UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



KVADRATICKÉ MATICOVÉ PROGRAMOVANIE - TEÓRIA A APLIKÁCIE

DIPLOMOVÁ PRÁCA

2019 Bc. Anna TUNOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

KVADRATICKÉ MATICOVÉ PROGRAMOVANIE - TEÓRIA A APLIKÁCIE

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomická a finančná matematika
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	doc. RNDr. Mária Trnovská, PhD.

Bratislava 2019 **Bc. Anna TUNOVÁ**



62878417

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Anna Tunová
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Kvadratické maticové programovanie - teória a aplikácie.
Quadratic matrix programming - theory and applications.

Cieľ: Kvadratické maticové programovanie je trieda vo všeobecnosti nekonvexných optimalizačných úloh s maticovou premennou. Cieľom práce je porovnať rôzne prístupy riešenia takýchto úloh a ilustrovať ich na aplikáciách z oblasti kombinatorickej optimalizácie.

Vedúci: doc. RNDr. Mária Trnovská, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, CSc.

Dátum zadania: 26.01.2017

Dátum schválenia: 27.01.2017

prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie Rada by som sa touto cestou úprimne poďakovala svojej vedúcej diplomovej práce doc. RNDr. Márii Trnovskej PhD. za rady, nápady a pripomienky pri písaní tejto práce. Obzvlášť, hlavne pri finalizácii práce, za veľkú ochotu s konzultovaním a poskytnutie výborných podkladových materiálov.

Abstrakt v štátnom jazyku

TUNOVÁ, Anna: Kvadratické maticové programovanie - teória a aplikácie [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: doc. RNDr. Mária Trnovská, Phd., Bratislava, 2019, 59s.

Kvadratické maticové programovanie je trieda vo všeobecnosti nekonvexných optimalizačných úloh. Cieľom našej práce je porovnať rôzne prístupy riešenia takýchto úloh a ilustrovať ich na aplikáciách z oblasti kombinatorickej optimalizácie. V práci definijeme pojem kvadratické maticové programovanie (QMP), nájdeme jeho semidefinitnú relaxáciu a skúmame možnosti riešenia na Ortogonálnej Procrutovej úlohe. Úlohu riešime dvomi rôznymi postupmi. Jedným z nich je algoritmus SOL-QMP a druhým alternatívna semidefinitná relaxácia. Výsledky oboch prístupov porovnávame. Ku koncu práce riešime Kombinatorickú Procrutovu úlohu, kde sledujeme ako si algoritmy poradia s hľadaním správnej permutačnej matice.

Kľúčové slová: Kvadratické maticové programovanie, Semidefinitná relaxácia, Ortogonálna Procrutova úloha, Nekonvexná optimalizácia

Abstract

TUNOVÁ, Anna: Quadratic matrix programming - theory and applications [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: doc. RNDr. Mária Trnovská, Phd., Bratislava, 2019, 59p.

Quadratic matrix programming is a class of generally non-convex optimization problems. The goal of our thesis is to compare different approaches to solving such problems and to illustrate them on applications in combinatorial optimization problems. We introduce the concept of quadratic matrix programming (QMP), construct its semidefinite relaxation and study the possibilities of solving this kind of problem by application to the Orthogonal Procrustes problem. We solve this problem in two different ways. One of them is the SOL-QMP algorithm and the other one is the alternative semidefinite relaxation. We compare the results of both approaches. At the end of the thesis, we solve the Combinatorial Procrustes problem and study how the algorithms handle the search for the correct permutation matrix.

Keywords: Quadratic matrix programming, Semidefinite relaxation, Orthogonal Procrustes Problem, Non-convex optimization

Obsah

Zoznam použitých symbolov **8**

Úvod **9**

1 Úlohy kvadratického maticového programovania **11**

1.1 Vektorizácia úlohy QMP 12

1.2 Konvexnosť úloh QMP 14

1.3 Motivačné príklady 15

1.3.1 Ortogonálna Procrutova úloha 15

1.3.2 Transformácia Ortogonálnej Procrutovej úlohy 16

1.3.3 Robustné najmenšie štvorce 19

2 Semidefinitná relaxácia úlohy QMP **21**

2.1 Homogenizovaná kvadratická maticová funkcia 21

2.1.1 Vlastnosti homogenizovanej QM funkcie 23

2.2 Konštrukcia SDR 25

2.3 Presnosť Semidefinitnej Relaxácie úlohy QMP 27

2.4 Algoritmus riešenia úlohy QMP 27

3 Riešenie Ortogonálnej Procrutovej úlohy **29**

3.1 Generovanie testovacej úlohy. 29

3.2 Formulácia úlohy ako SDR. 29

3.3 Alternatívna semidefinitná relaxácia 31

3.4 Výsledky a ich porovnania 32

4 Aplikácie z kombinatorickej optimalizácie **36**

4.1 Kombinatorická Procrutova úloha 36

4.2 Riešenie pomocou alternatívnej semidefinitnej relaxácie 37

4.3 Výsledky pre alternatívnu semidefinitnú relaxáciu 38

4.4 Riešenie pomocou algoritmu SOL-QMP 39

4.5 Výsledky pre algoritmus SOL-QMP 41

4.6 Porovnanie ASR a SOL-QMP 42

Záver **44**

Zoznam použitej literatúry **46**

Príloha A **48**

A.1 Kroneckerov súčin **48**

A.1.1 Niektoré vlastnosti Kroneckerovho súčinu 48

A.2 Algoritmus RED **49**

A.3 Niektoré Vety a Tvrdenia využité v práci **50**

Príloha B **52**

Zoznam použitých symbolov

$\mathbb{R}$	- množina všetkých reálnych čísel
$\mathbb{R}^n$	- množina všetkých n-rozmerných reálnych vektorov
$\mathbb{R}^{n\times n}$	- množina všetkých $n \times n$-rozmerných reálnych matíc
$\mathbb{S}^n$	- množina všetkých $n\times n$-rozmerných symetrických matíc, $\mathbb{S}^n = \{A \in \mathbb{R}^{n\times n} : A = A^T\}$
$\mathbb{S}^n_+$	- množina všetkých $n\times n$-rozmerných kladne definitných symetrických matíc, $\mathbb{S}^n = \{A \in \mathbb{R}^{n\times n} : A \succeq 0\}$
$\boldsymbol{I}_r$	- matica identity $r \times r$
$\mathbf{0}_{n\times m}$	- matica núl $n \times m$
$\boldsymbol{E}^r_{ij}$	- matica veľkosti $r\times r$ s 1 na ij-tom mieste a 0 na ostatných miestach
$[\boldsymbol{U}_r]$	- pre švorcovú maticu $\boldsymbol{U}$ označuje $r \times r$ podmaticu umiestnenú v pravom dolnom rohu matice $\boldsymbol{U}$
$s_i(X), r_j(X)$	- $i-ty$ stĺpec, $j-ty$ riadok matice X
δ_{ij}	- Kroneckerova delta, tj. $\delta_{ii} = 1$ a $\delta_{ij} = 0$ pre $i \neq j$
$\|\boldsymbol{A}\|_F$	- Frobeniova norma matice $\boldsymbol{A}$, $\|\boldsymbol{A}\|_F := \sqrt{Tr(A^TA)}$
$\|z\|$	- Euklidovská norma vektora z, $\|z\| := \sqrt{z^Tz}$

Zoznam operácií a relácií

$M \succ 0$	- matica M je kladne definitná
$M \succeq 0$	- matica M je kladne semidefinitná
$M_1 \succeq M_2$	- $M_1 - M_2 \succeq 0$
$vec(M)$	- vektorizácia matice M
$M_1 \otimes M_2$	- Kroneckerov súčin matíc M_1, M_2
$val(P)$	- optimálna hodnota účelovej funkcie optimalizačnej úlohy: (P) : $\min\{f(x) : x \in C\}$

Zoznam skratiek

SDP	- semidefinitné programovanie
SDR	- semidefinitná relaxácia
QCQP	- kvadratické programovanie s kvadratickými obmedzeniami
QMP	- kvadratické maticové programovanie

Úvod

Kvadratické maticové programovanie (QMP z anglického *quadratic matrix programming*) sú vo všeobecnosti nekonvexné optimalizačné úlohy s maticovou premennou. Tieto úlohy sú špeciálnym prípadom širšej triedy optimalizačných úloh - úloh kvadratického programovania s kvadratickými obmedzeniami (QCQP z anglického *quadratically constrained quadratic programming*).

Vzťah medzi úlohami QMP a QCQP však môžeme chápať aj opačne. Hoci úlohu QMP vieme vždy transformovať na úlohu QCQP, čo ukážeme aj v tejto práci, vektor sa dá prirodzene chápať ako matica veľkosti $n \times 1$. To podporuje druhý pohľad, teda, že trieda úloh QMP je všeobecnejšia a úlohy QCQP sú jej špeciálnym prípadom.

V našej práci sa budeme venovať úlohám QMP, ktorým zatiaľ nebolo v existujúcej literatúre vyčleneného veľa priestoru. Môže vznikať otázka, či má vôbec význam zaoberať sa úlohami QMP samostatne, keďže úlohy QCQP sú známe a spracované (príkladom toho môže byť [4]). Štruktúra úlohy QMP však môže byť v mnohých prípadoch výhodná z hľadiska efektivity výpočtu, relaxácie semidefinitného programovania či duality.

Naším cieľom v diplomovej práci je porovnať rôzne prístupy riešenia úloh QMP a následne ich ilustrovať na aplikáciách z oblasti kombinatorickej optimalizácie. V teoretickej časti budeme vychádzať hlavne z [1], kde autor predstavuje úlohy QMP a možnosti ich riešenia. Jeho postupy však budeme rozširovať a dopĺňať, tiež pridávať ďalší materiál tak, aby práca nebola určená len odborníkom na konvexnú či nekonvexnú optimalizáciu.

V prvej časti diplomovej práce sa budeme venovať kvadratickému maticovému programovaniu z teoretického hľadiska. Sú to prvá a druhá kapitola. Zadefinujeme úlohu QMP, jej vlastnosti, transformujeme úlohu QMP na jej vektorizovaný ekvivalent. Tiež ukážeme dva príklady úloh tohto typu a možnosti ich riešenia. Jednou z možností riešenia, ktorej sa budeme venovať je semidefinitná relaxácia (SDR). Tento prístup k riešeniu je aproximačný, ale ukážeme, že dáva veľmi dobré výsledky.

V druhej časti diplomovej práce sa pozrieme na aplikácie úloh kvadratického maticového programovania. Medzi úlohy, kde možeme využiť QMP patrí napríklad Ortogonálna Procrutova úloha. Ortogonálna Procrutova úloha je príklad maticovej optimalizačnej úlohy. V nej sa snažíme zmeniť danú množinu dát tak, aby zmenené dáta

vyjadrili inú množinu dát čo možno najlepšie. Táto úloha sa dá prepísať a následne riešiť ako úloha kvadratického maticového programovania.
Ortogonálnu Procrutovu úlohu budeme riešiť pre dva typy ortogonálnych matíc. Jedným z nich budú rotačné a druhým permutačné matice. Na oba typy úloh použijeme rôzne prístupy riešenia. Konkrétne algoritmus SOL-QMP, ktorý tiež budeme v práci odvádzať a pekný, jednoduchší postup pomocou alternatívnej semidefinitnej relaxácie. Získané výsledky spracujeme a porovnáme.
V prílohe práce uvádzame niektoré poznatky, ktoré sú potrebné pri odvádzaní matematických vzťahov v teoretickej ale aj praktickej časti práce.

1 Úlohy kvadratického maticového programovania

V prvej kapitole našej práce vychádzame hlavne z [1].
Na začiatok pripomíname, že kvadratická funkcia $g : R^n \to R$ je funkcia tvaru:

$$g(x) = x^T Ax + 2b^T x + c \tag{1}$$

kde $A \in S^n$, $b \in R^n$ a $c \in R$. V tejto práci budeme používať názov "kvadratická vektorová funkcia"namiesto názvu "kvadratická funkcia", aby sme ju vedeli dostatočne odlíšiť od termínu "kvadratická maticová funkcia".

Definícia 1.1 (Kvadratická maticová funkcia)**.** [1, str 3] Kvadratická maticová funkcia (QMF) je funkcia $f : R^{n\times r} \to R$ tvaru:

$$f(X) = Tr(X^T AX) + 2Tr(B^T X) + c \tag{2}$$

s premennou $X \in R^{n\times r}$, kde $A \in S^n$, $B \in R^{n\times r}$ a $c \in R$.
Ak $B = 0_{n\times r}$ a $c = 0$, potom sa f nazýva homogénna kvadratická maticová funkcia alebo aj kvadratická maticová forma.

Definícia 1.2 (Rád QMF)**.** Rád kvadratickej maticovej funkcie f je definovaný ako rozmer r premennej $X \in R^{n\times r}$ vstupujúcej do funkcie f.

Všimnime si, že každá kvadratická vektorová funkcia, je kvadratická maticová funkcia rádu jedna. Neskôr v práci tiež ukážeme, že každá kvadratická maticová funkcia sa dá naformulovať v tvare kvadratickej vektorovej funkcie. Pri tomto prevode využijeme funkciu vektorizácie matice.

Definícia 1.3 (Vektorizácia matice)**.** Vektorizácia matice je lineárna transformácia $vec : R^{n\times r} \to R^{nr}$ tvaru:

$$vec : X \to vec(X) = [x_{1,1}, ..., x_{n,1}, x_{1,2}, ..., x_{n,2}, ..., x_{1,r}, ..., x_{n,r}]^T.$$

Príklad 1.4. Majme 2×2 maticu A, $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \to vec(A) = \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}$.

V úlohách kvadratického maticového programovania sú účelová funkcia aj funkcie ohraničení kvadratické maticové funkcie. Úloha, akou sa v našej práci budeme zaoberať, je teda popísaná v nasledujúcej definícii.

