VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY

INSTITUTE OF MATHEMATICS

KVADRATICKÝ PŘIŘAZOVACÍ PROBLÉM – REFORMULACE A DOLNÍ MEZE

QUADRATIC ASSIGNMENT PROBLEM – REFORMULATIONS AND LOWER BOUNDS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **ONDŘEJ LIŠKA**

AUTHOR

VEDOUCÍ PRÁCE — **RNDr. PAVEL POPELA, Ph.D.**

SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Student:	**Ondřej Liška**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**RNDr. Pavel Popela, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Kvadratický přiřazovací problém – reformulace a dolní meze

Stručná charakteristika problematiky úkolu:

Kvadratický přiřazovací problém patří mezi fundamentální problémy kombinatorické optimalizace s mnoha inženýrskými aplikacemi. Jelikož nalezení optimálního řešení tohoto problému je velmi výpočetně náročné i pro relativně malé instance, bylo vyvinuto několik metod pro určení dolní meze hodnoty optimálního řešení. Cílem práce je seznámit se kvadratickým přiřazovacím problémem, jeho různými formulacemi a metodami pro určení dolních mezí. Dále bude provedena implementace a srovnání zvolených metod ve vhodném programovacím jazyku (Julia, Matlab, Python,...).

Téma bude konzultováno a řešeno ve spolupráci s Ing. J. Kůdelou (UAI).

Cíle bakalářské práce:

1. Seznámení se s kvadratickým přiřazovacím problémem a jeho různými formulacemi a aplikacemi.
2. Osvojení si metod pro určení dolních mezí.
3. Softwarová implementace a srovnání zvolených metod.

Seznam doporučené literatury:

CELA, Eranda. The Quadratic Assignment Problem. New York: Springer. ISBN 978-1-4419-4786-4.

ANSTREICHER, Kurt, M. Recent advances in the solution of quadratic assignment problems. Mathematical Programming, Series B. 2003, 97(1), 27-42.

ZHANG, Huizhen, BELTRAN-ROYO, Cesar a MA, Liang. Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. Annals of Operations Research. 2013, 207(1), 261–278.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Tato práce se zabývá kvadratickým přiřazovacím problémem. První část představuje přiřazovací problémy včetně základní aplikace. Po vymezení problematiky a základní konvence značení jsou popsány zvolené metody řešení tohoto problému. V třetí části jsou srovnány metody řešení implementované v jazyku Julia s využitím řešiče Gurobi.

Summary
This bachelor thesis is focused on quadratic assignment problém. In first part are introduced assignment problems, include basic application. After description of problem and marking convention follows introducing of selected solution metods. In third part are compared solution metods, which are implemented i Julia language with solver Gurobi.

Klíčová slova
Optimalizace, kvadratický přiřazovací problém, reformulace, linearizace, dolní meze, jazyk Julia, Gurobi

Keywords
Optimization, quadratic assignment problem, reformulations, linearization, lower bounds, Julia Language, Gurobi

LIŠKA, O. *Kvadratický přiřazovací problém – reformulace a dolní meze*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2019. 40 s. Vedoucí bakalářské práce RNDr. Pavel POPELA, Ph.D.

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Kvadratický přiřazovací problém – reformulace a dolní meze“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

V Brně .....

.....

Ondřej Liška

Poděkování

Tímto bych rád poděkoval nejen vedoucímu mé bakalářské práce RNDr. Pavlu Popelovi Ph.D. za metodické rady k zpracovaní problematiky. Poděkování náleží především Ing. Jakubu Kůdelovi za cenné odborné rady a jeho čas při konzultacích. Dále bych chtěl poděkovat Mgr. Kateřině Havránkové Ph.D. za pomoc při korektuře práce. Děkuji také své rodině a přátelům za neustálou podporu.

.....

Ondřej Liška

Obsah

Úvod

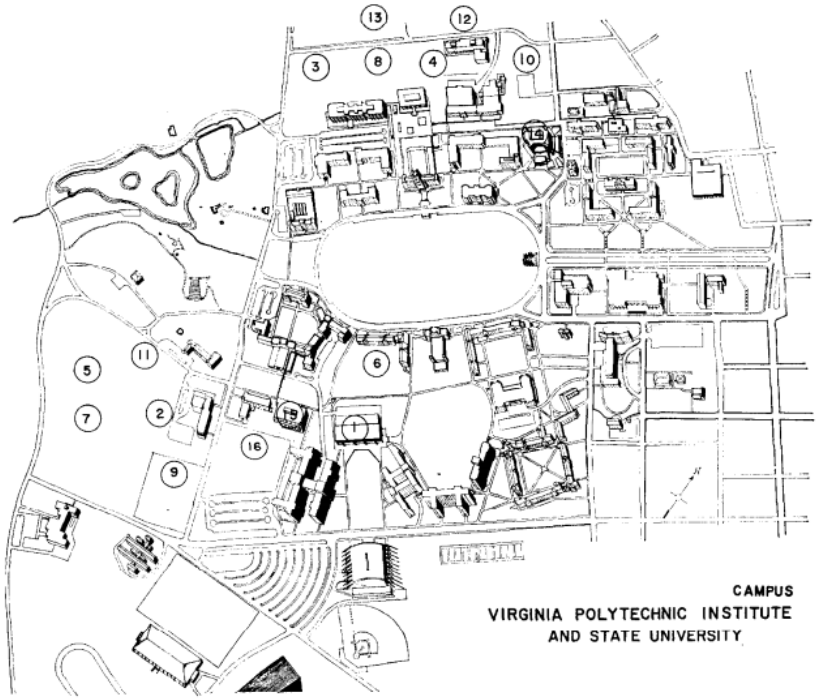
Jako kvadratické přiřazovací problémy je označena skupina specifických optimalizačních kombinatorických problémů. Jejich specifikum je s narůstající instancí výrazný nárůst obtížnosti problému. Jak bude vysvětleno v textu, tyto problémy spadají do kategorie NP-těžký. Především tato charakteristika přisuzuje tomuto tématu jeho přitažlivost. Navzdory desítkám let výzkumu neexistují algoritmy, jenž by dokázaly tyto problémy obecně spolehlivě řešit. K tomu, aby byl kvadratický přiřazovací problém i s dnešními metodami neřešitelný, stačí aby měl pouhých 30 prvků. Mnoho optimalizačních řešičů vůbec kvadratické přiřazovací problémy neumí řešit. Nejedná se o problém uměle vytvořený. Kvadratický přiřazovací problém je aplikovatelný při optimalizaci rozmanitých struktur a procesů. Z ekonomického pohledu velmi zajímavou aplikací je optimalizace rozmístění různých strojů v továrních halách z pohledu vzájemného toku součástek mezi stroji.

Motivace pro studium tohoto problému není malá a vidina možnosti posunout schopnosti současných algoritmů je lákavá. Dříve než lze pomýšlet na nové metody řešení, je potřeba porozumět současným metodám. Jak je obecně v matematice běžné, i v případě kvadratického přiřazovacího problému jsou první na řadě úvahy o převodu tohoto problému na jiný, již řešitelný problém. Další logickou volbou v případě, že problém se jeví jako neřešitelný, je odhad řešení. V této práci se dotkneme obou zmíněných možností. Závěrem srovnáme jednotlivé metody řešení pomocí implementované v jazyku Julia v kombinaci s řešičem Gurobi. Zvláště zajímavé bude srovnání dvou zmíněných možností při řešení kvadratického přiřazovacího problému „pouze“ s algoritmy Gurobi.

1 Formulace problému

1.1 Motivace

V hodinách matematiky na základní škole je nejoblíbenější otázkou pokládanou žáky „A k čemu je to dobré ?“. Mnoho lidí si zvyk pokládat tuto otázku, a především matematikům, uchová i v dospělosti. V žádném případě nelze žákům ani dospělým tento zvyk vyčítat, i kdyby byla otázka sebevíc provokativně formulovaná. Naopak je škoda, pokud nevíme na tuto otázku odpověď. Přestože naše práce je zaměřena pouze na metody řešení problému, je vhodné uvést tento problém s příkladem praktické aplikace. Díky uvedení následující motivační úlohy budeme schopni použité pojmy mnohem lépe ilustrovat. Při výběru motivační úlohy jsme se inspirovali slavným problémem rozmístění budov v kampusu Virginia Polytechnic Institute and State University [1].



Obrázek 1.1: Návrh na rozmístění budov v kampusu Virginia Polytechnic Institute and State University [1]

.

Motivační úloha

Představme si, že máme za úkol navrhnout rozmístění budov v plánovaném univerzitním kampusu. Máme dán počet budov daných funkcí, které mají být v kampusu postaveny. Počet volných stavebních parcel odpovídá počtu plánovaných budov. Naším úkolem je vybrat takové rozmístění budov, které je z pohledu zadaných kriterii nejvhodnější. Řešíme tedy úlohu ideálního uspořádání budov na stavební parcely. Z pohledu optimalizace se jedná o tzv. *přiřazovací problém* viz 1.2. K tomuto označení později přidáme další přívlastky podle zvolených kritérií.

