výrazný jako v případě ostatních solverů. Z těchto důvodů bychom solver CONOPT nedoporučili pro řešení úloh kvadratického programování.

U solverů CONOPT a COINIPOPT jsme zkoumali, zda je rozdíl mezi

řešením úlohy jako kvadratické (QCP) či jako obecné nelineární (NLP). I přes to, že ve většině případů jsou časy řešení srovnatelné, tak u některých úloh byl solver při řešení úlohy jako nelineární výrazně pomalejší. Proto je vhodné nastavit solveru, aby úlohu řešil jako úlohu kvadratického programování.

Posledním cílem této práce bylo porovnat časovou náročnost úlohy lineárního a kvadratického programování. Z výsledků uvedených v bakalářské práci [7] vyplývá, že řešení úlohy lineárního programování trvá maximálně 1.45 sekundy a to i v případě úlohy s 2 000 proměnnými. Úlohy s řídkou a blokově diagonální maticí omezení byly řešeny dokonce v časech 0.09 sekundy respektive 0.02 sekundy. V porovnání s časy řešení úlohy kvadratického programování se jedná o řádově nižší časy. Navíc u kvadratického programování má velký vliv počet proměnných. U lineárního programování záleží především na množství prvků matice omezení.

Výsledky této práce ukazují, že by bylo velmi zajímavé zkoumat rychlost výpočtu v závislost na velikosti vstupních matic. Nebo také maximální velikosti vstupních matic či množství nenulových prvků v maticích, které je ještě daný solver schopen vypočítat za předem určený časový úsek. Nicméně tyto dodatečné možnosti značně zvyšují již tak velmi vysokou výpočetní náročnost a proto je nebylo možné zahrnout do našich experimentů.

Literatura

[1] Wikipedia: *http://en.wikipedia.org/wiki/Quadratic_programming*

[2] GAMS Development Corporation: GAMS - The Solver Manual, 2008: *http://www.gams.com/dd/docs/solvers/alphaecp.pdf*

[3] GAMS Development Corporation: GAMS - A User's Guide, 2008: *http://www.gams.com/dd/docs/bigdocs/GAMSUsersGuide.pdf*

[4] Manuál solveru CPLEX: *http://www.gams.com/dd/docs/solvers/cplex.pdf*

[5] Manuál solveru COINIPOPT: *https://projects.coin-or.org/Ipopt*

[6] Manuál solveru CONOPT: *http://www.gams.com/dd/docs/solvers/conopt.pdf*

[7] Karel Murgaš: Řešení velkých úloh lineárního programování v GAMSu, Bakalářská práce, MFF UK, 2008

[8] Doc. RNDr. Petr Lachout, CSc.: *Pracovní text k přednášce z Optimalizace I, verze z 26.10.2008 (http://www.karlin.mff.cuni.cz/∼lachout/Vyuka/Optima1/081026-Opt-text.pdf)*

[9] Wikipedia: *http://en.wikipedia.org/wiki/Positive-definite_matrix*

[10] Wolfe, P.: The simplex method for quadratic programming. Econometrica 27,3(1959), 382-398.

[11] Wikipedia: *http://en.wikipedia.org/wiki/Frank–Wolfe_algorithm*

[12] M. Frank and P. Wolfe: An algorithm for quadratic programming, Naval Research Logistics Quarterly, 3 (1956), 95-110.

[13] Erwin Kalvelagen: Some Notes On Random Number Generation With GAMS *http://www.amsterdamoptimization.com/pdf/random.pdf*

[14] GAMS Development Corporation:
http://www.gamsworld.org/performance/trace.htm

[15] prof. RNDr. Ladislav Bican, DrSc.: *Lineární algebra a geometrie*, Academia, Praha, 2002

[16] Louisiana State University: Multi Variable Optimization Procedures
http://www.mpri.lsu.edu/textbook/Chapter6-b.htm#generalized

[17] Jitka Dupačová.: Markowitzův model
http://www.karlin.mff.cuni.cz/∼dupacova/downloads/Markowitz.pdf