Definícia 1.5 (1,str 1)**.** Úlohu kvadratického maticového programovania formulujeme nasledovne:

$$\begin{array}{lll} \min & Tr(X^T A_0 X) + 2Tr(B_0^T X) + c_0 & \\ & Tr(X^T A_i X) + 2Tr(B_i^T X) + c_i \quad \leq \quad \alpha_i, & i \in I \\ & Tr(X^T A_j X) + 2Tr(B_j^T X) + c_j \quad = \quad \alpha_j, & j \in E \\ & X \in R^{n \times r} & \end{array} \tag{3}$$

kde $A_i = A_i^T$, $B_i \in R^{n \times r}$, α_i, $c_i \in R$, $i \in \{0\} \cup I \cup E$.

1.1 Vektorizácia úlohy QMP

Jedným z našich cieľov v diplomovej práci je študovať úlohy kvadratického maticového programovania, v ktorých sa hľadá minimum kvadratickej maticovej účelovej funkcie v závislosti od ohraničení na rovnosti a nerovnosti. V nasledujúcej časti si ukážeme, ako sa funkcia (2) dá previesť do tvaru úloh QCQP.

Definícia 1.6 (Vektorizácia QMF)**.** [1, str 3] Vektorizácia kvadratickej maticovej funkcie f z (2) sa nazýva funkcia $f^V : R^{nr} \to R$ tvaru:

$$f^V(z) = z^T(I_r \otimes A)z + 2vec(B)^T z + c, \tag{4}$$

kde znak $\otimes$ označuje Kronekerov súčin, $(I_r \otimes A) \in R^{rn \times rn}$. Matica $(I_r \otimes A)$ je blokovo diagonálna s r blokmi matice $A \in R^{n \times n}$ a $z = vec(X)$.

Poznámka: Viac informácií o Kroneckerovom súčine sa nachádza v dodatku práce.

Kvadratickú maticovú funkciu f z (2) vieme po úpravách napísať nasledovne:

$$f(X) = f^V(vec(X)), \tag{5}$$

pričom po dosadení za z vo vzťahu (4) má funkcia (5) tvar:

$$f^V(vec(X)) = vec(X)^T(I_r \otimes A)vec(X) + 2vec(B)^T vec(X) + c. \tag{6}$$

Pre lepšiu názornosť uvádzame funkciu (6) rozpísanú aj pomocou matíc a vektorov, vtedy je funkcia v tvare:

$$f^V(vec(X)) = \Big(x_{1,1},\ldots,x_{n,1},\ldots,x_{n,r}\Big)_{1\times nr} \begin{pmatrix} A & 0 & 0 & & 0 \\ 0 & A & 0 & & 0 \\ 0 & 0 & A & & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & & A \end{pmatrix}_{nr\times nr} \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{n,1} \\ \vdots \\ x_{n,r} \end{pmatrix}_{nr\times 1}$$

$$+2\Big(b_{1,1},\ldots,b_{n,1},\ldots,b_{n,r}\Big)_{1\times nr} \begin{pmatrix} x_{1,1} \\ \vdots \\ x_{n,1} \\ \vdots \\ x_{n,r} \end{pmatrix}_{nr\times 1} + c$$

Vektorizáciu kvadratickej maticovej funkcie sme našli, chceli by sme ešte previesť celú úlohu QMP, teda úlohu (3), do tvaru úlohy QCQP. Využijeme k tomu už nájdený vzťah (5) medzi funkciou a jej vektorizáciou.

Kvôli skrátenému zápisu úlohu (3) napíšeme v tvare:

$$\begin{array}{lll} (\text{QMP}) \quad \min & f_0(X) & \\ & f_i(X) \quad \leq \quad \alpha_i, & i \in I \\ & f_j(X) \quad = \quad \alpha_j, & j \in E \\ & X \in R^{n\times r} & \end{array} \tag{7}$$

kde $f_i : R^n \to R$, $i \in \{0\} \cup I \cup E$ sú kvadratické maticové funkcie rádu r dané

$$f_i(X) = Tr(X^T A_i X) + 2Tr(B_i^T X) + c_i, X \in R^{n\times r},$$

kde $A_i = S^n$, $B_i \in R^{n\times r}$, $c_i \in R$, $i \in \{0\} \cup I \cup E$.

Definícia 1.7 (Homogénna úloha QMP)**.** [1, str 4] V prípade, že v úlohe (7) sú všetky funkcie f_i, $i \in \{0\} \cup I \cup E$ homogénne kvadratické maticové funkcie rádu r, teda $B = 0_{n\times r}$ a $c = 0$, úloha (7) sa nazýva homogénna úloha QMP (rádu r).

Pomocou vzťahu (5) môžeme prepísať úlohu QMP (7) ako úlohu QCQP:

$$\begin{array}{lll} \min & f_0^V(z) & \\ & f_i^V(z) \quad \leq \quad \alpha_i, & i \in I \\ & f_j^V(z) \quad = \quad \alpha_j, & j \in E \\ & z \in R^{n \times r} & \end{array} \tag{8}$$

kde $z := vec(X)$.

Úlohu (8) budeme potom nazývať *vektorizovaná úloha QMP*.

1.2 Konvexnosť úloh QMP

Zaujíma nás, či a za a akých podmienok môže byť funkcia (2) konvexná. Toto zisťovanie nám uľahčí nasledujúca lema.

Lema 1.8. Nech $A \in S^m$, $B \in S^n$. Potom pre všetky A, B platí

$$\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \succeq 0 \Longleftrightarrow A \succeq 0, B \succeq 0.$$

Dôkaz.

$\Rightarrow$ Ak je blokovo diagonálna matica s maticami A, B na diagonále kladne semidefinitná, potom matice A aj B sú kladne semidefinitné.

$$\begin{pmatrix} x_1, & x_2 \end{pmatrix}^T \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq 0, \forall x \in R^{m+n}$$

$$x_1^T A x_1 + x_2^T B x_2 \geq 0, \forall x_1 \in R^m, \forall x_2 \in R^n$$

Po vynásobení matice vektorom $(x_1, x_2)^T$ sme dostali súčet dvoch kvadratických foriem. Tento súčet je ≥ 0, $\forall x_1, x_2$. Kedže nerovnosť platí pre $\forall x_1, \forall x_2$, nech $x_1 = 0$. Po dosadení dostávame nerovnosť

$$x_2^T B x_2 \geq 0, \forall x_2 \in R^n. \tag{9}$$

Vďaka nerovnosti (9) vieme, že B je kladne semidefinitná matica.

Analogicky po dosadení nulového vektora za x_2, dostávame kladnú semidefinitnosť matice A. Teda matice A, B sú kladne semidefinitné.

$\Leftarrow$ Ak matice A, B sú kladne semidefinitné, potom je kladne semidefinitná aj blokovo diagonálna matica s maticami A, B na diagonále.

$$x_1^T A x_1 \geq 0, \forall x_1 \in R^m$$

$$x_2^T B x_2 \geq 0, \forall x_2 \in R^n$$

Po sčítaní týchto dvoch výrazov, dostávame

$$\begin{pmatrix} x_1, & x_2 \end{pmatrix}^T \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq 0, \forall x \in R^{m+n}.$$

Teda blokovo diagonálna matica s maticami A, B na diagonále je kladne semidefinitná. □

Vráťme sa teraz späť ku kvadratickej maticovej funkcii (2). Zaujíma nás, kedy je funkcia konvexná.

- Platí, že dvakrát diferencovateľná funkcia $f : K \subset R^n \to R$ definovaná na konvexnej množine K je konvexná práve vtedy, keď je jej Hessova matica kladne semidefinitná, tj. $\forall x : \nabla^2 f(x) \succeq 0$.

- Po rozpísaní funkcie (2) do tvaru funkcie (6) vidíme, že nás zaujíma kladná semidefinitnosť blokovo diagonálnej matice $I_r \otimes A$. Tá je podľa Lemy (1.8) kladne semidefinitná práve vtedy, keď sú jej diagonálne bloky kladne semidefinitné.

Teda môžeme povedať, že funkcia f z (2) je konvexná práve vtedy, keď $A \succeq 0$ a rýdzo konvexná práve vtedy, keď $A \succ 0$.

1.3 Motivačné príklady

Úlohy kvadratického maticového programovania sa vyskytujú v niekoľkých oblastiach aplikácií. V nasledujúcej časti sa pozrieme na také dva príklady. V oboch príkladoch sme pôvodný problém preformulovali do tvaru úlohy QMP, ktorú budeme vedieť riešiť.

1.3.1 Ortogonálna Procrutova úloha

Procrutove metódy sa používajú na transformovanie jednej množiny dát, aby reprezentovali inú množinu dát tak presne, ako je to len možné.

Názov úloh pochádza z Gréckej mytológie, kde, ako hovorí [7] v Theseovom príbehu, zlodej Procrutes pozýval pocestných na jedlo a tiež odpočinok v čarovnej posteli, ktorá vyhovuje potrebám každého návštevníka. Keď návštevník vošiel, Procrutes mu však končatiny buď natiahol, alebo mu z nich odrezal tak, aby návštevník presne zapasoval do jeho postele. Jeho vyčínanie zakončil až Theseus, ktorému sa podarilo vymeniť si s Procrutom úlohy a nechať ho nakoniec zakúsiť jeho vlastné techniky.
Ortogonálna Procrutova úloha je príklad maticovej optimalizačnej úlohy. V úlohe máme dané matice A a B a hľadáme štvorcovú ortogonálnu maticu X, ktorá „približuje“ A k B. Teda hľadáme X tak, aby suma štvorcov rezíduí bola minimálna. Úloha má tvar:

$$\begin{aligned} \min \quad & \|AX - B\|_F^2 \\ & X^T X = I_r \end{aligned} \tag{10}$$

s premennou $X \in R^{r\times r}$, kde $A \in R^{n\times r}$, $B \in R^{n\times r}$.
Autori v [1] pracujú s úlohou (10) v tvare úlohy QMP, kedže v tomto tvare je úloha riešiteľná metódami kvadratického maticového programovania:

$$\begin{aligned} \min \quad & Tr(X^T A^T A X) - 2Tr(B^T A X) + \|B\|_F^2 \\ & Tr(X^T (E_{ij}^r + E_{ji}^r) X) \;=\; 2\delta_{ij}, \qquad 1 \le i, j \le r \\ & X \in R^{r\times r} \end{aligned} \tag{11}$$

Matica E_{ij}^r je veľkosti $r \times r$ s 1 na ij-tom mieste a 0 na ostatných miestach. Znak δ_{ij} označuje Kroneckerovu deltu a jej hodnoty sú $\delta_{ii} = 1$ a $\delta_{ij} = 0$ pre $i \neq j$.
V nasledujúcej časti ukážeme postup, ktorý umožňuje prejsť od úlohy (10) k úlohe (11). Úlohu v našej práci neriešime, ako však bolo ukázané v [5], pre štvorcovú maticu X je Ortogonálna Procrutova úloha efektívne riešiteľná.

1.3.2 Transformácia Ortogonálnej Procrutovej úlohy

Úlohu (10) by sme chceli preformulovať do tvaru úlohy QMP (11). Potrebujeme teda prepísať účelovú funkciu a funkciu ohraničenia ako kvadratické maticové funkcie. Začneme účelovou funkciou. To nie je náročné, stačí ju rozpísať pomocou Frobeniovej normy a vhodne roznásobiť:

$$\|AX - B\|_F^2 = Tr((AX - B)^T (AX - B)) = Tr((X^T A^T - B^T)(AX - B)) =$$

$$= Tr(X^T A^T A X) - 2Tr(B^T A X) + Tr(B^T B).$$

Ako druhú preformulujeme funkciu ohraničení, s ktorou je to náročnejšie. Z formulácie funkcie (2) vidíme, že matica $(X^T(E^r_{ij} + E^r_{ji})X) = (X^T E^r_{ij} X + X^T E^r_{ji} X)$ musí byť symetrická, pokiaľ chceme, aby úloha (10) bola úlohou QMP. Táto podmienka však splnená je, keďže

$$(X^T E^r_{ij} X)^T = X^T E^r_{ji} X,$$

a teda sčítavame dve navzájom symetrické matice. Navyše v nám v ďalších výpočtoch stačí uvažovať len jednu z týchto matíc, napríklad $X^T E^r_{ij} X$, a pre symetrické matice uvažovať analogicky. Pri výpočtoch nám pomôže, že funkcia ohraničení z pôvodnej úlohy (10) má ekvivalentný tvar v podobe rovnice:

$$X^T X = I_r \Longleftrightarrow \left[X^T X\right]_{ij} = \delta_{ij}, \tag{12}$$

kde δ_{ij} označuje Kroneckerovu deltu. Jej hodnoty sú $\delta_{ii} = 1$ a $\delta_{ij} = 0$ pre $i \neq j$. Našou úlohou je ukázať, že ohraničenie (12) je ekvivalentné s rovnicou

$$Tr(X^T(E^r_{ij} + E^r_{ji})X) = 2\delta_{ij}, 1 \leq i, j \leq r, \tag{13}$$

ktorá určuje ohraničenie v preformulovanej úlohe (11). Tú vieme prepísať aj ako

$$Tr(X^T E^r_{ij} X + X^T E^r_{ji} X) = 2\delta_{ij}, 1 \leq i, j \leq r. \tag{14}$$

Ako sme už spomínali, matice z rovnice (14) sú symetrické, tak bez ujmy na všeobecnosti budeme zatiaľ pracovať len s maticou $X^T E^r_{ij} X$. Predpokladajme na začiatok $i = 1, j = 1$. Potom má matica $X^T E^r_{ij} X$ tvar:

$$X^T E^r_{ij} X = \begin{pmatrix} x_{1,1} & x_{2,1} & ... & x_{n,1} \\ x_{1,2} & x_{2,2} & ... & x_{n,2} \\ \vdots & \vdots & & \vdots \\ x_{1,n} & x_{2,n} & ... & x_{n,n} \end{pmatrix} \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 0 & ... & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & ... & 0 \end{pmatrix} \begin{pmatrix} x_{1,1} & x_{1,2} & ... & x_{1,n} \\ x_{2,1} & x_{2,2} & ... & x_{2,n} \\ \vdots & \vdots & & \vdots \\ x_{n,1} & x_{n,2} & ... & x_{n,n} \end{pmatrix}$$