Nejčastějším kritériem při optimalizaci je cena. Problém nastává při rozhodování o tom, která cena je zde důležitá. V návrhu lze zohlednit náklady na výstavby areálů, ale také náklady na provoz. Z pohledu pořizovací ceny můžeme uvažovat například, že stavební parcely mají obecně jinou úroveň inženýrských sítí, kvalitu podloží, povolnou maximální výšku výstavby a nebo jsou ovlivněny územním plánem města. Různé druhy budov mají rozdílné požadavky na kvality stavebních parcel. Například pro plavecký bazén lze očekávat požadavek na inženýrské sítě, ale pravděpodobně zde nebude problém s omezením výšky budovy. U plaveckého bazénu také nevadí rušné ulice v okolí, ale například u knihovny by to znamenalo další prodražení v důsledku požadavku na odhlučnění. Pokud na některé z parcel již stojí budova, lze uvažovat o její přestavbě na objekt našeho kampusu. Přestavba na učebny by mohla být relativně levné řešení, ale umístění sportovní haly do již existující budovy může být mnohem problematičtější. Pro každou budovu lze stavební parcely seřadit podle dodatečných nákladů pro její výstavbu. Naším cílem však není najít ideální stavební parcelu pro jednu budovu, ale najít ideální kombinaci budov a stavebních parcel. Chceme tedy najít přiřazení, které minimalizuje součet dodatečných nákladů pro postavení všech budov. Takto zadanou úlohu lze modelovat jako *lineární přiřazovací problém* viz 1.2.1.

Při plánování kampusu, můžeme také vzít v potaz, jak bude dané rozmístění budov výhodné z pohledu pohybu osob v areálu. Předpokládejme, že mezi různými budovami bude za den proudit rozdílné množství lidí. Vzdálenosti mezi stavebními parcelami se rovněž liší. Budeme tedy usilovat o to, aby dvojice budov, mezi nimiž očekáváme velký pohyb lidí, byly k sobě co nejblíž. Naopak vzdálenost mezi budovami, kde proudí menší množství lidí, může být velká. Naším výsledkem by mělo být rozmístění, které bude pro všechny budovy minimalizovat součet součinů vzdáleností budov s množstvím lidí, kteří mezi danými budovami za den projdou. Vidíme, jak nešťastný je v tomto případě slovní popis, ve kterém se snažíme o exaktní vyjádření. Zde se projevuje síla matematiky, kdy daný problém elegantně popisujeme jako tzv. *kvadratický přiřazovací problém* viz 1.3.

Namísto parametrů. které souvisely s jedním objektem, uvažujeme již při druhém pohledu parametry, které jsou určeny dvěma objekty. Jednoduše řečeno, dodatečné náklady jsou záležitostí jedné konkrétní budovy, ale vzájemná vzdálenost dává smysl jen pro dvojici budov. Takto lze vágně popsat rozdíl mezi lineárním a kvadratický přiřazovacím problémem. Nemá smysl zde uvádět další pojmy bez exaktní definice těchto základních. K našemu modelu se budeme nadále odkazovat.

1.2 Přiřazovací problém

Pod pojmem *přiřazovací problém* rozumíme v matematice otázku, jak přiřadit n objektů z jedné množiny k n objektům z jiné množiny. Konkrétní způsob, jakým jsou tato objekty přiřazeny, označujeme jako *přiřazení*. V matematice existuje více způsobů jak lze přiřazení vyjádřit. Přiřazení lze chápat jako bijekci φ mezi dvěma konečnými množinami U a V o n prvcích. Pro reprezentaci je praktické dívat se na přiřazení jako na permutaci. Poté, co prvky v množinách U a V doplníme indexy, je možné $\varphi : U \to V$ zapsat takto

$$\begin{pmatrix} 1 & 2 & \dots & n \\ \varphi(1) & \varphi(2) & \dots & \varphi(n) \end{pmatrix}$$

Pokud tedy například $\varphi(4) = 6$ znamená to, že čtvrtý prvek z množiny U je přiřazen k šestému prvku z množiny V. Každou permutaci lze zapsat pomocí permutační matice.

Definice 1.1. [2] *Nechť $X = (x_{ij})$ je matice $n \times n$. Matici X nazveme permutační matice, pokud platí následující podmínky*

$$\begin{aligned} &\sum_{i=1}^{n} x_{ij} = 1, \quad 1 \leq j \leq n, \\ &\sum_{j=1}^{n} x_{ij} = 1, \quad 1 \leq i \leq n, \\ &x_{ij} \in \{0, 1\}, \quad 1 \leq i, j \leq n. \end{aligned} \tag{1.1}$$

Množinu všech $n \times n$ permutačních matic označíme symbolem $\prod_n$.

Věta 1.2. [3] *Ke každé permutaci φ na množině $\{1, 2, \ldots, n\}$ existuje jedna a právě jedna $n \times n$ permutační matice X, taková že platí*

$$\begin{aligned} x_{ij} = 1 \Leftrightarrow \varphi(i) = j, \\ x_{ij} = 0 \Leftrightarrow \varphi(i) \neq j. \end{aligned} \tag{1.2}$$

Říkáme, že matice X odpovídá permutaci φ.

Přiřazovacím problémem rozumíme obecně otázku nalezení permutace splňující jisté podmínky. Obecně tak můžeme vzít takřka libovolné podmínky. Například v naší motivační úloze bychom mohli požadovat, aby budovy byly od západu k východu seřazené podle velikosti. Z matematického pohledu se snažíme rozdělit podmínky na takzvaná kritéria, která rozhodují o vhodnosti řešení a na podmínky omezující výběr řešení. Omezující podmínky budeme označovat slovem *omezení*. Matematický zápis kritérií do výrazu označíme jako *účelovou funkci*[1]. V naší motivační úloze jsme nejprve hledali přiřazení, které bude minimalizovat celkovou cenu. Účelová funkce pro tuto úlohu by mohla vypadat následovně

$$\min_{\varphi \in S_n} Celkovacena(\varphi, U, V). \tag{1.3}$$

Kde S_n je množina všech permutací $\{1, 2, \ldots, n\}$ a *Celkovacena()* je funkce závislá na přiřazení a na množinách U, V, mezi kterými je přiřazení realizováno. Takto obecné vyjádření celkové ceny není praktické a proto vyjádříme celkovou cenu jako součet dílčích cen za postavení jednotlivých budov.

1.2.1 Lineární přiřazovací problém

Definice 1.3. [2] *Nechť $C = (c_{ij})$ je matice $n \times n$. Matici C nazveme matice nákladů. Hodnota c_{ij} jsou náklady na přiřazení i-tého prvku z U k j-tému prvku z V. Označme S_n jako množinu všech možných permutací $\{1, 2, \ldots, n\}$, pak řešením Lineárního přiřazovacího problému rozumíme*

$$\min_{\varphi \in S_n} \sum_{i=1}^{n} c_{i\varphi(i)}. \tag{1.4}$$

V případě naší motivační úlohy rozumíme U množinu všech budov a V množinu všech stavebních parcel. Dále c_{ij} jsou náklady na postavení i-té budovy na j-tou stavební parcelu.

[1] Více o účelové funkci a základech optimalizace například v [4]

1.3 Kvadratický přiřazovací problém

Na rozdíl od lineárního přiřazovacího problému, v případě kvadratickém řešíme minimalizaci hodnoty, která je ovlivněna parametry náležícími dvojici prvků z jedné množiny. Tato dvojice je pak obvykle přiřazována k dvojici z druhé množiny. Jednoznačné určení přiřazení dané dvojice k dvojici z jiné množiny je zajištěno dvěma přiřazeními. V naší motivační úloze odpovídá kvadratický přiřazovací problém rozmístění budov s ohledem na množství lidí, kteří budou mezi danými budovami chodit. Vybereme libovolné přiřazení, a budeme chtít určit, jak velkou vzdálenost urazí lidé z menzy do knihovny. Nestačí nám pouze určit, na kterou stavební parcelu přiřazuje dané přiřazení menzu, ale musíme také zjistit, kam přiřadí toto přiřazení knihovnu a teprve poté můžeme dohledat vzdálenost těchto míst. Zdůvodnění volby slova kvadratický bude patrné z (1.8).