Po roznásobení matíc X^T a E^r_{ij} dostávame maticu $X^T E^r_{ij}$, ktorá vo svojom prvom stĺpci obsahuje prvý stĺpec matice X^T. To zodpovedá indexom i, j. Index i určuje, ktorý stĺpec s_i z matice X^T vyberáme, index j, v ktorom stĺpci novej matice sa $s_i(X^T)$ bude nachádzať.

$$X^T E^r_{ij} X = \begin{pmatrix} x_{1,1} & 0 & ... & 0 \\ x_{1,2} & 0 & ... & 0 \\ \vdots & \vdots & & \vdots \\ x_{1,n} & 0 & ... & 0 \end{pmatrix} \begin{pmatrix} x_{1,1} & x_{1,2} & ... & x_{1,n} \\ x_{2,1} & x_{2,2} & ... & x_{2,n} \\ \vdots & \vdots & & \vdots \\ x_{n,1} & x_{n,2} & ... & x_{n,n} \end{pmatrix}$$

Keď znásobíme aj matice $X^T E_{ij}^r$ a X, vidíme, že matica $X^T E_{ij}^r X$ má tvar závislý od indexov i, j. Ako si môžeme všimnúť v (15), v tomto tvare každá zložka stĺpca $s_i(X)^T$ postupne prenásobí riadok $r_j(X)$:

$$X^T E_{ij}^r X = \begin{pmatrix} x_{1,1}^2 & x_{1,1}x_{1,2} & ... & x_{1,1}x_{1,n} \\ x_{1,2}x_{1,1} & x_{1,2}^2 & ... & x_{1,2}x_{1,n} \\ \vdots & \vdots & & \vdots \\ x_{1,n}x_{1,1} & x_{1,n}x_{1,2} & ... & x_{1,n}^2 \end{pmatrix} = \begin{pmatrix} x_{1,1}r_1(X) \\ x_{1,2}r_1(X) \\ \vdots \\ x_{1,n}r_1(X) \end{pmatrix} \tag{15}$$

Odvodili sme ako vyzerá tvar matice $X^T E_{ij}^r X$ pre indexy $i = 1$, $j = 1$. Keďže sa snažíme analyzovať rovnicu (14) v našom ďalšom kroku potrebujeme spočítať stopu matice (15). Tú vieme zapísať nasledovne ako skalárny súčin vektorov:

$$Tr(X^T E_{1,1}^r X) = x_{1,1}^2 + x_{1,2}^2 + ... + x_{1,n}^2 = \big\langle s_1(X^T), r_1(X) \big\rangle.$$

Všeobecne pre indexy i, j má stopa matice $X^T E_{ij}^r X$ tvar

$$Tr(X^T E_{ij}^r X) = \big\langle s_i(X^T), r_j(X) \big\rangle = \big\langle r_i(X), s_j(X^T) \big\rangle. \tag{16}$$

Po pripočítaní stopy matice $X^T E_{ji}^r X$, ktorá je vďaka symetrii matíc rovnaká ako pre maticu $X^T E_{ij}^r X$ dostávame rovnicu

$$Tr(X^T E_{ij}^r X + X^T E_{ji}^r X) = 2\big\langle r_i(X), s_j(X^T) \big\rangle. \tag{17}$$

Posledné, čo potrebujeme ukázať je, že pre všetky $1 \leq i, j \leq r$ platí:

$$2\big\langle r_i(X), s_j(X^T) \big\rangle = 2\delta_{ij}. \tag{18}$$

Využijeme pri tom pôvodné ohraničenie $X^T X = I_r$. Zo zadania úlohy (10) vieme, že matica X je štvorcová a teda aj $XX^T = I_r$. Z toho môžeme môžeme určiť nasledujúce rovnosti:

$$\langle r_i(X), s_j(X^T) \rangle = \begin{cases} 1, & i = j \\ 0, & i \neq j. \end{cases} \tag{19}$$

Kedže rovnica (19) je totožná s definíciou Kronecrovej delty δ_{ij}, podarilo sa nám ukázať, že ohraničenia (12) a (13) sú ekvivalentné, rovnako ako aj úlohy (10) a (11).

1.3.3 Robustné najmenšie štvorce

Majme lineárny systém $Ax = b$, kde $A \in R^{n\times r}$, $b \in R^r$ a $x \in R^n$. Vieme, že približné riešenie takého systému môžeme riešiť ako úlohu najmenších štvorcov:

$$\min_x \|Ax - b\|_2^2.$$

Úloha, ktorej sa v tejto časti budeme venovať je však úlohou robustných najmenších štvorcov. Vychádzame hlavne z [1], kde autori ukazujú možnosť transformácie tejto úlohy na úlohu QMP. My ich postup podrobnejšie dopĺňame, prípadne vysvetľujeme. Špecifické pre úlohu robustných najmenších štvorcov je, že dáta vstupujúce do úlohy nie sú známe, ale je v nich odchýlka z danej množiny neurčitosti. V našom prípade bude odchýlka reprezentovaná maticou $\triangle$, ktorá patrí do množiny neurčitosti U. Chceli by sme riešiť úlohu najmenších štvorcov pre "najhorší" možný prípad voľby dát z danej množiny neurčitosti.

Predpokladajme teda, že matica A nie je fixná, ale je určená triedou matíc $A+\triangle^T$. A je v tomto prípade známa matica a $\triangle \in R^{n\times r}$ je neznáma matica odchýlky, o ktorej vieme, že je z kompaktnej množiny neurčitosti U. Metóda robustných najmenších štvorcov hľadá vektor $x \in R^n$, ktorý minimalizuje najhorší prípad chyby s ohľadom na všetky možné hodnoty $\triangle \in U$:

$$\min_x \max_{\triangle\in U} \|b - (A + \triangle^T)x\|_2^2 \tag{20}$$

Výraz z (20) vieme rozpísať pomocou Euklidovskej vektorovej normy $\|z\|_2 = \sqrt{z^T z}$ ako:

$$\|b - (A + \triangle^T)x\|_2^2 = (b - (A + \triangle^T)x)^T(b - (A + \triangle^T)x).$$

Kedže pre vektor z je $z^T z$ skalárna hodnota, ktorú si vieme predstaviť ako maticu 1×1, tak platí

$$z^T z = Tr(z^T z) = Tr(zz^T). \tag{21}$$

Pomocou tretej rovnosti z (21) a jednoduchých algebraických úprav dostávame (20) v tvare (22):

$$\|b - (A + \triangle^T)x\|_2^2 = Tr(b - (A + \triangle^T)x)(b - (A + \triangle^T)x)^T =$$

$$= Tr(b - Ax - \triangle^T x)(b^T - x^T A^T - x^T \triangle) =$$

$$= Tr(bb^T - Axb^T - \triangle^T xb^T - bx^T A^T + Axx^T A^T + \triangle^T xx^T A^T - bx^T \triangle + Axx^T \triangle + \triangle^T xx^T \triangle) =$$

$$= Tr(\triangle^T xx^T \triangle) + 2Tr((Ax - b)x^T \triangle) + Tr((b - Ax)(b - Ax)^T) \tag{22}$$

rovnako ako autori v [1] spravíme niekoľko substitúcii:

$$Q = xx^T \in S^n, F = x(Ax - b)^T \in R^{n \times r}, c = \|b - Ax\|^2 \in R \tag{23}$$

a maximalizačná úloha z (20) nadobúda tvar:

$$\max\{Tr(\triangle^T Q \triangle) + 2Tr(F^T \triangle) + c, \triangle \in U\}. \tag{24}$$

Množinu U z (24) môžeme zvoliť napríklad ako:

$$U = \{\triangle \in R^{n \times r} : Tr(\triangle^T \triangle) \leq \rho\} \tag{25}$$

$$= \{\triangle : \|\triangle\|_F^2 \leq \rho\}.$$

Maximalizačná úloha (24) s množinou U podľa (25) je úloha QMP rádu r. Táto úloha je špeciálnym prípadom úlohy (3), v ktorej je nerovnostné ohraničenie len jednoduché a dátová matica v ňom identita.

Poznámka 1.9. Riešením úlohy robustných najmenších štvorcov sa zaoberá napríklad článok [6].

2 Semidefinitná relaxácia úlohy QMP

Začneme s konštrukciou semidefinitnej relaxácie (SDR) pre úlohy kvadratického maticového programovania. Vo všeobecnosti je SDR aproximačná technika, ktorá je výpočtovo efektívna pre viaceré náročné optimaliznačné úlohy. Dá sa aplikovať na veľa nekonvexných úloh, podľa [2] skoro mechanicky na úlohy QCQP. Teda vhodný postup konštrukcie SDR by mohol vychádzať z vektorizovanej úlohy QMP (8):

$$\begin{array}{lll} \min & f_0^V(z) & \\ & f_i^V(z) \quad \leq \quad \alpha_i, & i \in I \\ & f_j^V(z) \quad = \quad \alpha_j, & j \in E \\ & z \in R^{n \times r} & \end{array}$$

ktorá pripomíname, že je úlohou QCQP. Autori článku [1] však bežnú schému konštrukcie SDR prispôsobili a konštuujú novú schému špeciálne pre QMP úlohy, tak, aby dostali SDR pre úlohy QMP.

V druhej kapitole našej práce, pri konštrukcii semidefinitnej relaxácie úlohy QMP, nasledujeme postup autorov z článku [1]. Tento postup dopĺňame a vysvetľujeme. Semidefinitnej relaxácii úloh QCQP sa venujú aj autori v článku [2], ktorý pomohol k pochopeniu tejto témy.

2.1 Homogenizovaná kvadratická maticová funkcia

Pri odvádzaní úlohy QMP (7), v prvej kapitole našej práce, sme začali kvadratickou vektorovou funkciou g (1). Rovnako sa teraz pozrime na jej homogenizovanú verziu.

Definícia 2.1. [1, str 6] Homogenizovaná kvadratická vektorová funkcia $g^H : R^{n+1} \to R$, ktorá je odvodená od funkcie g (1), je daná ako kvadratická forma:

$$g^H(x;t) = x^t Ax + 2b^T xt + ct^2, \tag{26}$$

ktorá má rozpísaná tvar:

$$g^H(x;t) = \begin{pmatrix} x \\ t \end{pmatrix}^T \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix}.$$

Maticu, ktorá sa viaže k funkcii g^H označíme ako:

$$M(g^H) = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \tag{27}$$

V našej práci by sme sa však chceli venovať kvadratickým maticovým (QM) funkciám rádu r. Autori článku [1] zovšeobecnili predchádzajúcu definíciu z "vektorovej"funkcie rádu jedna na maticové funkcie rádu r.

Definícia 2.2. [1, str 7] Máme dané matice $A \in S^n$, $B \in R^{n\times r}$ a $c \in R$. Homogenizovaná kvadratická maticová funkcia $f^H : R^{(n+r)\times r} \to R$, ktorá je odvodená od funkcie f (2), je daná ako:

$$f^H(X;Z) = Tr(X^T AX) + 2Tr(Z^T B^T X) + \frac{c}{r} Tr(Z^T Z) \tag{28}$$

pre všetky $X \in R^{n\times r}, Z \in R^{r\times r}$. Táto funkcia je rádu r, môžeme ju rozpísať ako:

$$f^H(X;Z) = Tr\left(\begin{pmatrix} X \\ Z \end{pmatrix}^T \begin{pmatrix} A & B \\ B^T & \frac{c}{r} I_r \end{pmatrix} \begin{pmatrix} X \\ Z \end{pmatrix} \right)$$

a teda k nej prislúcha matica:

$$M(f^H) = \begin{pmatrix} A & B \\ B^T & \frac{c}{r} I_r \end{pmatrix} \tag{29}$$

Všimnime si, že pokiaľ je $r = 1$, teda premenná X vstupujúca do funkcie f^H nie je matica, ale vektor, sú definície (28) a (29) zhodné s definíciami (26) a (27).

Definícia 2.3. [1, str 7] Homogenizovaná úloha QMP je úloha tvaru:

$$\begin{array}{lll} \min & Tr(X^T A_0 X) + 2Tr(Z^T B_0^T X) + \frac{c_0}{r} Tr(Z^T Z) & \\ & Tr(X^T A_i X) + 2Tr(Z^T B_i^T X) + \frac{c_i}{r} Tr(Z^T Z) & \leq \alpha_i, \quad i \in I, \\ & Tr(X^T A_j X) + 2Tr(Z^T B_j^T X) + \frac{c_j}{r} Tr(Z^T Z) & = \alpha_j, \quad j \in E, \\ & Tr(Z^T (E_{ij}^r + E_{ji}^r) Z) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r, \end{array} \tag{30}$$

kde $Y \in \mathbb{R}^{n\times r}$, $Z \in \mathbb{R}^{r\times r}$ a δ_{ij} je Kroneckerova delta. Túto úlohu môžeme v skrátenej formulácii zapísať nasledovne:

$$\begin{array}{lll} \min & f_0^H(Y;Z) & \\ & f_i^H(Y;Z) & \leq \alpha_i, \quad i \in I, \\ & f_j^H(Y;Z) & = \alpha_j, \quad j \in E, \\ & \psi_{ij}(Y;Z) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r, \end{array} \tag{31}$$

kde $\psi_{ij}(Y;Z) = Tr(Z^T(E_{ij}^r + E_{ji}^r)Z)$, $Y \in \mathbb{R}^{n\times r}$, $Z \in \mathbb{R}^{r\times r}$.

2.1.1 Vlastnosti homogenizovanej QM funkcie

V tejto časti uvedieme tri lemy, ktoré sa týkajú homogenizovanej QM funkcie f^H a jej vlastností. Poznatky z prvých dvoch sú potrebné pri dôkaze tretej lemy. Tretia lema sa využije k odvodeniu semidefinitnej relaxácie úlohy QMP v druhej časti kapitoly.
Prvé dve lemy sú v [1] uvedené ako tvrdenia bez dôkazu, ten k nim dopĺňame v našej práci. Dôkaz tretej lemy uvádzajú v stručnejšej forme aj autori článku [1], my sme ho doplnili o medzikroky.

Lema 2.4. Pre všetky $X \in \mathbb{R}^{n\times r}$ platí:

$$f^H(X; I_r) = f(X). \tag{32}$$

Dôkaz.
Dôkaz je jednoduchý, stačí rozpísať funkciu $f^H(X; I_r)$ podľa (28) a využiť vlastnosti matice I_r:

$$f^H(X; I_r) = Tr(X^T AX) + 2Tr(I_r^T B^T X) + \frac{c}{r} Tr(I_r^T I_r) =$$
$$= Tr(X^T AX) + 2Tr(B^T X) + c = f(X).$$

□

Lema 2.5. Pre všetky $X \in \mathbb{R}^{n\times r}, Z \in \mathbb{R}^{r\times r}$, také, že $Z^T Z = I_r$ platí:

$$f^H(X; Z) = f(YZ^T). \tag{33}$$

Dôkaz.
Začnime rozpísaním funkcie $f^H(X; Z)$ podľa (28) a využitím $Z^T Z = I_r$:

$$f^H(X; Z) = Tr(X^T AX) + 2Tr(Z^T B^T X) + \frac{c}{r} Tr(Z^T Z) =$$
$$= Tr(X^T AXZ^T Z) + 2Tr(Z^T B^T X) + \frac{c}{r} Tr(I_r) =$$

Využívame komutatívnu vlastnosť stopy, pri ktorej môžeme cyklicky meniť poradie násobených matíc, pričom ich stopa zostane rovnaká:

$$= Tr(ZX^T AXZ^T) + 2Tr(B^T XZ^T) + \frac{c}{r} r =$$

Po transponovaní dostávame (nehomogénnu) QM funkciu (2):

$$Tr((XZ^T)^T A(XZ^T)) + 2Tr(B^T(XZ^T)) + c = f(YZ^T).$$

□

S využitím predchádzajúcich dvoch vzťahov môžeme skonštruovať nasledujúcu lemu. Tá hovorí o tom, že homogenizovaná úloha QMP (31) rádu r je ekvivalentná (nehomogenizovanej) úlohe QMP (7) z prvej kapitoly práce:

$$\begin{array}{lllll} \text{(QMP)} & \min & f_0(X) & & \\ & & f_i(X) & \leq & \alpha_i, \quad i \in I \\ & & f_j(X) & = & \alpha_j, \quad j \in E \\ & & X \in R^{n\times r} & & \end{array}$$

kde $f_i : R^n \to R$, $i \in \{0\} \cup I \cup E$ sú kvadratické maticové funkcie rádu r dané

$$f_i(X) = Tr(X^T A_i X) + 2Tr(B_i^T X) + c_i, X \in R^{n\times r},$$

$A_i = S^n$, $B_i \in R^{n\times r}$, $c_i \in R$.

Lema 2.6. [1, str 7] Uvažujme homogenizovanú úlohu QMP (31) a (nehomogenizovanú) úlohu QMP (7):

1. Nech je úloha QMP (7) riešiteľná a nech je X^* jej optimálne riešenie. Potom je úloha (31) riešiteľná, jej optimálne riešenie je $(X^*; I_r)$ a $val(31) = val(7)$[1].

2. Nech je úloha (31) riešiteľná a nech je $(Y^*; Z^*)$ jej optimálne riešenie. Potom je úloha QMP (7) riešiteľná, jej optimálne riešenie je $X^* = Y^*(Z^*)^T$ a $val(7) = val(31)$.

Dôkaz.
Ako prvé si všimnime, že systém rovností:

$$Tr(Z^T(E_{ij}^r + E_{ji}^r)Z) = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r$$

môžeme vďaka cyklickej vlastnosti stopy matice prepísať ako

$$Tr((E_{ij}^r + E_{ji}^r)ZZ^T) = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r.$$

Kedže matica ZZ^T je symetrická, predchádzajúca rovnosť, ako sme ukázali v prvej kapitole našej práce, je ekvivalentná s rovnosťou:

$$Z^T Z = ZZ^T = I_r.$$

[1] $val(P)$ označuje optimálnu hodnotu účelovej funkcie optimalizačnej úlohy: $(P) : \min\{f(x) : x \in C\}$. Optimálna hodnota sa však nemusí vždy nadobudnúť. Je teda vhodnejšie označiť $val(P) = \inf_{x\in\mathcal{P}} f(x)$.

Tým sme našli podmienku na maticu Z. Teraz sa pozrime postupne na obe časti lemy.

1. Nech X^* je optimálne riešenie úlohy (7). Pre každé $(Y; Z)$, $Y \in R^{n\times r}, Z \in R^{r\times r}$, ktoré vyhovujú podmienkam úlohy (31), vrátane podmienky $Z^T Z = I_r$, môžeme ukázať:

$$f_0^H(Y; Z) \overset{(33)}{=} f_0(YZ^T) \geq f_0(X^*) \overset{(32)}{=} f_0^H(X^*; I_r).$$

   Z čoho dostávame, že $(X^*; I_r)$ je optimálne riešenie úlohy (31) a $val(7) = val(31)$.

2. Nech je (Y^*, Z^*), $Y \in R^{n\times r}, Z \in R^{r\times r}$, optimálne riešenie úlohy (31). Určíme X^* ako násobok týchto dvoch matíc, teda $X^* = Y^*(Z^*)^T$. Potom pre každé $X \in R^{n\times r}$, ktoré je prípustné pre úlohu (7) platí:

$$f_0(X) \overset{(32)}{=} f_0^H(X; I) \geq f_0^H(Y^*; Z^*) \overset{(33)}{=} f_0(Y^*(Z^*)^T) = f_0(X^*),$$

   dostávame, že X^* je optimálne riešenie úlohy (7) a $val(31) = val(7)$.

□

O spojením oboch častí Lemy (2.6) hovorí Dôsledok (2.7), informácie z ktorého budú neskôr využité pri konštrukcii semidefinitnej relaxácie pre úlohu QMP.

Dôsledok 2.7. Úloha QMP (7) je riešiteľná práve vtedy, keď je úloha (31) riešiteľná a v tom prípade sa optimálna hodnota úlohy QMP (7) rovná optimálnej hodnote úlohy (31). Teda val(QMP) = val(31).

2.2 Konštrukcia SDR

Naším cieľom stále zostáva zostrojiť semidefinitnú relaxáciu úlohy kvadratického maticového programovania. Vďaka Dôsledku 2.7 vieme, že môžeme pracovať s homogenizovanou úlohou QMP (31) namiesto úlohy QMP (7).

Poznámka 2.8. Pripomíname, že účelovú funkciu aj ohraničenia úlohy (31) tvoria funcie (28), ktoré majú tvar:

$$f^H(X; Z) = Tr(X^T AX) + 2Tr(Z^T B^T X) + \frac{c}{r} Tr(Z^T Z),$$

kde $X \in R^{n\times r}, Z \in R^{r\times r}$.

Ku každej z nich tiež prislúcha matica tvaru (29):

$$M(f^H) = \begin{pmatrix} A & B \\ B^T & \frac{c}{r} I_r \end{pmatrix}.$$

Označme $W = (X; Z) \in \mathbb{R}^{(n+r)\times r}$. Potom môžeme úlohu (31) prepísať ako:

$$\begin{array}{lll} \min & Tr(W^T M(f_0^H) W) & \\ & Tr(W^T M(f_i^H) W) & \leq \alpha_i, \quad i \in I, \\ & Tr(W^T M(f_j^H) W) & = \alpha_j, \quad j \in E, \\ & Tr(W^T N_{ij} W) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r, \\ & W \in \mathbb{R}^{(n+r)\times r}, & \end{array} \tag{34}$$

kde matica M je definovaná v (29) a N_{ij} definujeme ako:

$$N_{ij} = \begin{pmatrix} 0_{n\times n} & 0_{n\times r} \\ 0_{r\times n} & (E_{ij}^r + E_{ji}^r) \end{pmatrix}, 1 \leq i \leq j \leq r.$$

Vďaka vlastnostiam stopy, môžeme zameniť poradie násobenia matice W^T. Z účelovej funkcie aj funkcií ohraničení v úlohe (34) vzniknú funkcie tvaru:

$$f^H = Tr(M(f^H) W W^T), \tag{35}$$

kde WW^T je symetrická kladne definitná matica.

Posledným krokom je substitúcia matice WW^T za premennú U: $U = WW^T$, pričom $U \in \mathbb{S}_+^{n+r}$:

$$f^H = Tr(M(f^H) U). \tag{36}$$

Odvodením z úlohy (31) a použitím funkcií tvaru (36) sme konečne dostali semidefinitnú relaxáciu úlohy kvadratického maticového programovania (7):

$$(SDR) \qquad \begin{array}{lll} \min & Tr(M(f_0) U) & \\ & Tr(M(f_i) U) & \leq \alpha_i, \quad i \in I, \\ & Tr(M(f_j) U) & = \alpha_j, \quad j \in E, \\ & Tr(N_{ij} U) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r. \\ & U \in \mathbb{S}_+^{n+r} & \end{array} \tag{37}$$

2.3 Presnosť Semidefinitnej Relaxácie úlohy QMP

V tejto časti v stručnosti zhrnieme, že za platnosti istých nenáročných podmienok, je SDR úlohy QMP rádu r s najviac r ohraničeniami presná. Ak to však chceme ukázať, potrebujeme potvrdiť, že úloha (SDR) (37) dáva riešenie s hodnosťou menšou alebo rovnou r. To nás nabáda k otázke uvažovania existencie riešení s nízkou hodnosťou pre úlohy semidefinitného programovania. Táto oblasť bola študovaná vo viacerých dielach, na ktoré sa odkazujú autori článku [1], ako príklad uvádzame [11].

Uvažujme všeobecnú formu úlohy SDP:

$$\begin{aligned} \min \quad & Tr(C_0U) \\ & Tr(C_iU) \quad \leq \alpha_i, \quad i \in I, \\ & Tr(C_jU) \quad = \alpha_j, \quad j \in E, \\ & U \in \mathbb{S}^n_+, \end{aligned} \tag{38}$$

kde $C_i \in \mathbb{S}^n$, $\alpha_i \in \mathbb{R}$.

V článku [11] autor ukazuje, že pokiaľ je počet ohraničení nižší ako horná hranica, ktorá je daná ako určitá kvadratická funkcia v premennej r, potom existuje riešenie úlohy (38) s hodnosťou menšou ako r. Dôkaz tohto výsledku je konštruktíny, náročný a založený na procese znižovania hodnosti. Hodnosť sa znižuje tak, aby sme našli extrémne body konvexnej množiny, ktorá prislúcha k $\mathbb{S}^n_+ \cap A$, kde A je afinný priestor. Pre úplnosť a tiež kedže proces znižovania hodnosti je súčasťou algoritmu na riešenie úloh QMP, autori článku [1] spomínajú aj tvrdenia z [11] aj algoritmus na znižovanie hodnosti.

Vďaka výsledkom z článku [11] sme schopní ukázať, že úlohy QMP rádu r s najviac r ohraničeniami dávajú presnú SDR za predpokladu splnenia podmienok uvedených v [1]. Potom sa dá ukázať, že ak je úloha SDR riešiteľná, potom je riešiteľná aj k nej prislúchajúca úloha QMP a val(SDR)=val(QMP). Presnú formuláciu lemy aj jej dôkaz uvádzajú autori v [1].

2.4 Algoritmus riešenia úlohy QMP

Teraz môžeme popísať algoritmus s pracovným názvom SOL-QMP, ktorého teoretické predpoklady sme spomínali v predchádzajúcej časti. Algoritmus SOL-QMP slúži na získanie riešenia úlohy QMP z jej semidefinitnej relaxácie. Je založený na inom algoritme - algoritme na znižovanie hodnosti RED, ktorý uvádzame v dodatku práce.

Oba algoritmy sú uvedené v článku [1] ako vhodný postup na riešenie úlohy QMP. My funkčnosť SOL-QMP overíme neskôr v práci.

Algoritmus SOL-QMP

1. Vyriešiť úlohu semidefinitného programovania SDR (37) a nájsť optimálne riešenie $U^* \in S_+^{n+r}$.

2. Prejsť algoritmom RED so vstupom U^* a nájsť optimálne riešenie $U_1^* \in S_+^{n+r}$, pre ktoré platí, že hodnosť $U_1^* \leq r$.

3. Spočítať rozklad $U_1^* = WW^T$, kde $W \in R^{(n+r)\times r}$.

4. Nech $W = (Y; Z)$, kde $Y \in R^{n\times r}$ a $Z \in R^{r\times r}$. Dostávame optimálne riešenie úlohy QMP: $X^* = YZ^T$.

3 Riešenie Ortogonálnej Procrutovej úlohy

V predchádzajúcich kapitolách sme sa venovali odvodeniu postupu na riešenie úloh kvadratického maticového programovania QMP. V tejto časti by sme chceli odvodený postup aplikovať a overiť jeho funkčnosť pri riešení Ortogonálnej Procrutovej úlohy z prvej kapitoly našej práce.

Ako prvé potrebujeme nájsť vhodnú úlohu, najlepšie takú, pri ktorej poznáme jej presné riešenie. Potom pre túto úlohu nájdeme jej semidefinitnú relaxáciu, tak, aby sme mohli aplikovať algoritmus SOL-QMP. Optimálne riešenie, ktoré získame algoritmom môžeme porovnať s presným riešením a tak určiť presnosť algoritmu.

3.1 Generovanie testovacej úlohy.

Na začiatok pripomíname, že Ortogonálna Procrutova úloha je úloha tvaru:

$$\begin{aligned} \min \quad & \|AX - B\|_F^2 \\ & X^T X = I_r \end{aligned} \tag{39}$$

s premennou $X \in R^{r\times r}$, kde $A \in R^{n\times r}$, $B \in R^{n\times r}$.

Aby sme mohli určiť presnosť algoritmu, vytvorili sme si vlastnú Ortogonálnu Procrutovu úlohu, pri ktorej poznáme jej riešenie, presné $\overline{X}$. To sme najskôr vypočítali ako násobok rotačných matíc R_1, R_2, R_3 s náhodne zvolenými uhlami φ, ψ, θ.

$$\overline{X} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) \\ 0 & -\sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

Maticu $A \in R^{n\times r}$ sme určili ako ľubovoľnú obdĺžnikovú maticu. Maticu B sme určili ako:

$$B = A\overline{X}.$$

Po dosadení matíc A a B do úlohy (10) je úloha daná a naším cieľom je nájsť jej optimálne riešenie X^*. To potom môžeme porovnať s presným riešením $\overline{X}$.