Definice 1.4. [2, 3] *Nechť* $A = (a_{ik}), B = (b_{jl}), C = (c_{ij})$ *jsou* $n \times n$ *matice. Nazveme* A, B *matice koeficientů. Kde* a_{ik} *je tok*[2] *mezi* i*-tým a* k*-tým prvkem* U*. Dále* b_{jl} *je vzdálenost od* j*-tého k* l*-tému prvku z* V*. Hodnota* c_{ij} *jsou náklady na přiřazení* i*-tého prvku z* U *k* j*--tému prvku z* V*. Označme* S_n *jako množinu všech permutací* $\{1, 2, \ldots, n\}$*, pak řešením zobecněného kvadratického přiřazovacího problému rozumíme*

$$\min_{\varphi \in S^n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\varphi(i)\varphi(k)} + \sum_{i=1}^{n} c_{i\varphi(i)}. \tag{1.5}$$

Pokud $(c_{ij}) = 0, \forall i, j$ *pak mluvíme pouze o kvadratickém přiřazovacím problému. Dále výraz*

$$\min_{\varphi \in S^n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\varphi(i)\varphi(k)}, \tag{1.6}$$

budeme zkráceně označovat jako QAP(A,B)[3]*. Dále nechť*

$$Z(A, B, \varphi) = \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\varphi(i)\varphi(k)}, \tag{1.7}$$

pak funkci $Z(A, B, \varphi)$ *nazveme účelová funkce QAP(A,B).*

Definici zobecněného kvadratického problému jsme uvedli pouze pro úplnost, ale v našem textu se nadále budeme zabývat pouze QAP(A,B).

Instance problému Velikost problému je určena velikostí vstupních matic koeficientů. Všechny vstupní matice koeficientů musí být čtvercové a pro daný problém stejně velké. Velikost matice koeficientů je značena n. Hodnotu n budeme nadále označovat jako *instance problému* nebo jako *velikost problému*.

Symetrický kvadratický přiřazovací problém [2] Při řešení problému hraje důležitou roli jeho symetrie. Jako *symetrický* označíme QAP(A,B) tehdy, pokud alespoň jedna z jeho matic koeficientů A, B je symetrická. V opačném případě označíme problém jako *nesymetrický*.

[2] V našem motivačním problému rozumíme a_{ik} množství lidí, jenž za sledované období projde z i-té do j-té budovy. Podobně b_{jl} je vzdálenost od j-té k l-té stavební parcele

[3] Zkratka QAP(A,B) pochází z anglického označení Quadratic Assignment Problem.

1.3.1 Ekvivalentní formulace

Formulace 1.6 nejlépe vystihuje strukturu problému, což je hlavním důvodem, proč je využívána nejčastěji [2]. Další formulace získávají na hodnotě v případě, že chceme pro řešení využít například celočíselné programování.

Věta 1.5. [3] *Nechť X odpovídá permutaci φ, viz (1.2). QAP(A,B) odpovídá následujícímu minimalizačnímu problému*

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}b_{jl}x_{ij}x_{kl}, \\ s.t. \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n, \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n, \\ & x_{ij} \in \{0,1\}, \quad 1 \le i,j \le n. \end{aligned} \tag{1.8}$$

Takto definovanou úlohu budeme nazývat Koopmans-Beckmann formulace.

Další možností jak formulovat Kvadratický přiřazovací problém je využití stopy matice. Připomeňme nejprve jak je stopa matice definována

Definice 1.6. [2, 3] *Pro každou $n \times n$ matici M platí.*

$$tr(M) = \sum_{i=1}^{n} m_{ii}. \tag{1.9}$$

Kde $tr(M)$ je funkce označovaná jako stopa.

Věta 1.7. [2][4] *Nechť X náleží do množiny $\prod_n$. QAP(A,B) odpovídá následujícímu minimalizačnímu problému*

$$\min_{x \in \prod_n} \operatorname{tr}\left(AXB^TX^T\right). \tag{1.10}$$

Takto definovanou úlohu budeme nazývat Trace[5] formulace.

[4]Oproti původnímu textu jsme definici trace formulace upravili tak, aby platila i pro nesymetrické úlohy.

[5]Z anglického označení pro stopu matice jako „trace“

2 Metody řešení

V předchozí kapitole jsme se krátce věnovali aplikaci Kvadratického přiřazovacího problému. Je však zřejmé, že se nespokojíme s pouhou matematickou formulací našeho problému, ale budeme požadovat také nalezení optimálního řešení.

Výpočetní složitost [5] Již v úvodu bylo zmíněno, že kvadratický přiřazovací problém je NP-těžký. Do třídy problémů NP řadíme takové problémy, u nichž je možno verifikovat řešení v polynomiálním čase. Jako NP-těžký označujeme problémy, které jsou minimálně tak těžké jako nejtěžší problém z třídy NP. Naopak jako P označíme každý problém, který je možno řešit v polynomiálním čase. Na rozdíl do kvadratického přiřazovacího problému je lineární přiřazovací problém třídy P. Lineární přiřazovací probléme je snadno řešitelný například pomocí Maďarské metody[1]. Řešení QAP(A,B) instance 30 může trvat desítky hodin nebo se nemusí podařit vůbec[2] . Ve srovnání kvadratickými, jsou lineární přiřazovací problémy řešitelné, i pokud dosahují velikosti v řádu tisíců prvků

2.1 Linearizace

Jednou skupinou metod řešení kvadratického přiřazovacího problému jsou metody označované jako reformulace. Reformulace spočívá v transformaci úlohy do podoby jiné úlohy, která bude mít v jistém smyslu podobné řešení. Označením podobné rozumíme, že bude existovat zpětná transformace, která řešení transformované úlohy převede na řešení úlohy původní. Jako nejjednodušší příklad reformulace můžeme uvést změnu úlohy zadané pomocí Koopmans-Beckmann formulace na trace formulaci. Cílem reformulace je převedení na takový tvar, který je při známých algoritmech řešení mnohem snáze řešitelný. Mezi reformulace patří i tvz. linearizace. Linearizací je myšlen převod úlohy kvadratické na úlohu lineární. Při linearizaci značně narůstá počet proměnných, ale i přes to bývá takto přeformulovaná úloha snáze řešitelná.

2.1.1 Frieze-Yadegar linearizace

Jako první si představíme z jistého pohledu nejnázornější linearizaci. Budeme vycházet z Koopmans-Beckmann formulace. Myšlenkou Frieze-Yadegar linearizace je nahrazení součinu dvojice prvků jedním prvkem. Nadefinujeme nové n^4 pole konstant D.

$$d_{ijkl} = a_{ik}b_{jl}, \tag{2.1}$$

a nové n^4 pole proměnných Y takto

$$y_{ijkl} = x_{ik}x_{jl}. \tag{2.2}$$

Pole konstant skutečně vyrobíme tak, jak je popsáno v (2.1). V případě pole proměnných Y již nemůžeme jednotlivé prvky položit součinu prvků z X, jak je popsáno

[1] Anglicky Hungarian metod, tato metoda je jednoduše vysvětlena například v [6],komplexnější popis metody v původním článku [7]

[2] Mezi doposud nevyřešené patří úloha Tai30a [8] která je řádu 30

v (2.2). Vyměnili bychom pouze kvadratický tvar účelové funkce za kvadratický tvar omezení. Z pohledu řešení by takováto úprava neposkytovala výhodu. Vztah (2.2) je pouze návodný, ale požadovaného propojení X a Y docílíme pomoci většího množství omezení.

Věta 2.1. [3] *Nechť pro n^4 pole D platí $d_{ijkl} = a_{ik}b_{jl}$, poté QAP(A,B) odpovídá následujícímu minimalizačnímu problému:*

$$\begin{aligned}
\min \quad & \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n}\sum_{l}^{n} d_{ijkl}y_{ijkl} \\
s.t. \quad & \sum_{i=1}^{n} x_{ij} = 1 && 1 \le j \le n, \\
& \sum_{j=1}^{n} x_{ij} = 1, && 1 \le i \le n, \\
& x_{ij} \in \{0,1\}, && 1 \le i,j \le n, \\
& \sum_{i=1}^{n} y_{ijkl} = x_{kl}, && 1 \le j,k,l \le n, \\
& \sum_{j=1}^{n} y_{ijkl} = x_{kl}, && 1 \le i,k,l \le n, \\
& \sum_{k=1}^{n} y_{ijkl} = x_{ij}, && 1 \le i,j,l \le n, \\
& \sum_{l=1}^{n} y_{ijkl} = x_{ij}, && 1 \le i,j,k \le n, \\
& 0 \le y_{ijkl} \le 1, && 1 \le i,j \le n.
\end{aligned} \tag{2.3}$$

Formulace (2.3) obsahuje n^4 reálných proměnných,n^2 binárních poměrných a n^4+4n^3+2n omezení plus omezení nezápornosti proměnné y_{ijkl}. Formulaci (2.3) budeme nazývat *Frieze-Yadegar* linearizace.

2.1.2 Adams-Johnson linearizace

Linearizace podle Adamse a Johnsona navazuje na Frieze-Yadegar. Pole koeficientů D je vytvořeno podle (2.1). Pole proměnných Y má i zde reprezentovat součin prvků z X, jak je popsáno v (2.2). S využitím Kroneckerova součinu[3] jsou odvozeny vztahy, jenž popisují strukturu Y s využitím menšího počtu omezení.

[3]Více o Kroneckerovu součinu v [3] (strana 213)

Věta 2.2. [3] *Nechť pro n^4 pole D platí $d_{ijkl} = a_{ik}b_{jl}$, poté QAP(A,B) odpovídá následujícímu minimalizačnímu problému:*

$$\begin{aligned}
\min \quad & \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n}\sum_{l}^{n} d_{ijkl}y_{ijkl} && , \\
s.t. \quad & \sum_{i=1}^{n} x_{ij} = 1, && 1 \le j \le n, \\
& \sum_{j=1}^{n} x_{ij} = 1, && 1 \le i \le n, \\
& \sum_{i=1}^{n} y_{ijkl} = x_{kl}, && 1 \le j,k,l \le n, \\
& \sum_{j=1}^{n} y_{ijkl} = x_{kl}, && 1 \le i,k,l \le n, \\
& x_{ij} \in \{0,1\}, && 1 \le i,j \le n, \\
& y_{ijkl} = y_{klij}, && 1 \le i,j,k,l \le n, \\
& 0 \le y_{ijkl}, && 1 \le i,j \le n.
\end{aligned} \tag{2.4}$$

Dále budeme v textu (2.7) nazývat *Adams-Johnson* linearizace. Adams-Johnson linearizace obsahuje n^4 reálných proměnných, n^2 binárních poměrných a $n^4 + 2n^3 + 2n$ omezení plus omezení nezápornosti proměnné y_{ijkl} [3]. Počet omezení v Adams-Johnson linearizaci je oproti Frieze-Yadegar o $2n^3$ menší. Na řešitelnosti má ještě výraznější vliv také přítomnost požadavku na celočíselnost proměnné x_{ij}. Řešení takto reformulovaného problému je však stále velmi obtížné.

2.1.3 Kaufman-Broeckx linearizace

Na rozdíl od předchozích, je Kaufman-Broeckx linearizace jednou z nejmenších linearizaci z pohledu počtu omezení. Výhodou této linearizace je také to, že funguje i pro zobecněný Kvadratický přiřazovací problém. V linearizaci vystupuje $n \times n$ matice proměnných Y. Pro prvky matice požadujeme splnění následující rovnosti:

$$y_{ij} = x_{ij}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}b_{jl}x_{kl}. \tag{2.5}$$

Věta 2.3. [2] *Pro y_{ij} definované podle (2.5) platí:*

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}b_{jl}x_{ij}x_{kl} = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}, \quad 1 \le i,j,k,l \le n. \tag{2.6}$$

Reformulace naznačená v (2.6) není vhodná z analogického důvodu jako v případě pole proměnných Y ve Frieze-Yadegar linearizaci. Použití (2.5) by znamenalo přítomnost nelineárních omezení. V Kaufman-Broeckx linearizaci vystupují omezení, jenž implikují splnění rovnosti (2.5) bez použití kvadratických omezení.

Věta 2.4. [2, 3] *QAP(A,B) odpovídá následující formulaci s n^2 reálnými proměnnými, n^2 celočíselnými proměnnými a $n^2 + 2n$ omezeními:*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}, && , \\
s.t. \quad & d_{ij}x_{ij} + \sum_{k=1}^{n}\sum_{l=1}^{n} a_{ik}b_{jl}x_{kl} - y_{ij} \leq d_{ij}, && 1 \leq i,j \leq n, \\
& 0 \leq y_{ijkl}, && 1 \leq i,j \leq n, \\
& \sum_{i=1}^{n} x_{ij} = 1, && 1 \leq j \leq n, \\
& \sum_{j=1}^{n} x_{ij} = 1, && 1 \leq i \leq n, \\
& x_{ij} \in \{0,1\}, && 1 \leq i,j \leq n,
\end{aligned} \tag{2.7}$$

kde $d_{ik} = \sum_{j=1}^{n}\sum_{l=1}^{n} a_{ij}b_{jl}$ *pro všechny* i,k.

Formulaci (2.7) budeme nazývat *Kaufman-Broeckx* linearizace. I přes řádově menší počet omezení v případě větších instancí, i nejsilnější metody řešení lineárních celočíselných problému nefungují uspokojivě. Obecně celočíselnost problému je největší překážkou při řešení. Na místě jsou úvahy, zda-li není možné požadavek na celočíselnost odstranit. Tyto úvahy dále rozvedeme v sekci 2.2.3

2.2 Meze

Jak jsme viděli v předchozí kapitole, tak metody řešení využívající linearizace operují s celočíselným programováním. Rychlost naznačených metod se i pro malé[4] instance problému může pohybovat v řádu hodin. V horším případě se můžeme potkat s problémem, jenž díky kombinaci velikosti a struktury je stále neřešitelný. Respektive i s nejmodernějšími algoritmy a výpočetní technikou nejsme schopni pro některé problémy najít optimální řešení. Zadání problémů, které jsou obecně známé, ale stále nebylo nalezeno jejich řešení, najdeme například v [8]. Naštěstí ne vždy potřebujeme najít optimální řešení, ale může nám stačit i „dostatečně dobré řešení". Budeme tedy požadovat nalezení odhadu řešení. Přirozený způsob odhadu řešení, je nalezení dolní a horní meze problému.

Horní a dolní mez Jako *horní mez* rozumíme hodnotu, jenž je s jistotou horší nebo rovna optimálnímu řešení[5]. Nejčastěji jako horní mez bereme nejlepší známé řešení problému. Pojmem *dolní mez* rozumíme takovou hodnotu, která je s jistotou lepší nebo rovna optimálnímu řešení. Pro dolní mez většinou neznáme řešení, při kterém je této hodnoty dosaženo. Navíc často bereme jako dolní mez hodnotu, která je lepší než řešení optimální, tudíž neexistuje přípustné řešení, kterým lze takové hodnoty dosáhnout. Dále v textu budeme jako *mez* označovat dolní mez.

[4]Problémy malé instance rozumíme $n \leq 25$

[5]Formulace "horší/lepší"je obecná, v případě QAP(A,B), kde účelovou funkci minimalizujeme, bereme jako lepší řešení to menší a jako horší to větší.

GAP V případě, že se hodnoty horní a dolní meze rovnají, znamená to, že jsme nalezli optimální řešení. Jako *GAP*[6] označíme hodnotu výrazu ((horní mez - dolní mez)/ horní mez). Na snaze postupného zmenšování GAP, tedy nacházení lepších horních a dolních mezi, je založena podstatná část optimalizačních algoritmů.

2.2.1 Gilmore-Lawler mez

Název Gilmore-Lawler mez je odvozen z historického vývoje. Gilmore formuloval roku 1962 tuto mez pro QAP(A,B), o rok později rozšířil Lawler tuto mez na zobecněný kvadraticky problém.

Hlavní myšlenka Základem při výpočtu Gilmore-Lawler meze je vytvoření lokálně optimálního řešení pro každou dvojici prvku z V přiřazeného do U. Lokální řešení vzhledem k jednomu přiřazenému prvku rozumíme přiřazení, které bere v potaz pouze vztahy s daným prvkem. Vztahy ostatních prvků mezi sebou nebereme při hledání lokálního optima v potaz. Například pro zvolené přiřazení a_i na b_j tedy hledáme takové přiřazení φ_L, které získáme řešením:

$$\min_{\varphi_L \in S^n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{j\varphi_L(k)}. \tag{2.8}$$

Lokální řešení obecně není řešením globálním, ale minimalizuje příspěvek dvojice a_i,b_j k hodnotě účelové funkce. Těmito minimálními příspěvky můžeme pro každou dvojici vytvořit novou matici nákladů na přiřazení. Řešením takto vytvořeného přiřazovacího problému bude hodnota, která je obecně lepší nebo rovna, než je hledané optimální řešení původního kvadratického přiřazovacího problému. Rovna bude pouze v případě, že přiřazení φ_L bude stejné pro všechny vybrané dvojice. Obecně tedy takovýto postup generuje dolní mez pro zadaný kvadratický přiřazovací problém.

Algoritmizace Takto popsaný postup je dobrý spíše pro představu o principech, na kterých metoda stojí. Nyní uvedeme několik definic, díky kterým následně popíšeme postup exaktněji.

Definice 2.5. [2] *Nechť U,V jsou vektory dimenze n. Pak skalární součin označíme $\langle U, V \rangle$ a definujeme:*

$$\langle U, V \rangle = \sum_{i=1}^{n} u_i v_i. \tag{2.9}$$

Definice 2.6. [2] *Nechť U, V jsou vektory dimenze n. Označme U^ϕ jako vektor U uspořádaný vzestupně podle velikosti. Vektor V uspořádaný sestupně podle velikosti označme V^ψ. Pak definujeme $\langle U, V \rangle^-$ a definujeme:*

$$\langle U, V \rangle^- = \langle U^\phi, V^\psi \rangle. \tag{2.10}$$

Přičemž $\langle U, V \rangle^-$ nazýváme minimální skalární součin.