3.2 Formulácia úlohy ako SDR.

Na to, aby sme mohli použiť algoritmus SOL-QMP, potrebujeme Ortogonálnu Procrutovu úlohu preformulovať do tvaru všeobecnej úlohy QMP. Následne nájdeme jej

semidefinitnú relaxáciu, ktorá bude vstupovať do algoritmu. Po prejdení algoritmom SOL-QMP, by sme mali dostať optimálne riešenie úlohy X^*. Začnime preformulovaním úlohy do tvaru QMP. Na pripomenutie, úloha QMP má tvar:

$$\begin{array}{lll} \min & Tr(X^T A_0 X) + 2Tr(B_0^T X) + c_0 & \\ & Tr(X^T A_i X) + 2Tr(B_i^T X) + c_i \quad \leq \quad \alpha_i, & i \in I \\ & Tr(X^T A_j X) + 2Tr(B_j^T X) + c_j \quad = \quad \alpha_j, & j \in E \\ & X \in R^{n \times r} & \end{array} \tag{40}$$

kde $A_i = A_i^T$, $B_i \in R^{n \times r}$, α_i, $c_i \in R$, $i \in \{0\} \cup I \cup E$.
Ortogonálnu Procrutovu úlohu v tvare úlohy QMP sme už odvádzali v prvej kapitole našej práce. Vďaka tomu vieme, že má tvar:

$$\begin{array}{lll} \min & Tr(X^T A^T A X) - 2Tr(B^T A X) + Tr(B^T B) & \\ & Tr(X^T (E_{ij}^r + E_{ji}^r) X) \quad = \quad 2\delta_{ij}, & 1 \leq i, j \leq r \\ & X \in R^{r \times r} & \end{array} \tag{41}$$

Môžeme si všimnúť, že pri Procrutovej úlohe uvažujeme len ohraničenie v tvare rovnosti. Tiež pri porovnaní všeobecnej úlohy QMP a úlohy (41) si možeme všimnúť substitúcie, ktoré využijeme pri hľadaní semidefinitnej relaxácie SDR:

- $A_0 = A^T A$
- $B_0^T = -B^T A$
- $c_0 = Tr(B^T B)$
- $A_j = E_{ij}^r + E_{ji}^r$

Pripomíname, z druhej kapitoly našej práce, že SDR pre všeobecnú úlohu QMP má tvar:

$$\begin{array}{lll} & \min & Tr(M(f_0)U) \\ & & Tr(M(f_i)U) \quad \leq \alpha_i, \quad i \in I, \\ (SDR) & & Tr(M(f_j)U) \quad = \alpha_j, \quad j \in E, \\ & & Tr(N_{ij}U) \quad = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r \\ & & U \in \mathbb{S}_+^{n+r}, \end{array} \tag{42}$$

kde matica $M(f_i)$ a matica N sú tvaru:

$$M(f^H) = \begin{pmatrix} A & B \\ B^T & \frac{c}{r} I_r \end{pmatrix}, N_{ij} = \begin{pmatrix} 0_{n \times n} & 0_{n \times r} \\ 0_{r \times n} & (E_{ij}^r + E_{ji}^r) \end{pmatrix}, 1 \leq i \leq j \leq r. \tag{43}$$

Skúsme úlohu (42) spolu s maticami z (43) prepísať ako SDR pre Ortogonálnu Procrutovu úlohu (41). Vieme, že ohraničenie v tvare nerovnosti nemusíme brať v úvahu a tiež, že účelovú funkciu a ostatné funkcie ohraničení vyjadríme pomocou uvedených substitúcií. Dostávame úlohu tvaru:

$$(SDR_P) \qquad \begin{array}{lll} \min & Tr(M(f_0)U) & \\ & Tr(M(f_1)U) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r \\ & Tr(N_{ij}U) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r \\ & U \in \mathbb{S}_+^{n+r}, & \end{array} \tag{44}$$

s maticami $M(f_i)$ a N tvaru:

$$M(f_0) = \begin{pmatrix} A^TA & -A^TB \\ -B^TA & \frac{Tr(B^TB)}{r} I_r \end{pmatrix}, M(f_1) = \begin{pmatrix} E_{ij}^r + E_{ji}^r & 0 \\ 0 & 0 \end{pmatrix}, N_{ij} = \begin{pmatrix} 0_{n\times n} & 0_{n\times r} \\ 0_{r\times n} & (E_{ij}^r + E_{ji}^r) \end{pmatrix}.$$

Úloha (44) potom vstupuje do algoritmu SOL-QMP naprogramovaného v softvéri Matlab pri hľadaní optimálneho riešenia Ortogonálnej Procrutovej úlohy.

3.3 Alternatívna semidefinitná relaxácia

Ortogonálna Procrutova úloha môže byť riešená aj inak ako metódami na riešenie úloh QMP, resp. algoritmom SOL-QMP. Ďalší spôsob, ktorý sme použili na jej riešenie, tak aby sme vedeli porovnať presnosť SOL-QMP, má nasledujúci tvar:

$$\begin{array}{ll} \min & Tr(XX^TA^TA) - 2Tr(B^TAX) + Tr(B^TB) \\ & XX^T = I \\ & X \in R^{r\times r} \end{array} \tag{45}$$

Je to ortogonálna Procrutova úloha, ale vďaka ortogonálnosti matice X sme mohli ohraničenie $X^TX = I$ zameniť za

$$XX^T = I. \tag{46}$$

Tiež vďaka vlastnostiam stopy matice sme zamenili poradie násobenia v kvadratickom člene, ktorý sa potom po využití (46) redukuje na konštantu a teda ho môžeme zanedbať ďalšom pri hľadaní minima účelovej funkcie.

Ohraničenie (46) však stále nie je konvexné. Preto sme ho aproximovali, tak aby sme

našli konvexnú relaxáciu úlohy (45) do nasledujúceho tvaru:

$$XX^T \preceq I.$$

Odčítaním XX^T od oboch strán, dostávame kladnú semidefinitnosť matice $I - XX^T$:

$$I - XX^T \succeq 0.$$

Prenásobenie matice XX^T maticou I^{-1} nebude mať vplyv na výsledok, avšak my dostávame rovnosť:

$$I - XI^{-1}X^T \succeq 0,$$

ktorá spĺňa podmienky z Vety o Schurovom doplnku, uvádzanej v prílohe našej práce. Kedže máme, že matica $I - XI^{-1}X^T$ je kladne semidefinitná a to platí aj pre maticu I, z Vety o Schurovom doplnku možeme povedať:

$$\begin{pmatrix} I & X^T \\ X & I \end{pmatrix} \succeq 0. \tag{47}$$

Relaxácia úlohy (45) vstupujúca do našich výpočtov, s ktorou aj porovnávame prenosť algoritmu SOL-QMP, má teda tvar:

$$\begin{aligned} \min \quad & Tr(B^TB) - 2Tr(B^TAX)) \\ & \begin{pmatrix} I & X^T \\ X & I \end{pmatrix} \succeq 0, \end{aligned} \tag{48}$$

kde $X \in R^{r \times r}$.

3.4 Výsledky a ich porovnania

Oba spôsoby riešenia sme naprogramovali v softvéry Matlab a optimalizačné úlohy riešili pomocou jeho rozšírenia pre modelovanie konvexných úloh CVX.

V oboch prípadoch sme zgenerovali 100 vstupných úloh s maticami s rôznymi rozmermi n a r a merali presnosť výsledkov pre jednotlivé prípady. Ako presnosť sme brali Frobeniovu normu $\|AX^* - B\|_F$, kde X^* je výsledná matica algoritmu. Výsledná presnosť pre maticu s rozmermi n a r je priemer presností získaných pri 100 opakovaniach algoritmu. Pri algoritme SOL-QMP sme nepoužili algoritmus na znižovanie hodnosti RED, ako však uvidíme, aj bez neho dával veľmi presné výsledky.

Prvý prípad je pre fixné $r = 3$. Teda rotačná matica $\overline{X}$ a tiež výsledná ortogonálna matica X^* majú rozmer 3×3.

Zvolili sme si niekoľko n, konkrétne $n = \{1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 20\}$. Pre každé n sme náhodne zgenerovali 100 vstupov a s týmito vstupmi sme prešli aj algoritmom SOL-QMP aj alternatívnou relaxáciou. Presnosť pre jednotlivé prípady vypočítanú ako priemer zo 100 získaných vidíme v Tabuľke 1 a 2. Kedže presnosť je v našom

	n=1	n=2	n=3	n=4	n=5	n=6
SOL - QMP	1.593e-11	5.697e-10	4.183e-10	4.048e-10	5.137e-11	5.956e-11
Alternatívna rel.	1.939e-06	8.619e-09	1.458e-08	6.556e-09	3.572e-09	4.628e-09

Tabuľka 1: Priemerná chyba pre matice veľkosti $n \times 3$

	n=7	n=8	n=10	n=15	n=20	Priemer
SOL - QMP	8.251e-11	9.045e-11	1.020e-10	1.620e-09	2.267e-10	3.310e-10
Alternatívna rel.	5.649e-09	4.470e-09	5.672e-09	7.969e-09	3.927e-09	1.822e-07

Tabuľka 2: Priemerná chyba pre matice veľkosti $n \times 3$, kde $n \geq 7$

ponímaní "priemerná chyba", v ideálnom prípade by mala byť nulová. Teda čím má nižšiu hodnotu, tým je metóda presnejšia. Z Tabuľky 1 a 2 vidíme, že oba prístupy sú vhodné, SOL-QMP však pre každé n dáva presnejšie výsledky. Presnosť alebo priemerná chyba sú rádovo podobné pre všetky n v rámci jednotlivých metód.

V druhom prípade sme zvolili fixné $r = 4$. Rovnako ako v prvom prípade sme generovali 100 úloh pre rôzne n, kde $n \in \{1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 20\}$ a presnosť oboch metód vypočítali ako priemernú chybu cez Frobeniovu normu. Výsledky možeme vidieť v Tabuľkách 3 a 4. Môžeme si všimnúť, že oba algoritmy sú v tomto prípade stále presné.

	n=1	n=2	n=3	n=4	n=5	n=6
SOL - QMP	9.699e-10	7.428e-10	2.931e-09	3.962e-09	1.284e-09	2.240e-09
Alternatívna rel.	4.812e-05	3.475e-05	2.310e-06	4.367e-09	2.869e-09	4.888e-09

Tabuľka 3: Priemerná chyba pre matice veľkosti $n \times 4$

	n=7	n=8	n=10	n=15	n=20	Priemer
SOL - QMP	2.332e-09	3.044e-09	4.161e-09	3.210e-09	3.686e-09	2.597e-09
Alternatívna rel.	4.862e-09	6.216e-09	7.328e-09	9.930e-09	5.588e-09	7.748e-06

Tabuľka 4: Priemerná chyba pre matice veľkosti $n \times 4$, kde $n \geq 7$

SOL-QMP je výrazne presnejší pri rozmere $n = 1$, teda, keď A vstupujúce do úlohy je vektor. Pre rozmer $n \geq 4$ sú oba algoritmy rádovo rovnako presné, pričom SOL-QMP je približne dvakrát presnejší. Presnosť, resp. priemerná chyba je približne rovnaká pre všetky $n \geq 4$ pre SOL-QMP aj druhý prístup. Pre $n \leq 3$ má SOL-QMP výrazne lepšie výsledky.

V treťom a poslednom prípade, ktorým sme sa zaoberali v našej práci, sme zvolili rozmer rotačnej matice väčší, konkrétne $r = 10$. Postupovali sme rovnako ako v prvých dvoch prípadoch. Postupne sme zvolili n z množiny $\{1, 2, 3, 4, 5, 6, 7, 8, 10, 15, 20\}$ a pre každé n sme zgenerovali 100 úloh. Presnosť bola vypočítaná ako priemerná chyba cez Frobeniovu normu. Výsledná presnosť pre jednotlivé n, ktorá je uvedená v Tabuľkách 5 a 6, je priemerom presností zo 100 generovaných úloh.

	n=1	n=2	n=3	n=4	n=5	n=6
SOL - QMP	1.18e-07	2.17e-04	1.51e-04	3.22e-05	9.34e-05	1.53e-09
Alternatívna rel.	2.17e-04	6.67e-04	1.32e-03	9.14e-04	3.21e-04	3.44e-04

Tabuľka 5: Priemerná chyba pre matice veľkosti $n \times 10$

	n=7	n=8	n=10	n=15	n=20	Priemer
SOL - QMP	1.987e-09	9.882e-09	2.263e-08	2.829e-09	3.699e-09	4.488e-05
Alternatívna rel.	7.625e-04	1.361e-04	2.624e-08	5.390e-08	5.405e-09	4.256e-04

Tabuľka 6: Priemerná chyba pre matice veľkosti $n \times 10$, kde $n \geq 7$

Všimnime si, že pre väčšie n, teda $n \geq 6$ má SOL-QMP presnosť rádovo rovnakú ako pri prvých dvoch prípadoch, teda $e - 09$. Pre nižšie n však dosahuje výrazne horšie výsledky (v porovnaní s predchádzajúcimi dvoma prípadmi). Presnosť alternatívnej

relaxácie je pre nižšie n horšia ako v predchádzajúcich prípadoch, rovnako aj v porovnaní so SOL-QMP. Pre vysoké n je však stále veľmi presná a schopná konkurovať SOL-QMP.

V poslednej tabuľke tejto časti, v Tabuľke 7 porovnávame presnosť, resp. priemernú chybu v závislosti od rozmeru r. Sú to priemerné hodnoty z troch prípadov, ktoré sme vyššie rozoberali. Vidíme, že oba algoritmy dávajú veľmi dobré výsledky pre nižšie r, pre vyššie hodnoty r je ich presnosť nižšia. Tiež vidíme, že SOL-QMP je vo všeobecnosti presnejší v porovnaní s alternatívnou relaxáciou.

	r=3	r=4	r=10
SOL - QMP	3.310e-10	2.597e-09	4.488e-05
Alternatívna rel.	1.822e-07	7.748e-06	4.256e-04

Tabuľka 7: Priemerná chyba v závislosti od r.