[6] Z anglického výrazu *gap* který lze přeložit jako mezera.

[3] Nechť A,B jsou matice dimenze n vystupující ve QAP(A,B). Pro každé i označme $\hat{a}_i$ jako vektory dimenze $n-1$, který vytvoříme z prvků i-tého řádku matice A mimo prvek a_{ii}. Analogicky pro každé j označme $\hat{b}_j$ jako vektory dimenze $n-1$, který vytvoříme z prvků j-tého řádku matice B mimo prvek b_{jj}. S využitím předchozího značení nadefinujeme novou matici nákladů $L = l_{ij}$ následovně:

$$l_{ij} = c_{ij} + a_{ii}b_{jj} + \langle \hat{a}_i, \hat{b}_j \rangle^-. \tag{2.11}$$

Věta 2.7. [2] *Nechť L je matice $n \times n$ splňující 2.11 pro QAP(A,B). Pak platí:*

$$z = \min_{\varphi_L \in S^n} \sum_{j=1}^{n} l_{j\varphi_L(j)} \leq \min_{\varphi \in S^n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\varphi(i)\varphi(k)}. \tag{2.12}$$

Hodnotu z nazveme Gilmore-Lawler mez.

2.2.2 Meze pomocí vlastních čísel

Mezi hodnoty, které do jisté míry charakterizují matici, patří také vlastní čísla matice. Přirozeně se tedy nabízí myšlenka využít vlastní čísla matic koeficientů pro odhad řešení. Při odvození mezi využívajících vlastní čísla je využívána trace formulace.

Vlastní čísla

Definice 2.8. *Nechť A je $n \times n$ matice. Existuje číslo λ a vektor x dimenze n takový, že platí*

$$Ax = \lambda x, \tag{2.13}$$

pak λ nazveme vlastní číslo matice A a x vlastní vektor příslušný vlastnímu číslu λ.

Počet vlastních čísel matice je roven její dimenzi. Obecně tato vlastní čísla nemusí být různá. Pro výpočet vlastních čísel existuje mnoho numerických metod jejichž efektivita se liší podle struktury matice.

EV mez [2] Pro tuto metodu požadujeme, aby matice A,B byly symetrické. Díky symetrii matic bude zaručeno, že jejich vlastní čísla budou reálná. Nechť $\lambda_1, \lambda_2, \ldots, \lambda_n$ jsou vlastní čísla matice A a $\mu_1, \mu_2, \ldots, \mu_n$ jsou vlastní čísla matice B. Dále vytvoříme následující diagonální matice $\Lambda_1 = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ a $\Lambda_2 = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_n)$[7]. Dolní mez získaná metodou EV je rovna hodnotě výrazu $\langle \Lambda_1, \Lambda_2 \rangle^-$. Tato metoda je pouze základní ukázkou toho, jak lze vlastní čísla využít při výpočtu dolní meze. Existuje mnohem více metod založených na vlastních číslech, které jsou propracovanější a dávají výrazně lepší meze viz například [9].

[7] Diagonální matice je taková matice, která mimo svoji diagonálu obsahuje pouze nulové prvky. Nechť v je vektor dimenze n, pak označením diag(v) rozumíme čtvercovou matici řádu n, která má na diagonále složky vektoru v. Například pro $v = [1, 3, 5]$:

$$diag(v) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{pmatrix} \tag{2.14}$$

2.2.3 Relaxace

Vynechání podmínek celočíselnosti v zadání celočíselného optimalizačního problému nazýváme *relaxace* daného problému. V případě přiřazovacího problému nebude řešení relaxované úlohy dávat smysl. Přeci jen přiřazení například poloviny plaveckého bazénu na jedno místo a druhé poloviny na místo jiné smysl příliš nedává. Takové řešení však bude mít hodnotu zřejmě lepší anebo stejnou jako má optimální řešení původní úlohy[8]. Relaxací libovolné z celočíselných linearizací získáváme metodu pro výpočet dolní meze. Řešení relaxované úlohy je řádově rychlejší než u původní celočíselné formulace.

[8] Je zřejmé, že všechna celočíselná řešení jsou podmnožinou řešení s reálnými čísly, tedy i optimální řešení je obsaženo v množině přípustných řešení, ze kterých vybíráme optimum relaxované úlohy.

3 Srovnání metod

V této kapitole se budeme věnovat rozboru řešení dosažených jednotlivými metodami. Jednotlivé metody budeme porovnávat především v kontextu jiných metod námi spočítaných. Metodika srovnání metod je inspirována [10].

3.1 Softwarová implementace

Implementace metod popsaných v předchozí kapitole byla provedena v jazyku Julia[1]. Hlavní náplní bylo především přeformulování úlohy a následné řešení pomocí řešiče Gurobi[2]. Pro sestavení modelu v jazyku Julia jsme použili balíček JuMP.jl[3], který je určen pro optimalizaci obecně. Pro komunikaci s řešičem Gurobi jsme využili balíček Gurobi.jl[4]. Dále jsme také využili balíček LinearAlgebra[5], který obsahuje funkce lineární algebry jako například skalární součin a nebo funkci na určení vlastních čísel matice. Zdrojové kódy implementovaných funkcí a spouštěcí skript jsou obsaženy v příloze k elektronické verzi práce.

Julia

Pro krátké představení jazyka Julia uvádíme část článku Pavla Tišnovského. [11]*Jedná se o programovací jazyk, který byl navržen takovým způsobem, aby dokázal nahradit takové nástroje, jakými jsou Matlab, R, GNU Octave či Python s knihovnami Numpy a SciPy a současně dokázal překládat programy do optimalizovaného kódu, který by mohl konkurovat Céčku či (dokonce) Fortranu. Důvod je jednoduchý – tvůrci jazyka Julia používali pro svou práci (numerická matematika a statistika) různé nástroje, z nichž každý byl specializován pro určitou oblast (Matlab pro výpočty s maticemi), R pro statistické výpočty apod.), ovšem kooperace mezi těmito nástroji nebyla ideální. Obecné jazyky typu Ruby či Python byly zase (opět podle mínění autorů jazyka Julia) pomalé, zejména při porovnání s klasickým céčkem, ale i s Javou.*

[11]*Snahou původního autora Stefana Karpinského bylo vytvořit nový programovací jazyk založený na ve své době nejnovějších technologiích typu LLVM a současně používající to nejlepší ze světů imperativních i funkcionálních programovacích jazyků. Výsledkem měl být jazyk, který by byl snadno použitelný (zejména pro amatérské programátory), robustní, škálovatelný a současně i dostatečně rychlý, aby mohl soutěžit s C či Fortranem. Navíc měl jazyk Julia nabízet možnost snadné kooperace s knihovnami vytvořenými právě v C či Fortranu.*

Gurobi

Pro řešení úloh jsme použili akademickou licenci, která je z pohledu počtu proměnných neomezená. Volbu metod jsme ponechali v Pro srovnání jsme všechny úlohy řešili také

[1] JuliaPro 1.0.3.1 viz https://juliacomputing.com/products/juliapro
[2] Gurobi 8.1.1 viz http://www.gurobi.com
[3] JuMP.jl v0.18.5 viz http://www.juliaopt.org/JuMP.jl/v0.18/
[4] Gurobi.jl v0.5.9 viz https://github.com/JuliaOpt/Gurobi.jl
[5] viz https://github.com/JuliaStdlibs/LinearAlgebra.jl

v původní kvadratické formulaci pomocí zmíněného řešiče gurobi. Řešení provedené bez jakékoliv reformulace jsme označili jako *GUR*.

Hardware

Všechny prezentované výsledky byly dosaženy na stolním počítači s procesorem Intel(R) Core(TM) i3-4160 CPU @ 3.60GHZ a velikostí RAM 12GB (1600MHz). Po dobu výpočtů byl počítač odpojen ze sítě a nebyly prováděny jiné operace.

3.2 Řešené problémy

Pro testování s srovnání implementovaných metod jsme zvolil asi nejznámější databázi kvadratických přiřazovacích problémů zvanou QAPLIB[6][8]. Mezi zde uvedenými úlohami jsme vybrali několik skupin problémů rozdělených podle autorů. První tři písmena v názvu problému odkazují na autory, kteří daný problém uvedli. Dále následuje číslo označující instanci problému. Poslední písmeno rozlišuje jednotlivá zadání v případě, že daný autor uvedl více problémů stejné velikosti.

Tabulka 3.1: Autoři zadání

zkratka	autor/autoři [8]
els19	A.N. Elshafei
chrXX	N. Christofides a E. Benavent
nugXX	C.E. Nugent, T.E. Vollmann a J. Ruml
scrXX	M. Scriabin a R.C. Vergin
taiXX	E.D. Taillard
hadXX	S.W. Hadley, F. Rendl a H. Wolkowicz

Většina zadání má specifickou strukturu matic koeficientů, která až na výjimky je v rámci skupiny problémů podobná. Očekáváme jinou efektivitu metod v závislosti na struktuře problému. Při vyhodnocení metod rozdělíme řešené problémy do skupin podle autorů jak je naznačeno v tabulce 3.1.

3.2.1 Linearizace

Jako první uvedeme řešení problémů pomocí linearizace úlohy danými metodami[7]. Pro čas řešení byl nastaven limit 1000 sekund. Hodnoty GAP byly zaokrouhleny dolů, aby se podtrhl rozdíl mezi dosažením malého a žádného pokroku. Připomeňme, že GAP je označení hodnoty výrazu ((horní mez - dolní mez)/ horní mez).

[6]Označení QAPLIB je zkratkou anglického: A Quadratic Assignment Problem Library

[7]GUR–Koopmans-Beckmann formulace bez reformulace řešena pomocí Gurobi, KBL–Kaufman-Broeckx linearizace, FYL–Frieze-Yadegar linearizace , AJL–Adams-Johnson linearizace,

Tabulka 3.2: Linearizace

	čas[s]				GAP[-]			
	GUR	KBL	FYL	AJL	GUR	KBL	FYL	AJL
els19	60.91	1000	1000	1000	0	0.25	1	1
chr12a	1.86	8.67	3.6	3.44	0	0	0	0
chr12b	0.83	1.28	4.36	5.6	0	0	0	0
chr12c	18.32	59.46	6.18	4.59	0	0	0	0
chr15a	43.38	173	391	275	0	0	0	0
chr15b	3.41	59.2	112	120	0	0	0	0
chr15c	577	379.8	23.3	12.94	0	0	0	0
chr18a	430.55	1000	1000	1000	0	0.53	1	1
chr18b	1000	5.23	1000	1000	0.67	0	1	0.47
chr20b	1000	1000	1000	1000	0.84	0.007	1	1
chr22a	1000	1000	1000	1000	0.07	0.42	1	1
chr25a	1000	1000	1000	1000	0.61	0.56	1	1
ϕ	461	426	504	493	0.2	0.13	0.45	0.4
nug12	940	313	1000	1000	0	0	0.09	0.08
nug14	1000	1000	1000	1000	0.46	0.36	0.08	0.08
nug15	1000	1000	1000	1000	0.69	0.33	1	1
nug16a	1000	1000	1000	1000	0.67	0.35	1	1
nug16b	1000	1000	1000	1000	0.67	0.35	1	1
nug17	1000	1000	1000	1000	0.81	0.63	1	1
nug18	1000	1000	1000	1000	0.92	0.83	1	1
nug20	1000	1000	1000	1000	0.93	0.91	1	1
nug21	1000	1000	1000	1000	0.93	0.92	1	1
nug22	1000	1000	1000	1000	0.96	0.93	1	1
nug24	1000	1000	1000	1000	0.99	0.94	1	1
nug25	1000	1000	1000	1000	0.99	0.96	1	1
nug27	1000	1000	1000	1000	0.99	0.97	1	1
nug28	1000	1000	1000	1000	0.98	0.97	1	1
nug30	1000	1000	1000	1000	0.99	0.97	1	1
ϕ	996	954	1000	1000	0.8	0.7	0.87	0.87
scr12	2.045	12.12	1000	820	0	0	0.01	0
scr15	36.5	181	1000	1000	0	0	0.05	1
scr20	1000	1000	1000	1000	0.36	0.36	1	1
ϕ	346	398	1000	940	0.12	0.12	0.35	0.66
tai12a	1000	1000	73.6	68.3	0.18	0.48	0	0
tai12b	435	360	1000	840	0	0	0.48	0
tai15a	1000	1000	1000	1000	0.75	0.86	0.14	0.1
tai17a	1000	1000	1000	1000	0.86	0.92	1	1
tai20a	1000	1000	1000	1000	0.95	0.99	1	1
tai20b	1000	1000	1000	1000	0.99	0.94	1	1
ϕ	776	766	724	701	0.53	0.60	0.51	0.44
had12	1000	1000	511.48	647.58	0.13	0.15	0	0
had14	1000	1000	1000	1000	0.5	0.41	0.02	0.02
had16	1000	1000	1000	1000	0.66	0.89	1	1
had18	1000	1000	1000	1000	0.87	0.94	1	1
had20	1000	1000	1000	1000	0.93	0.94	1	1
ϕ	1000	1000	902	930	0.62	0.67	0.6	0.6

Na první pohled vidíme, že pro výraznou část problému jsou všechny metody vzhledem k našemu časovému limitu příliš pomalé. Jako nejtěžší ze zvolených se jeví úlohy *nugXX* a *hadXX*. Naopak velmi dobrých výsledků bylo dosaženo u problému *chrXX* a *scrXX*. Při prozkoumání matic koeficientů těchto problémů pozorujeme souvislost mezi řídkostí matice a řešitelností problémů. Tuto souvislost ovšem nemůžeme generalizovat, jak dokládá srovnání tai12a s tai12b, kde druhá jmenovaná je výrazně řidší než tai12a. Řídkost jedné z matic koeficientů není jediným parametrem, který ovlivňuje rychlost řešení. Dle očekávání je v získaných datech zřejmá souvislost mezi velikostí problému a časem řešení. Pozorujeme že tato závislost není lineární. Toto odpovídá tvrzení, že kvadratický přiřazovací problém je NP-težký. Některé problémy výrazně porušují trend nárůstu výpočetního času s velikostí problému. Nejlepším příkladem je úloha chr18b řešená pomocí Kaufman-Broeckx linearizace za 5.23 sekundy, což je druhý nejrychlejší čas dosažený touto metodou.

Metoda Kaufman-Broeckx linearizace obecně dosahuje překvapivě dobrých výsledků. Pro podstatnou část problému je nejrychlejší anebo bylo pomoci ní dosaženo největšího pokroku v řešení. V případě Frieze-Yadegar a Adams-Johnson linearizací pozorujeme většinou dva scénáře. Pro jednodušší problémy je úloha za daný čas vyřešena nebo k jejímu řešení nezbývá mnoho. V opačném případě je dosažený pokrok takřka žádný, přesněji nedochází k zmenšení hodnoty GAP. I tyto linearizace jsou však schopny dosáhnout pro jisté úlohy nejlepšího výsledku. Vidíme, že lepšího výsledku než univerzální algoritmus od Gurobi dosahují u jiných úloh než Kaufman-Broeckx linearizace. Jak uvidíme později, není vhodné výsledky hodnotit bez kontextu metod odhadujících dolní mez problému.

3.2.2 Meze

Výsledky prezentované v tabulce 3.2 nás utvrzují v požadavku na rozumný odhad řešení v reálném čase. V následující tabulce uvádíme pouze prvních 10 problému a odhad jejich mezí danými metodami[8].

Tabulka 3.3: Meze

	čas [s]				Hodnota meze			
	EV	GLB	KBLr	AJLr	EV	GLB	KBLr	AJLr
els19	6.5E-4	5.0E-3	2.9E-2	216.22	-2639E+5	11971949	0	9806505
chr12a	4.2E-4	3.0E-3	2.3E-2	2.48	-135327	7245	0	3581
chr12b	3.2E-4	2E-3	3.0E-3	2.16	-136734	7146	0	5693.5
chr12c	3.3E-4	2.3E-2	3.0E-3	2.62	-127514	7976	0	2573.5
chr15a	5.1E-4	3.0E-3	2.1E-2	14.29	-190769	5625	0	1702.2
chr15b	4.4E-4	3.0E-3	4.0E-3	14.89	-196658	4653	0	3249.072
chr15c	4.5E-4	4.0E-3	4.0E-3	69.8	-186404	6165	0	1599.5
chr18a	8.7E-4	4.0E-3	7.0E-3	58.18	-241984	6779	0	4189.7
chr18b	6.7E-4	5.0E-3	1.2E-2	61.24	-10945	1534	0	536
chr20b	7.6E-4	2.4E-2	8.0E-3	401.14	-30995	2196	0	827

Z tabulky 3.3 je dostatečně vidět diametrální rozdíl mezi kvalitou zvolených mezí. Vzhledem k nezápornosti koeficientů v zadání problému musí být i optimální řešení kladné.

[8]GLB – Gilmore-Lawler mez, KBLr – Relaxace metody Kaufman-Broeckx linearizace, AJLr – Relaxace metody Adams-Johnson linearizace

U metody EV je zřejmá naprostá nepoužitelnost získané meze. Podobný soud si dovolíme i v případě relaxace Kaufman-Broeckx linearizace. Relativně rozumné hodnoty mezí obdržíme relaxací Adams-Johnson linearizace. Zřejmou nevýhodou této meze je doba jejího výpočtu, která je řádově podobná metodám hledajícím přesné řešení. Malá rychlost by se dala omluvit v případě kvality hodnot meze. Bohužel pro všechny zadané problémy byla tato hodnota horší než pro Gilmore-Lawler mez. Tato metoda je navíc řádově rychlejší, a proto je jediná ze zvolených metod, kterou lze označit jako smysluplnou. Pro lepší demonstraci použitelnosti této meze provedeme její srovnání s metodami pro přesné řešení.