4 Aplikácie z kombinatorickej optimalizácie

Jedným z cieľov našej práce bolo ilustrovať prístupy k riešeniu úloh QMP aj na aplikáciách z oblasti kombinatorickej optimalizácie.

Úlohy kombinatorickej optimalizácie sú optimalizačné úlohy, v ktorých prípustné riešenie môže byť vyjadrené pomocou konceptov z kombinatoriky alebo teórie grafov. Medzi tieto koncepty patria napríklad množiny, kombinácie, permutácie, či vrcholy, hrany, cesty, rezy. Optimálne riešenie pochádza z konečnej množiny riešení. Riešenia sú vo všeobecnosti diskrétne, alebo ich vieme preformulovať do diskrétneho tvaru. Kombinatorická optimalizácia je téma, ktorá má význam napríklad v umelej inteligencii či machine learningu.

My sa v práci venujeme kombinatorickej Procrutovej úlohe, ktorú rieši napríklad autor v [9]. V tejto úlohe kombinatorickej optimalizácie hľadáme vhodnú permutačnú maticu spĺňajúcu podmienky Procrutovej úlohy.

4.1 Kombinatorická Procrutova úloha

V [9] autor hľadá riešenie kombinatorickej Procrutovej úlohy nasledujúceho tvaru:
Nech $A, B \in R^r$. Vieme, že matica B je permutáciou matice A. Nájdite riešenie ortogonálnej Procrutovej úlohy:

$$\begin{aligned} \min \quad & \|AX - B\|_F \\ & X^T = X^{-1} \\ & X_{ij} \in \{0, 1\} \end{aligned} \tag{49}$$

s premennou $X \in R^{r \times r}$, kde X je permutačná matica.

Oproti úlohám z tretej kapitoly máme teraz rozdiel, že žiadame, aby optimálna matica X^*, ktorú dostaneme z algoritmu, bola permutačná. Permutačné matice tvoria špeciálnu podtriedu ortogonálnych matíc so zložkami 0 a 1. Tiež autor [9] uvažuje A, B len ako vektory, v čom sme sa ho rozhodli nasledovať, teda rozmer n je zafixovaný na $n = 1$.

Úlohu sme riešili aj alternatívnou semidefinitnou relaxáciou, aj algoritmom SOL-QMP a porovnali ich výsledky.

4.2 Riešenie pomocou alternatívnej semidefinitnej relaxácie

Prvý spôsob riešenia úlohy (49), ktorý sme zvolili je alternatívnou semidefinitnou relaxáciou. K tomuto spôsobu sa dá jednoduchšie pridať podmienka na permutačnú maticu X^*. Všimnime si, že podmienka $X_{ij} \in \{0, 1\}$ nie je konvexná, autor [9] však nachádza riešenie a aproximuje ju podmienkami

$$Xe = 1, \quad X^T e = 1, \quad X \geq 0, \tag{50}$$

kde e je vektor jednotiek. Jeho nápad používame v našom algoritme aj my a alternatívna semidefinitná relaxácia úlohy (49) nadobúda tvar:

$$\begin{aligned} \min \quad & Tr(B^T B) - 2Tr(B^T AX)) \\ & \begin{pmatrix} I & X^T \\ X & I \end{pmatrix} \succeq 0 \\ & Xe = 1, \quad X^T e = 1 \\ & X \geq 0 \end{aligned} \tag{51}$$

kde $X \in R^{r \times r}$. Takto vstupovala úloha do programu v Matlabe, ktorý úlohu riešil pomocou rozšírenia pre konvexnú optimalizáciu CVX. Nečakane najlepšie výsledky či už v presnosti alebo v správnom určení permutačnej matice program dával, v prípade, keď bola použitá len prvá z troch podmienok, tj. $Xe = 1$.

Pri riešení úlohy a zisťovaní presnosti algoritmu sme postupovali podobne ako v tretej kapitole našej práce. Kedže rozmer n je zafixovaný na $n = 1$, v jednotlivých úlohách sme menili len rozmer ortogonálnej matice r. Úlohy sme riešili pre $r \in \{3, 5, 8, 10, 15, 20, 25\}$. Pre každé r sme zgenerovali 100 úloh, kde sme náhodne určili vektor $A \in R^r$ a náhodnú permutačnú maticu $\overline{X} \in R^{r \times r}$. Vektor B sme vypočítali ako $B = A\overline{X}$. Do algoritmu potom vstupovali vektory A a B, pričom týmto postupom sme zaistili, že riešenie úlohy X^* existuje.

Pre každú zo 100 úloh sme zisťovali presnosť ako Frobeniovu normu $\|AX^* - B\|_F$. Tiež sme zisťovali, či algoritmus vedel správne určiť permutačnú maticu X^* rovnakú, ako bola vstupná matica $\overline{X}$. Presnosť pre jednotlivé r sme vypočítali ako priemer zo 100 generovaných úloh.

Príklad 4.1. Máme daný náhodne vygenerovaný vektor $A = (11, 23, 15)$. K nemu sme náhodne vygenerovali permutačnú maticu $\overline{X}$. Znásobením $A\overline{X}$ dostávame vektor

$B = (15, 11, 23)$.

Vektory A a B sú vstupy do nášho algoritmu a chceli by sme pomocou neho nájsť permutačnú maticu X^*, tak, aby chyba vypočítaná ako Frobeniova norma $\|AX^* - B\|_F$ bola čo najmenšia.

$$\overline{X} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} X_{alg} = \begin{pmatrix} 0.4286 & 0.7143 & -0.1429 \\ 0.2143 & -0.1429 & 0.9286 \\ 0.3571 & 0.4286 & 0.2143 \end{pmatrix} X^* = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Z algoritmu dostávame maticu X_{alg}, ktorej chyba je $1.0592e-07$. Nie je to permutačná matica, ako by sme potrebovali, vieme ju na ňu však zaokrúhliť s ohľadom na to, aby v každom riadku a stĺpci bola práve jedna jednotka a zvyšok nuly.

Na zaokrúhľovanie sme použili nasledujúci algoritmus, pričom v tomto príklade je $r = 3$:

0. Maticu X^* určíme ako maticu rozmeru $r \times r$ obsahujúcu len 0. Opakujeme r-krát pokyny 1 až 3:
1. Nájdeme najväčší prvok matice X_{alg}, nech je to x_{ij}.
2. Prvok y_{ij} matice X^* položíme rovný 1.
3. Všetky prvky matice X_{alg} v riadku i a v stĺpci j položíme rovné 0.

Po zaokrúhlení dostávame permutačnú maticu X^*, ktorá je zhodná s $\overline{X}$ a teda aj jej chyba je rovná nule.

Z Príkladu 4.1 vidíme, že matice, ktoré vypočítala optimalizačná úloha je potrebné vhodne zaokrúhliť, aby sa z nich stala permutačná matica. V algoritme sme teda na konci zaokrúhlili prvky matice X_{alg} podľa najväčších na 1 a 0 tak, aby bola dodržaná kolmosť riadkov a stĺpcov výslednej matice X^*.

4.3 Výsledky pre alternatívnu semidefinitnú relaxáciu

V Tabuľke 8 môžeme vidieť presnosť resp. priemernú chybu pre matice X_{alg} a X^* pre rôzne rozmery r permutačnej matice. Tiež uvádzame počet správne určených permutačných matíc po zaokrúhlení.

Vidíme, že algoritmus je presný, s rastúcim r jeho presnosť postupne klesá. Pre všetky nižšie rozmery určil správne permutačnú maticu X^* rovnakú, ako bola počiatočná $\overline{X}$,

	r=3	r=5	r=8	r=10	r=15	r=20	r=25
Chyba X_{alg}	3.114e-07	7.244e-07	2.986e-06	6.561e-06	1.570e-05	1.495e-05	1.40e-04
Chyba X^*	0	0	0	0	0	0.01414	0.03162
Počet spr.	100	100	100	100	100	99	99

Tabuľka 8: ASR: Priemerná chyba X_{alg} a X^* pre rôzne rozmery r permutačnej matice

z ktorej sme vychádzali. Kedže matice X^* a $\overline{X}$ sú rovnaké, Chyba X^* je pre väčšinu romerov r rovná nule, teda ideálnemu prípadu.

V Tabuľke 8 sú uvedené len výsledky, keď vektor A vstupujúci do úlohy (51) mal všetky zložky rôzne. Prípad, keď do algoritmu mohol vstupovať aj vektor s viacerými zložkami rovnakej hodnoty uvádzame v Tabuľke 9. Opakovanie zložiek vo vektore A ovplyvnilo

	r=4	r=6	r=10
Chyba X_{alg}	3.315e-07	1.238e-06	5.402e-06
Chyba X^*	0	0	0
Počet spr.	90	72	54

Tabuľka 9: ASR: Priemerná chyba X_{alg} a X^* pre rôzne rozmery r permutačnej matice s aj opakujúcimi sa zložkami vo vektore A

hľadanie presnej permutačnej matice, rovnakej, ako bola počiatočná matica $\overline{X}$. Vidíme však, že matice, ktoré algoritmus našiel sú stále správnym riešením, nakoľko priemerná chyba X^* pre uvádzané rozmery je rovná nule. Presnosť resp. priemerná chyba pre X_{alg} zostáva na podobných hodnotách ako v Tabuľke 8.

4.4 Riešenie pomocou algoritmu SOL-QMP

Druhý spôsob, ktorým sme riešili úlohu (49) je algoritmus SOL-QMP. Aj v tomto prípade máme problém s ohraničením $X_{ij} \in \{0, 1\}$, ktoré potrebujeme preformulovať do tvaru úlohy QMP. Opäť využijeme poznatok z [9], že ohraničenie možeme prepísať pomocou rovníc (50), teda:

$$Xe = 1, \quad X^T e = 1, \quad X \geq 0,$$

kde e je vektor jednotiek. Ohraničenia (50) však ešte potrebujeme pretransformovať do tvaru úlohy QMP. Použijeme na to nasledujúci spôsob:

- Ohraničenie $Xe = 1$ môžeme prepísať ako $Tr(B_i X) = 1$, kde B_i je matica núl s jednotkami v jej i-tom stĺpci. $Tr(B_i X)$ potom reprezentuje súčet zložiek v jej i-tom riadku.
- Ohraničenie $X^T e = 1$ môžeme prepísať ako $Tr(D_i X) = 1$, kde D_i je matica núl s jednotkami v jej i-tom riadku. $Tr(D_i X)$ potom reprezentuje súčet zložiek v jej i-tom stĺpci.
- Ohraničenie $X \geq 0$, reprezentujúce nezápornosť zložiek X, vyjadríme tak, že zložku X_{ij} prepíšeme ako $Tr(E_{ij}X)$ a prehlásime ju za nezápornú: $Tr(E_{ij}X) \geq 0$.

Úlohu (49) možeme teraz vyjadriť ako úlohu kvadratického maticového programovania v tvare:

$$\begin{aligned} \min \quad & Tr(X^T A^T A X) - 2Tr(B^T A X) + Tr(B^T B) \\ & Tr(X^T (E_{ij}^r + E_{ji}^r) X) = 2\delta_{ij}, \quad 1 \leq i, j \leq r \\ & Tr(B_i X) = 1 \qquad\qquad 1 \leq i \leq r \\ & Tr(D_i X) = 1 \qquad\qquad 1 \leq i \leq r \\ & Tr(E_{ij} X) \geq 0 \qquad\qquad 1 \leq i, j \leq r \\ & X \in R^{r \times r} \end{aligned} \tag{52}$$

Úlohu ešte prepíšeme do tvaru semidefinitnej relaxácie, ktorá potom vstupuje do výpočtu. Oproti prípadu z tretej kapitoly nám pribudli dve ohraničenia na rovnosť a jedno ohraničenie na nerovnosť:

$$(SDR_{KP}) \qquad \begin{aligned} \min \quad & Tr(M(f_0)U) & \\ & Tr(M(f_1)U) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r \\ & Tr(M(f_2)U) & = 1 \\ & Tr(M(f_3)U) & = 1 \\ & Tr(M(f_4)U) & \geq 0 \\ & Tr(N_{ij}U) & = 2\delta_{ij}, \quad 1 \leq i \leq j \leq r \\ & U \in \mathbb{S}_+^{n+r}, & \end{aligned} \tag{53}$$

kde $M(f_2), M(f_3), M(f_4)$ sú matice tvaru:

$$M(f_2) = \begin{pmatrix} 0 & B_i \\ B_i^T & 0 \end{pmatrix} M(f_3) = \begin{pmatrix} 0 & D_i \\ D_i^T & 0 \end{pmatrix} M(f_4) = \begin{pmatrix} 0 & E_{ij} \\ E_{ij}^T & 0 \end{pmatrix} \tag{54}$$

Úloha (53) teraz vstupuje do algoritmu SOL-QMP naprogramovaného v Matlabe, kde sme zisťovali presnosť metódy pri Kombinatorickej Procrutovej úlohe. Tiež sme sledovali, či algoritmus vie nájsť vhodnú permutačnú maticu.

Pri riešení sme postupovali rovnako ako v predchádzajúcom prípade. Rozmer n je zafixovaný na $n = 1$, v jednotlivých úlohách sme menili len rozmer r ortogonálnej matice. Úlohy sme riešili pre $r \in \{3, 5, 8, 10, 15, 20, 25\}$ a pre každé r zgenerovali 100 úloh. V každej z nich sme náhodne určili vektor $A \in R^r$ a permutačnú maticu $\overline{X} \in R^{r \times r}$. Vektor B sme vypočítali ako $B = A\overline{X}$. Do algoritmu potom vstupovali vektory A a B a my máme zaistené, že riešenie úlohy X^* existuje.

Pre každú zo 100 úloh sme zisťovali presnosť ako Frobeniovu normu $\|AX^* - B\|_F$. Tiež sme zisťovali, či algoritmus vedel správne určiť permutačnú maticu X^* rovnakú, ako bola vstupná matica $\overline{X}$. Presnosť pre jednotlivé r sme vypočítali ako priemer zo 100 generovaných úloh.

Rovnako ako pri alternatívnej semidefinitnej relaxácii sme prvky matice X_{alg}, ktorú našla optimalizačná úloha, zaokrúhlili na jednotky a nuly tak, aby výsledná matica X^* bola permutačná.

4.5 Výsledky pre algoritmus SOL-QMP

V Tabuľkách 10 a 11 môžeme vidieť výsledky pre algoritmus SOL-QMP. Tabuľka 10 obsahuje výsledky, keď vektor A vstupujúci do algoritmu obsahoval rôzne prvky, v Tabuľke 11 sa zložky A mohli opakovať. Treba spomenúť, že napriek hodnosti $U \geq r$ sme v algoritme SOL-QMP nepoužili algoritmus na znižovanie hodnosti RED, nedával v tomto prípade dobré výsledky, preto priemerná chyba pre X_{alg} je pomerne vysoká.

Z Tabuľky 10 však vidíme, že algoritmus SOL-QMP, aj napriek tomu správne nachádza permutačné matice, dokonca aj pre vysoké r. Vďaka tomu je priemerná chyba X^* rovná nule pre všetky nami sledované rozmery r.

Výsledky, keď do algoritmu mohol vstupovať aj vektor s rovnakými prvkami, môžeme vidieť v Tabuľke 11. Ako sme už spomínali chyba X_{alg} je pomerne vysoká, kedže sme nezniżovali hodnosť matice U pomocou algoritmu RED. Riadok "Počet X^* spr."hovorí o počte správne nájdených permutačných maticiach, rovnakých, ako bola povôdná matica $\overline{X}$. Keďže sme vstupný vektor A generovali z 30 náhodných čísel je prirodzené,

	r=3	r=5	r=8	r=10	r=15	r=20	r=25
Chyba X_{alg}	1.505e+1	1.978e+1	2.496e+1	2.777e+1	3.44e+1	3.953e+1	4.441e+1
Počet spr.	100	100	100	100	100	100	100
Chyba X^*	0	0	0	0	0	0	0
Rank X*	3	5	8	10	15	20	25
Rank U	6	10	16	20	30	40	50

Tabuľka 10: SOL-QMP: Priemerná chyba, počet správne určených matíc, rank pre rôzne rozmery r perm. matice

že pri vyššom rozmere sa opakovali viackrát a teda bolo viac možných správnych riešení iných ako pôvodná matica $\overline{X}$.

Riadok "Počet X^* s chybou"hovorí o počte chybne nájdených matíc X^*. Teda takých, pre ktoré $\|AX^* - B\|_F$ bolo rôzne od nuly. Objavilo sa pár takých matíc, je možné, že vhodný RED by vedel túto chybu odstrániť.

	r=4	r=6	r=10
Chyba X_{alg}	1.784e+01	2.099e+01	2.804e+01
Počet X^* spr.	86	74	44
Počet X^* s chybou	1	5	7
Chyba X^*	6.589e-01	9.024	3.043e+01
Rank X*	4	6	10
Rank U	8	12	20

Tabuľka 11: SOL-QMP: Priemerná chyba, počet správne určených matíc, rank pre rôzne rozmery r perm. matice, s opakovaním prvkov vo vstupnom vektore

4.6 Porovnanie ASR a SOL-QMP

Kombinatorickú Procrutovu úlohu sme riešili pomocou alternatívnej semidefinitnej relaxácie a algoritmom SOL-QMP. Oba prístupy veľmi dobre nachádzajú vhodné permutačné matice, pokiaľ sa prvky vstupného vektora neopakujú. V tom prípade má jemne navrch SOL-QMP. V prípade, ak vstupný vektor môže obsahovať aj rovnaké zložky, dosahuje postup pomocou ASR výrazne lepšie výsledky. Kedže však bohužiaľ

pri SOL-QMP algoritmus RED nedával dobré výsledky, sme ho z algoritmu vynechali, čo zrejme negatívne ovplyvnilo jeho schopnosť hľadania. Tiež mu to pohoršilo z hľadiska presnosti pri chybe X_{alg} a teda výrazne presnejšie matice X_{alg} dáva postup pomocou ASR.

Záver

Táto diplomová práca sa zamerala na oboznámenie s neveľmi preskúmanou triedou optimalizačných úloh - na úlohy kvadratického maticového programovania (QMP). Úlohy QMP sú vo všeobecnosti nekonvexné a teda ich riešenie vyžaduje zložitejšie postupy alebo aproximácie pri riešení. Cieľom našej práce bolo porovnať rôzne prístupy riešenia úloh QMP a ilustrovať ich na aplikáciách. Práca je rozdelená na štyri kapitoly, v ktorých sme sa snažili cieľ splniť.

V prvej kapitole diplomovej práce sme zadefinovali úlohu kvadratického maticového programovania, transformovali úlohu na jej vektorizovaný ekvivalent a zisťovali za akých podmienok možu byť kvadratické maticové funkcie, ktorá tvoria úlohu QMP konvexné. Tiež sme ukázali dva príklady, v ktorých sa vyskytujú úlohy QMP. Bola to Ortogonálna Procrutova úloha a úloha Robustných najmenších štvorcov. Vychádzali sme hlavne z článku [1]. Kapitola slúži ako úvodné oboznámenie sa s problematikou aj pre čitateľa, ku ktorému sa táto téma zatiaľ nedostala. Ako prínos by sme uviedli prehľadné zhrnutie teórie, ktorej zatiaľ nebolo v literatúre venovanej veľa pozornosti. Tiež za vlastný prínos považujeme vysvetlenie, odvodenie a doplnenie krokov, ktoré boli v [1] vynechané, napríklad Lemu 1.8 či transformáciu úlohy (10) na úlohu (11).

V druhej kapitole práce sme odvodili Semidefininú relaxáciu ako aproximačnú techniku pre úlohu QMP. Uviedli sme viaceré tvrdenia alebo lemy, ktorých zdroj bol väčšinou [1]. Dôkazy niektorých z nich však pochádzajú od nás. Príkladom môže byť dôkaz Lemy 2.4 a Lemy 2.5. Ku koncu kapitoly sme uviedli algoritmus na riešenie úlohy QMP s názvom SOL-QMP. Ten sa neskôr v práci ukázal ako vhodný na riešenie úloh tohto typu.

V tretej kapitole práce sa venujeme aplikácii teoretických poznatkov z predchádzajúcich kapitol. Spôsob riešenia úlohy QMP sme ilustrovali na riešení Ortogonálnej Procrutovej úlohy. V tejto úlohe sme sa snažili nájsť vhodnú rotačnú maticu spĺňajúcu zadané podmienky. Riešenie X^* sme hľadali dvomi spôsobmi. Prvý z nich bol algoritmus SOL-QMP. Druhý postup, ktorý sme v tejto kapitole aj odvodili, sme v práci nazvali alternatívna semidefinitná relaxácia. Ukázalo sa, že oba postupy dávajú veľmi dobré výsledky pri hľadaní čo najpresnejšej rotačnej matice. Výsledky môžeme vidieť v Tabuľke 5 až v Tabuľke 6. Pri porovnaní algoritmov, v Tabuľke 7, sa ukázalo, že

SOL-QMP dáva v tomto prípade presnejšie výsledky ako alternatívna semidefinitná relaxácia.

Posledná, štvrtá kapitola je tiež aplikačná a zaoberá sa riešením Kombinatorickej Procrutovej úlohy. V tejto úlohe sme hľadali vhodnú ortogonálnu maticu, tentokrát však hľadaná matica bola permutačná, teda obsahovala len zložky 0 a 1. Zdrojom úlohy bola publikácia [9]. Úlohu sme rozdelili na dva prípady. Jeden z nich bol, ked vektor vstupujúci do algoritmu obsahoval rôzne zložky, v druhom prípade sa jeho zložky mohli aj opakovať. Aj algoritmus SOL-QMP aj alternatívna semidefinitná relaxácia v prvom prípade nachádzali permutačné matice veľmi presne. Treba však spomenúť, že sme ich najskôr prispôsobili na hľadanie permutačných matíc aj pomocou zaokrúhľovania. V druhom prípade sa lepšie osvedčila alternatívna semidefinitná relaxácia. Dôvodom však mohlo byť aj to, že sme pri SOL-QMP nepoužívali jeho zlepšienie pomocou algoritmu na znižovanie hodnosti RED, ako tvrdili autori [1], nakoľko nám nedával dobré výsledky.

Za prínos tejto práce považujeme prehľadné spracovanie teórie o úlohách kvadratického maticového programovania a ilustrácie ich riešenia na Ortogonálnej Procrutovej úlohe. Tiež sme overili vhodnosť riešenia úloh QMP pomocou algoritmu SOL-QMP z [1] a ukazujeme ďalší možný postup na ich riešenie, alternatívnu semidefinitnú relaxáciu, ktorý dáva tiež dobré výsledky.

Zoznam použitej literatúry

[1] Beck, A.: *Quadratic Matrix Programming*, SIAM Journal on Optimization, 2006, dostupné na internete (7.1.2018): https://ie.technion.ac.il/ becka/papers/quadmat10.pdf

[2] Luo, Z. Q., Ma, W. K., So, A. M. Ch.: *Semidefinite Relaxation of Quadratic Optimization Problems*, IEEE Signal Processing Magazine, 2010, dostupné na internete (7.1.2018): http://www1.se.cuhk.edu.hk/ manchoso/papers/sdrapp-SPM.pdf

[3] Laub, A. J.: *Matrix Analysis for Scientists and Engineers*, SIAM, 2005, dostupné na internete (30.1.2018): http://www.siam.org/books/textbooks/OT91sample.pdf

[4] Stehlík, M.: *Conic Relaxations*, Diplomová práca, Univerzita Komenského, Bratislava, 2016

[5] Schrönemann, P. H.: *A Generalized Solution of the Orthogonal Procrustes Problem*, Psychometrika, 1966, 31:1-10, dostupné na internete (16.1.2018): http://web.stanford.edu/class/cs273/refs/procrustes.pdf

[6] Ghaoui, L. E., Lebret, H.: *Robust Solutions to Least-Squares Problems with Uncertain Data*, SIAM Journal on Matrix Analysis and Applications, 1997, 18(4), 1035–1064, dostupné na internete (27.1.2018): https://people.eecs.berkeley.edu/ elghaoui/Pubs/rob-ls.pdf

[7] Berens, E. M.: *The Myths and Legends of Ancient Greece and Rome*, MetaLibri Digital Library, 2009, dostupné na internete (28.11.2018): https://www.ibiblio.org/ml/libri/b/BerensEM-MythsLegends-s.pdf

[8] Ben-Tal, A., Nemirovski A.: *Lectures on Modern Convex Optimization*, MPS-SIAM Series on Optimization, 2001, dostupné na internete (7.1.2019): http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D90AC40B371F40E071CAF13E450E858C?doi=10.1.1.721.9910&rep=rep1&type=pdf

[9] Dattorro, J.: *Convex Optimization and Euclidean Distance Geometry*, Meboo Publishing, 2005-2015, 290-292, dostupné na internete (1.5.2019): https://ccrma.stanford.edu/~dattorro/0976401304_v2016.12.18.pdf

[10] Zhang, F.: *Matrix Theory*, Springer, 1999, 175, dostupné na internete (1.5.2019): https://link.springer.com/content/pdf/10.1007%2F978-1-4757-5797-2.pdf

[11] Pataki, G.: *The geometry of semidefinite programming.*, Handbook of Semidefinite Programming, Springer, 2000, 29-65

Príloha A

A.1 Kroneckerov súčin

Kroneckerov súčin je maticová operácia, do ktorej vstupujú dve matice ľubovoľnej veľkosti a ich výsledkom je jedna "bloková"matica. Kroneckerov súčin má označenie $\otimes$. Nech A je matica veľkosti $m \times n$ a B je matica veľkosti $p \times q$. Potom Kroneckerov súčin $A \otimes B$ je matica veľkosti $mp \times nq$.

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{pmatrix}$$

Príklad:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} & 2 \cdot \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} \\ 3 \cdot \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} & 4 \cdot \begin{pmatrix} 0 & 5 \\ 6 & 7 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{pmatrix}$$

A.1.1 Niektoré vlastnosti Kroneckerovho súčinu

Kroneckerov súčin má veľa zaujímavých vlastností, ktoré využívame aj v našej práci. Mnohé z nich sú uvádzané a dokázané v literatúre. Príkladom takej literatúry je [3], Kapitola 13, z ktorej sme čerpali a uvádzajú sa tam aj dôkazy nasledujúcich tvrdení.

- ([3],Theorem 13.3.): Nech $A \in R^{m\times n}$, $B \in R^{r\times s}$, $C \in R^{n\times p}$, a $D \in R^{s\times t}$. Potom

$$(A \otimes B)(C \otimes D) = AC \otimes BD (\in R^{mr\times pt}). \tag{A.1}$$

- ([3],Theorem 13.4.): Pre všetky A, B platí: $(A \otimes B)^T = A^T \otimes B^T$

- ([3],Theorem 13.6.): Ak sú A, B regulárne platí: $(B \otimes C)^{-1} = B^{-1} \otimes C^{-1}$

- ([3],Theorem 13.12.): Nech vlastné hodnoty matice $A \in R^{n\times n}$ sú λ_i, $i \in n$ a vlastné hodnoty matice $B \in R^{m\times m}$ sú μ_j, $j \in m$. Potom $A \otimes B$ má nm vlastných hodnôt:

  $$\lambda_1\mu_1, \ldots, \lambda_1\mu_m, \ldots, \lambda_2\mu_1, \ldots, \lambda_2\mu_m, \ldots, \lambda_n\mu_m.$$

  Navyše, ak $x_1, \ldots, x_p$ sú lineárne nezávislé vlastné vektory matice A, prislúchajúce k $\lambda_1, \ldots, \lambda_p$ $(p \leq n)$ a $z_1, \ldots, z_q$ sú lineárne nezávislé vlastné vektory matice B, prislúchajúce k $\mu_1, \ldots, \mu_q$ $(q \leq m)$, potom $x_i \otimes z_j \in R^{mn}$ sú lineárne nezávislé vlastné vektory $A \otimes B$ prislúchajúce k $\lambda_i\mu_j$, $i \in p$, $j \in q$.
  Dôkaz:

  $$(A \otimes B)(x \otimes z) = Ax \otimes Bz = \lambda x \otimes \mu z = \lambda\mu(x \otimes z).$$

  □

  Dôsledok: Nech $A \in R^{n\times n}$ a $B \in R^{m\times m}$. Potom

  1. $Tr(A \otimes B) = (TrA)(TrB) = Tr(B \otimes A)$
  2. $det(A \otimes B) = (detA)^m(detB)^n = det(B \otimes A)$

A.2 Algoritmus RED

V tejto časti preskúmame algoritmus na znižovanie hodnosti z [1] na riešenie úloh SDP tvaru (40). Základný predpoklad, ktorý zaisťuje platnosť procesu, je to, že úloha (40) je riešiteľná a že platí $|I_1| + |E_1| \leq \binom{r+2}{2} - 1$.