Kvalita mezí v metodách řešení

Připomeňme, že původní Kaufman-Broeckx formulace, stejně jako formulace získané linearizací, jsou řešeny algoritmy řešiče Gurobi. Tyto algoritmy využívají zlepšování mezí nejen pro nalezení optimálního řešení, ale především pro dokázání optimality nalezeného řešení. V následující tabulce uvedeme nejlepší algoritmem dosažené dolní a horní meze pro jednotlivé metody. Dolní meze porovnáme s Gilmore-Lawler mezí a horní meze porovnáme s optimální hodnotou[9].

Tabulka 3.4: Meze v metodách přesného řešení

		horní mez					dolní mez			
	opt	GUR	KBL	FYL	AJL	GLB	GUR	KBL	FYL	AJL
chr12a	9552	9552	9552	9552	9552	7245	9552	9552	9552	9552
chr12b	9742	9742	9742	9742	9742	7146	9742	9742	9742	9742
chr12c	11156	11156	11156	11156	11156	7976	11156	11156	11156	11156
chr15a	9896	9896	9896	9896	9896	5625	9896	9896	9896	9896
chr15b	7990	7990	7990	7990	7990	4653	7990	7990	7990	7990
chr15c	9504	9504	9504	9504	9504	6165	9504	9504	9504	9504
chr18a	11098	11098	11496	68402	68402	6779	11098	5392	0	0
chr18b	1534	1534	1534	2926	2926	1534	498	1534	0	1534
chr20b	2298	2432	2298	8390	8390	2196	388	2283	0	0
chr22a	6156	6156	6156	13978	13978	5924	5710	3510	0	0
chr25a	3796	4062	4400	18804	18804	2765	1564	1904	0	0
nug12	578	578	578	586	578	493	578	578	531	526
nug14	1014	1030	1052	1014	1014	852	550	666	923	923
nug15	1150	1186	1168	1492	1492	963	364	777	0	0
nug16a	1610	1638	1654	2184	2184	1314	532	806	0	0
nug16b	1240	1240	1268	1676	1676	1022	406	814	0	0
nug17	1732	1734	1790	2334	2334	1388	318	662	0	0
nug18	1930	2000	1936	2602	2602	1554	158	313	0	0
nug20	2570	2826	2628	3444	3444	2057	194	231	0	0
nug21	2438	2468	2572	3474	3474	1833	158	187	0	0
nug22	3596	3686	3684	5030	5030	2483	112	223	0	0
nug24	3488	3706	3600	4622	4622	2676	20	197	0	0
nug25	3744	4222	3902	4838	4838	2869	42	145	0	0
nug27	5234	5322	5484	7392	7392	3701	38	137	0	0
nug28	5166	5458	5434	7010	7010	3786	88	128	0	0
nug30	6124	6624	6430	8060	8060	4539	14	137	0	0

pokračování na další stránce

[9]opt – optimální hodnota řešení podle[8] GLB – Gilmore-Lawler mez, GUR–Koopmans-Beckmann formulace bez reformulace řešena pomocí Gurobi, KBL–Kaufman-Broeckx linearizace, FYL–Frieze-Yadegar linearizace , AJL–Adams-Johnson linearizace.

Tabulka 3.4 – pokračování z předchozí stránky

		horní mez					dolní mez			
	opt	GUR	KBL	FYL	AJL	GLB	GUR	KBL	FYL	AJL
scr12	31410	31410	31410	31410	31410	27858	31410	31410	30988	31410
scr15	51140	51140	51140	52340	98416	44737	51140	51140	49265	0
scr20	110030	110676	111100	199318	199318	86766	70462	70458	0	0
tai12a	224416	230704	224416	224416	224416	195918	188712	116286	224416	224416
tai15a	388214	401382	407356	413366	393052	327501	99540	53432	352891	352917
tai17a	491812	506860	505358	629294	629294	412722	66780	40033	0	0
tai20a	703482	755124	744782	888940	888940	580674	31440	1084	0	0
had12	1652	1660	1654	1652	1652	1536	1436	1394	1652	1652
had14	2724	2744	2744	2724	2724	2492	1368	1610	2669	2668
had16	3720	3776	3734	4058	4058	3358	1248	405	0	0
had18	5358	5422	5432	6086	6086	4776	652	277	0	0
had20	6922	6952	6982	7690	7690	6166	456	378	0	0

Z tabulky 3.4 je patrné, že hlavním problémem při určování optimálního řešení není jeho nalezení, ale dokázání jeho optimality. Především v metodách Frieze-Yadegar a Adams-Johnson linearizací není horní mez příliš vzdálená optimální hodnotě, ale v případě dolní meze nebylo za 1000 sekund učiněno žádného pokroku. Podobně metody GUR a KBL mohou určit hodnotu, která je optimální, ale nemohou to potvrdit, neboť nemají požadovanou dolní mez. Bez dokázání optimality se může i optimální řešení zdát bezcenné. Podobně řešení, které je optimální hodnotě relativně blízko, může být označeno za nepoužitelné v důsledku špatné dolní meze.

Korekce GAP pomocí GLB V případě, že zkombinujeme horní mez linearizací či metody GUR s dolní mezí GLB, získáme mnohem lepší představu o kvalitě nalezeného řešení. V následující tabulce budeme rozlišovat hodnotu GAP, kterou budeme chápat jako hodnotu výrazu ((horní mez(metody) - dolní mez(metody))/ horní mez(metody)). Zavedeme novou hodnotu *gap* jako v jistém smyslu redukovanou hodnotu, neboli hodnotu kombinující nalezené řešení a dolní mez určenou GLB. Hodnota gap odpovídá výrazu ((horní mez(metody) - dolní mez(GLB))/ horní mez(metody)).

Tabulka 3.5: Modifikované hodnoty GAP

	GUR		KBL		FYL		AJL	
	GAP	gap	GAP	gap	GAP	gap	GAP	gap
els19	0	0.3	0.25	0.3	1	0.59	1	0.59
chr12a	0	0.24	0	0.24	0	0.24	0	0.24
chr12b	0	0.26	0	0.26	0	0.26	0	0.26
chr12c	0	0.28	0	0.28	0	0.28	0	0.28
chr15a	0	0.43	0	0.43	0	0.43	0	0.43
chr15b	0	0.41	0	0.41	0	0.41	0	0.41
chr15c	0	0.35	0	0.35	0	0.35	0	0.35
chr18a	0	0.38	0.53	0.41	1	0.9	1	0.9
chr18b	0.67	0	0	0	1	0.47	0.47	0.47

pokračování na další stránce

Tabulka 3.5 – pokračování z předchozí stránky

	GUR		KBL		FYL		AJL	
	GAP	gap	GAP	gap	GAP	gap	GAP	gap
chr20b	0.84	0.09	0	0.04	1	0.73	1	0.73
chr22a	0.07	0.03	0.42	0.03	1	0.57	1	0.57
chr25a	0.61	0.31	0.56	0.37	1	0.85	1	0.85
ϕ	0.19	0.25	0.13	0.25	0.45	0.49	0.4	0.49
nug12	0	0.14	0	0.14	0.09	0.15	0.08	0.14
nug14	0.46	0.17	0.36	0.19	0.08	0.15	0.08	0.15
nug15	0.69	0.18	0.33	0.17	1	0.35	1	0.35
nug16a	0.67	0.19	0.51	0.2	1	0.39	1	0.39
nug16b	0.67	0.17	0.35	0.19	1	0.39	1	0.39
nug17	0.81	0.19	0.63	0.22	1	0.4	1	0.4
nug18	0.92	0.22	0.83	0.19	1	0.4	1	0.4
nug20	0.93	0.27	0.91	0.21	1	0.4	1	0.4
nug21	0.93	0.25	0.92	0.28	1	0.47	1	0.47
nug22	0.96	0.32	0.93	0.32	1	0.5	1	0.5
nug24	0.99	0.27	0.94	0.25	1	0.42	1	0.42
nug25	0.99	0.32	0.96	0.26	1	0.4	1	0.4
nug27	0.99	0.3	0.97	0.32	1	0.49	1	0.49
nug28	0.98	0.3	0.97	0.3	1	0.45	1	0.45
nug30	0.99	0.31	0.97	0.29	1	0.43	1	0.43
ϕ	0.79	0.24	0.7	0.23	0.87	0.38	0.87	0.38
scr12	0	0.11	0	0.11	0.01	0.11	0	0.11
scr15	0	0.12	0	0.12	0.05	0.14	1	0.54
scr20	0.36	0.21	0.36	0.21	1	0.56	1	0.56
ϕ	0.12	0.14	0.12	0.14	0.35	0.27	0.66	0.4
tai12a	0.18	0.15	0.48	0.12	0	0.12	0	0.12
tai12b	0	0.75	0	0.75	0.48	0.77	0	0.75
tai15a	0.75	0.18	0.86	0.19	0.14	0.2	0.1	0.16
tai17a	0.86	0.18	0.92	0.18	1	0.34	1	0.34
tai20a	0.95	0.23	0.99	0.22	1	0.34	1	0.34
tai20b	0.99	0.88	0.94	0.88	1	0.96	1	0.96
ϕ	0.62	0.39	0.69	0.39	0.6	0.45	0.51	0.44
had12	0.13	0.07	0.15	0.07	0	0.07	0	0.07
had14	0.5	0.09	0.41	0.09	0.02	0.08	0.02	0.08
had16	0.66	0.11	0.89	0.1	1	0.17	1	0.17
had18	0.87	0.11	0.94	0.12	1	0.21	1	0.21
had20	0.93	0.11	0.94	0.11	1	0.19	1	0.19
ϕ	0.61	0.09	0.66	0.09	0.6	0.14	0.6	0.14

Jak se nyní ukázalo, u mnoha metod je ve stanovený čas nalezeno řešení, které je mnohem blíže optimu, než by se mohlo zdát podle hodnot GAP. V případě použití korigované hodnoty gap namísto původní GAP se tedy metody mohou jevit jako úspěšnější. Na druhou stranu není tato úprava vhodná při snaze odhadnout, kolik času by bylo třeba přidat k limitu řešení, aby daná metoda nalezla optimální řešení. Není možné obecně říci, jaká metody či kombinace je ideální. Dokonce omezíme-li se na jednu oblast problémů, tak je výběr metody obtížný a nelze jej provést bez zohlednění velikosti daného problému. Pro zpřehlednění výsledků našeho srovnání metod uvedeme tabulku doporučených metod pro dané skupiny problému.

Tabulka 3.6: Doporučené metody řešení

zkratka	Doporučené metody
els19	GUR
chrXX	Kaufman-Broeckx linearizace
nugXX	Kaufman-Broeckx linearizace
scrXX	GUR (připadně KBL)
taiXX	AJL (připadně GUR[10])
hadXX	AJL/FYL(připadně GUR)

Bereme-li v potaz pouze metody zmíněné v této práci, tato tabulka může být návodem pro první volbu při hledání ideální metody. Moderní metody zahrnují i další možnosti řešení, které v této práci neuvažujeme.

[10]V případě větších instancí se zdá pravděpodobnější že GUR dojde výsledku, oproti Adams-Johnson linearizaci, kde vidíme tendenci výraznějšího poklesu úspěšnosti při nárůstu velikosti problému.

Závěr

V teoretické části práce bylo popsáno požadované množství metod řešení kvadratického přiřazovacího problému. Jako neuspokojivý se jeví výběr metod pro odhad dolních mezi. Především v případě metod založených na vlastních číslech nevystihuje vybraná metoda potenciál této skupiny metod. Naproti tomu, výsledky linearizací jsou uspokojivé i přes použití méně efektivní formulace Adams-Johnson linearizace. Ve srovnání s metodou GUR nepoužívající reformulaci byla pro každou skupinu problémů nalezena alespoň jedna reformulační metoda, která byla pro daný problém efektivnější či srovnatelně efektivní. Z provedeného srovnání plyne, že při řešení kvadratického přiřazovacího problému je vhodné posoudit strukturu matic koeficientů. Podle charakterizace problému se následně stále vyplatí vybrat konkrétní metodu řešení a nepoužívat pouze obecný algoritmus od Gurobi. V případě, že nepožadujeme nalezení optimálního, ale pouze dostatečně dobrého řešení, se jeví jako dobrá volba kombinace linearizace a metody na odhad dolní meze. Právě neefektivní odhad dolních mezí je stále pozorovatelnou slabinou algoritmu na řešení kvadratického přiřazovacího problému od řešiče Gurobi. Například v úloze chr18b nezvládl Gurobi za 990 sekund ověřit optimalitu nalezeného řešení, kde shodou okolností je Gilmore-Lawler mez rovna optimálnímu řešení. Vzhledem k neznalosti algoritmu, který Gurobi využívá, není ale fér tento algoritmus kritizovat, neboť je pravděpodobně zatížen snahou o univerzálnost.

Literatura

[1] DICKEY, J.W. a J.W. HOPKINS. CAMPUS BUILDING ARRANGEMENT USING TOPAZ. Transportation Research [online]. Great Britani: Pergamon Press, 1972, 17 červen 1972, (6), 59–68 [cit. 2019-05-21]. Dostupné z: https://www.sciencedirect.com/science/article/pii/0041164772901116

[2] CELA, Eranda. The Quadratic Assignment Problem: Theory and Algorithms. Kluwer Academic Publishers, 1998. ISBN 978-1-4419-4786-4.

[3] BURKARD, Rainer, Mauro DELL'AMICO a Silvano MARTELLO. Assignment Problems. 1. Philadelphia: Society for Industrial and Applied Mathematics, 2008. ISBN 978-0-898716-63-4.

[4] FERGUSON, Thomas S. LINEAR PROGRAMMING: A Concise Introduction [online]. [cit. 2019-05-21]. Dostupné z: https://www.math.ucla.edu/ tom/LP.pdf

[5] KNUTH, D. E. Postscript about NP-hard problems. ACM SIGACT News [online]. New York USA: ACM SIGACT, 1974, duben 1974, 6(2), 15-16 [cit. 2019-05-22]. DOI: 10.1145/1008304.1008305. ISSN 01635700. Dostupné z: http://portal.acm.org/citation.cfm?doid=1008304.1008305

[6] The Hungarian algorithm. Hungarian algorithm [online]. 2013, 2013 [cit. 2019-05-22]. Dostupné z: http://www.hungarianalgorithm.com/hungarianalgorithm.php

[7] KUHN, H. W. The Hungarian method for the assignment problem. Naval Research Logistics [online]. Wiley Periodicals, 1955, Březen 1955, 2(1-2), 83-97 [cit. 2019-05-22]. ISSN 1520-6750. Dostupné z: https://web.eecs.umich.edu/ pettie/matching/Kuhn-hungarian-assignment.pdf

[8] BURKARD, Rainer, Eranda ÇELA, S.E. KARISCH , ANJOS, Miguel a Peter HAHN, ed. Problem Instances and Solutions. *QAPLIB - A Quadratic Assignment Problem Library* [online]. Philadelphia: University of Pennsylvania, 200 S. 33rd Street, Philadelphia, PA 19104, USA, 2002, 27 Květen 2018, Září 2017 [cit. 2019-05-17]. Dostupné z: http://anjos.mgi.polymtl.ca/qaplib/inst.html

[9] ANSTREICHER, Kurt, M. Recent advances in the solution of quadratic assignment problems. Mathematical Programming, Series B. 2003, 97(1), 27-42.

[10] ZHANG, Huizhen, Cesar BELTRAN-ROYO a Liang MA. Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. Annals of Operations Research [online]. Springer Science+Business Media, 2013, 7 February 2012, , 261–278 [cit. 2019-05-20].

[11] TIŠNOVSKÝ, Pavel. Programovací jazyk Julia: další stříbrná kulka v IT?. In: Root.cz [online]. Praha: Internet Info, 1998, 26. 5. 2016 [cit. 2019-05-20]. Dostupné z: https://www.root.cz/clanky/programovaci-jazyk-julia-dalsi-stribrna-kulka-v-it/

Seznam použitých zkratek a symbolů

φ	Přiřazení (reprezentované permutací)
X	Permutační matice
i, j, k, l	Sumační indexy
Π_n	Množina všech $n \times n$ permutačních matic
V, U	Množiny přiřazovaných prvků/Vektory
C	Matice nákladů
A	Matice koeficientů (matice toků)
B	Matice koeficientů (matice vzdáleností)
n	Velikost matic A, B, C, instance problému
S_n	Množina všech permutací řádu n
QAP(A,B)	Kvadratický přiřazovací problém s maticemi koeficientů A, B
Z(A,B,φ)	Hodnota účelové funkce QAP(A,B)pro přiřazení φ.
M	Obecná čtvercová matice řádu n
$tr(M)$	Stopa matice M
D	Pole/matice substitučních koeficientů
Y	Pole/matice substitučních proměnných při linearizaci
φ_L	Přiřazení použité při hledání lokálního optima
L	Matice lokálních řešení (matice nákladů)
λ	Vlastní číslo matice
GUR	Koopmans-Beckmann formulace (metoda řešení bez reformulace)
KBL	Kaufman-Broeckx linerizace
FYL	Frieze-Yadegar linerizace
AJL	Adams-Johnson linerizace
GLB	Gilmore-Lawler mez
KBLr	Relaxace Kaufman-Broeckx linerizace
AJLr	Relaxace Adams-Johnson linerizace
s.t.	Označení omezujících podmínek z anglického *subject to*