Algoritmus RED

Vstup: Optimálne riešenie úlohy (40): X_0.

Výstup: Optimálne riešenie úlohy (40): X^*, pre ktoré platí, že hodnosť $X^* \leq r$.

1. Ak je hodnosť $X_0 \leq r$, choď na krok 3. Inak choď na krok 2.
2. Kým je hodnosť $X_0 > r$ opakuj nasledovné príkazy:
   - Urči d ako hodnosť matice X_0.
   - Spočítaj rozklad matice X_0: $X_0 = UU^T$, kde $U \in R^{n\times d}$.

- Nájdi netriviálne riešenie Z_0 pre systém homogénnych lineárnych rovníc v $d \times d$ symetrickej matici premenných Z ($Z = Z^T$):

$$Tr(U^T C_i U Z) = 0, i \in I_1 \cup E_1.$$

- Ak $Z_0 \succeq 0$, tak urči $W \leftarrow -Z_0$. Inak urči $W \leftarrow Z_0$.
- Urči $X_0 \leftarrow U(I + \beta W)U^T$, kde $\beta = -1/\lambda_{min}(W)$.

3. Urči $X^* \leftarrow X_0$ a KONIEC.

Poznamenajme, že algoritmus nevyužíva maticu C_0, ktorá zodpovedá účelovej funkcii v úlohe (40). Dokonca sa dá ukázať, že kedže vstup do algoritmu je optimálne riešenie úlohy SDP (40), potom hodnota $Tr(C_0 X_0)$ zostáva konštantná počas celého procesu.

A.3 Niektoré Vety a Tvrdenia využité v práci

Veta A.3.1. Veta o Schurovom doplnku. [10, str 175] Daná je bloková symetrická matica $M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$ s regulárnou podmaticou A. Potom platí:

- M je kladne definitná $\Leftrightarrow$ matica A a matica $C - B^T A^{-1} B$ sú kladne definitné.
- M je kladne semidefinitná $\Leftrightarrow$ matica A a matica $C - B^T A^{-1} B$ sú kladne semidefinitné.

Matica $C - B^T A^{-1} B$ sa nazýva Schurov doplnok.

Dôkaz.

Náčrt dôkazu pochádza z [10]. Nech matica U je daná ako:

$$U = \begin{pmatrix} I & 0 \\ -B^T A^{-1} & I \end{pmatrix}.$$

Znásobením blokových matíc U, M, U^T dostávame nasledujúcu rovnosť:

$$\begin{pmatrix} I & 0 \\ -B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{pmatrix}.$$

Determinat matíc U, U^T je jedna. Potom dostávame:

$$det(M) = det(A) det(C - B^T A^{-1} B).$$

Kedže M je kladne semidefinitná, jej determinant je nezáporný, rovnako ako aj determinant jej podmatice A. Z toho dostávame, že aj $\det(C - B^T A^{-1} B) \geq 0$. Teda matice A a $C - B^T A^{-1} B$ majú nezáporný determinat a teda musia byť kladne semidefinitné. Opačná implikácia je zrejmá.

Pri dôkaze pre kladnú definitnosť sa postupuje analogicky. □

Lema A.3.2. [10, str 175] Máme danú maticu $M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix}$. Potom pre regulárnu maticu A platia nasledujúce rovnosti:

$$rank \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = rank \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} = rank(A) + rank(C - B^T A^{-1} B)$$

Dôkaz.

$$M = \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ -B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \begin{pmatrix} I & -A^{-1} B \\ 0 & I \end{pmatrix} = \begin{pmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{pmatrix}.$$

Po prenásobení sme dostali blokovo diagonálnu maticu, teda jej hodnosť je súčet hodností matíc na diagonále:

$$rank \begin{bmatrix} A & 0 \\ 0 & C - B^T A^{-1} B \end{bmatrix} = rank(A) + rank(C - B^T A^{-1} B)$$

□

Príloha B

Zdrojový kód

Stokrát generovaná Ortogonálna Procrutova úloha s rotačnou maticou rozmeru 3×3. Riešenie pomocou algoritmu SOL-QMP aj alternatívnej semidefinitnej relaxácie

```
clear all
Vysledky=zeros(100,2);  %1.stlpec SOL-QMP, 2.stlpec ASR


for q = 1:100
   %Generovanie rotacnej matice X_presne
    fi = randi(360);
    psi = randi(360);
    rho = randi(360);
    R1 = [cosd(fi) sind(fi) 0; -sind(fi) cosd(fi) 0; 0 0 1];
    R2 = [1 0 0; 0 cosd(psi) sind(psi); 0 -sind(psi) cosd(psi)];
    R3 = [cosd(rho) 0 sind(rho); 0 1 0; -sind(rho) 0 cosd(rho)];
    X_presne = R1 * R2 * R3;
%% Definovanie rozmerov a vstupov
np = 1;         %np, rp rozmery matic vstupujucich do Procrut. ulohy
rp = 3;
n=rp;           %po prenasobeni A,B (vid M0) do tvaru QMP su tieto rozmery
r=rp;
A=randi(30,np,rp)   %nahodna matica A
B = A * X_presne;
c = trace(B'*B);
E1=zeros(r,r);
E2=zeros(r,r);
M0 = [A'*A -A'*B; -B'*A c/r*eye(r)];

%% 1.cast algoritmu SOL-QMP - najst optimalne U
cvxp = cvx_precision( 'high' );
cvx_begin SDP
    variable U(n+r,n+r) symmetric
    minimize( trace(M0*U))
    subject to
for i=1:n
    for j=1:r
```

```
        E1=zeros(r,r);
        E2=zeros(r,r);
        ES=zeros(r,r);
        E1(i,j) = 1;
        E2(j,i) = 1;
        M = [E1+E2 zeros(n,r); zeros(r,n) zeros(n,n)];
        N = [zeros(n,n) zeros(n,r); zeros(r,n) E1+E2];
        trace(M*U) == 2* eq(i,j)  %funkcia eq == Kronecker delta
        trace(N*U) == 2*eq(i,j)
     end
end
U >= 0;
cvx_end
U  ;            %optimalne symetricke U*

%% 3 cast, pocitanie rozkladu U
[V,D] =eig(full(U));
W = zeros(n+r,r);
j=1;
for i = 1:n+r
    d_i=D(i,i);             %i-ta vlastna hodnota
   if abs(d_i) > 1.e-11     %ak je vacsia ako tolerancia => vstupuje do alg.
    v_i=V(:,i);             %i-ty vlastny vektor
    w_i=v_i*sqrt(d_i);      %vektor w, vznikol prenasobenim vl. vektora + sqrt(vl. cisla)
    W(:,j) = w_i;           %vektor w sa vklada do matice W
    j=j+1;
   end
end
W;                          %W je matica rozkladu: U = W*W'

%% 4. cast algoritmu SOL-QMP, vypocitanie X_alg (= X*)
Y= W(1:n,:);
Z= W(n+1:n+r,:);
X_alg = Y*Z';       %Koniec algoritmu SOL-QMP, X_alg je najdena matica
presnost = norm(A*X_alg-B, 'fro');
Vysledky(q,1)=presnost;
%% Alternativna semidefinitna relaxacia
cvx_begin SDP
```

```
    variable X(3,3)
    minimize(c-2* trace(B'*A*X))
    subject to
       [eye(3) X' ; X eye(3)]>=0
cvx_end
presnost2 = norm(A*X-B, 'fro') ;
Vysledky(q,2)=presnost2;
end
Vysledky
```

Stokrát generovaná Ortogonálna Procrutova úloha s permutačnou maticou rozmeru 6×6, riešenie pomocou alternatívnej semidefinitnej relaxácie.

```
clear all
Vysledky=zeros(100,3);
for q = 1:100
   rp=6;
   X_presne=eye(rp);
   X_presne=X_presne(randperm(rp),:);

%% Definovanie rozmerov a vstupov
np = 1;     %np, rp rozmery matic vstupujucich do Procrut. ulohy
n=rp;        %po prenasobeni A,B (vid M0) do tvaru QMP su tieto rozmery
r=rp;
%A=randi(30,np,rp);    %vektor A, ked sa prvky mozu opakovat
A= randperm(30,rp);
B = A * X_presne;
c = trace(B'*B);

%Alternativna semidefinitna relaxacia
cvx_begin SDP
    variable X(rp,rp)
    minimize(c-2* trace(B'*A*X))
    subject to
       [eye(rp) X' ; X eye(rp)]>=0
       X*ones(n,1)==1;
       %X'*ones(n,1)==1;
       %X>=0;
cvx_end
```

```
%% Zistenie ci sa matice rovnaju po zaokruhleni
X_prac = X;
X_zaok = zeros(rp,rp);
for i=1:rp
Maxi = max(X_prac,[],'all');    %maximalny prvok matice
[row col] = find(X_prac == Maxi);
X_zaok(row, col) = 1;
X_prac(:,col)=0;
X_prac(row,:)=0;
end

if isequal(X_zaok,X_presne)
    Vysledky(q,3)=1;    %3. stlpec, ci sa matice X_zaok a X_presne rovnaju
end
presnostZ = norm(A*X_zaok-B, 'fro');
Vysledky(q,2)=presnostZ;     %2. stlpec, presnost s X_zaok
presnost = norm(A*X-B, 'fro');
Vysledky(q,1)=presnost;     %1. stlpec, presnost s X_alg
end
Vysledky
```

Stokrát generovaná Ortogonálna Procrutova úloha s permutačnou maticou rozmeru 4×4, riešenie pomocou algoritmu SOL-QMP.

```
clear all
Vysledky=zeros(100,5);

for q = 1:100
   rp=4;
   X_presne=eye(rp);
   X_presne=X_presne(randperm(rp),:);

   %% Definovanie rozmerov a vstupov
np = 1;     %np, rp rozmery matic vstupujucich do Procrut. ulohy
n=rp;        %po prenasobeni A,B (vid M0) do tvaru QMP su tieto rozmery
r=rp;
%A=randi(30,np,rp);     % prvky vektora A sa mozu opakovat
A= randperm(30,rp);
```

```
B = A * X_presne;
c = trace(B'*B);
M0 = [A'*A -A'*B; -B'*A c/r*eye(r)];
X_zaok = zeros(rp);
E1=zeros(r,r);
E2=zeros(r,r);
Bi=zeros(r,r);
Di=zeros(r,r);

% % 1.cast algoritmu SOL-QMP - najst optimalne U
cvx_begin SDP
    variable U(n+r,n+r) symmetric
    minimize( trace(M0*U))
    subject to
for i=1:n
    Bi=zeros(r,r);
    Di=zeros(r,r);
    Bi(:,i)=1;
    Di(i,:)=1;
    for j=1:n
        E1=zeros(n,r);
        E2=zeros(n,r);
        ES=zeros(n,r);
        E1(i,j) = 1;
        E2(j,i) = 1;
        M = [E1+E2 zeros(n,r); zeros(r,n) zeros(n,n)];
        M4 = [zeros(r,r) E1; E1' zeros(n,n)];
        N = [zeros(n,n) zeros(n,r); zeros(r,n) E1+E2];
        trace(M*U) == 2* eq(i,j)  %funkcia eq == Kronecker delta
        trace(N*U) == 2*eq(i,j)
        - trace(M4*U)<=0;
    end
    M2 = [zeros(r,r) Bi; Bi' zeros(n,n)];
    M3 = [zeros(r,r) Di; Di' zeros(n,n)];
    trace(M2*U)==1;
    trace(M3*U)==1;
end
U >= 0;
```

```
cvx_end
U;          %optimalne symetricke U*
%% 3. Rozklad matice U (vypocitanej v prvej casti)
% U = X_redd;
[V,D] =eig(full(U));
W = zeros(n+r,r);
j=1;
for i = 1:n+r
    d_i=D(i,i);              %i-ta vlastna hodnota
   if abs(d_i) > 1.e-07     %ak je vacsia ako tolerancia => vstupuje do alg.
    v_i=V(:,i);              %i-ty vlastny vektor
    w_i=v_i*sqrt(d_i);        %vektor w, vznikol prenasobenim vl. vektora + sqrt(vl. cisla)
    W(:,j) = w_i;           %vektor w sa vklada do matice W
    j=j+1;
   end
end
W;       %W je matica rozkladu: U = W*W'

% 4. cast algoritmu SOL-QMP, vypocitanie X_alg (= X*)
Y= W(1:n,:);
Z= W(n+1:n+r,:);
X_alg = Y*Z';

%% rovnaju sa po zaokruhleni zistenie
X_prac =X_alg;
X_zaok = zeros(rp,rp);
for i=1:rp
Maxi = max(X_prac,[],'all');    %maximalny prvok matice
[row col] = find(X_prac == Maxi);
X_zaok(row, col) = 1;
X_prac(:,col)=0;
X_prac(row,:)=0;
end

if isequal(X_zaok,X_presne)
    Vysledky(q,3)=1;    %3. stlpec, ci sa matice zaok a presne rovnaju
end
presnostZ = norm(A*X_zaok-B, 'fro');
```

```
Vysledky(q,2)=presnostZ;     %2. stlpec, presnost s X_zaok
presnost = norm(A*X_alg-B, 'fro');
Vysledky(q,1)=presnost;       %1. stlpec, presnost s X_alg
Vysledky(q,4)=rank(X_alg);
Vysledky(q,5)=rank(full(U),1e-8);
end
Vysledky
```