Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Studijní program: N1101
Studijní obor: 1101T016

DIPLOMOVÁ PRÁCE

Semidefinitní programování v kombinatorické optimalizaci

Autor: Ondřej Špaček
Vedoucí práce: Doc. Ing. Roman Čada, Ph.D.

Plzeň, 2020

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně s použitím odborné literatury uvedené v seznamu, který je uveden na konci této práce.

V Plzni dne ....
podpis

Poděkování

Především bych chtěl poděkovat svému vedoucímu diplomové práce Doc. Ing. Romanu Čadovi, Ph.D. za vstřícnost, spoustu času, který mi věnoval a cenné rady při řešení problémů spojených s diplomovou prací.

Abstrakt

Práce se zabývá využitím semidefinitního programování v kombinatorické optimalizaci. V první části je shrnuta teorie, která je potřebná k práci v této oblasti. V části druhé se zabýváme Shannonovou kapacitou grafu, Lovászovou ϑ funkcí a úlohou maximálního řezu. Zmíněné algoritmy jsou implementovány a testovány v programovacím jazyce Python 3.

Klíčová slova

kombinatorická optimalizace; semidefinitní programování; aproximační algoritmus; vektorové programování; Shannonova kapacita grafu; Lovászova ϑ funkce; MAX CUT; MAX k-CUT; kapacitní MAX k-CUT.

Abstract

Thesis deals with semidefinite programming in combinatorial optimization. The first part is summarizes the theory needed to work in this field. In the second section we are dealing with Shannon capacity of the graph, Lovasz ϑ function, and with MAX CUT problem. The mentioned algorithms are implemented and tested in the Python 3 programming language.

Keywords

combinatorial optimization; semidefinite programming; vector programming; approximation algorithm; Shannon capacity of the graph; Lovast ϑ function; MAX CUT; MAX k-CUT; capacited MAX k-CUT.

Obsah

Úvod

Matematické programování (optimalizace) se zabývá hledáním optimálních řešení matematických modelů. Modely, ve kterých se vyskytují pouze lineární funkce se zabývá lineární programování. Začátky lineárního programování jsou spojeny s druhou světovou válkou. George Dantzig, který měl na starosti vývoj logistických plánu na straně amerického letectva, v roce 1947 vymyslel Simplexovou metodu. S podobným konceptem přišel už dříve v roce 1939 Leonid Kantorovič, ale na jeho práci bohužel nikdo nenavázal. Další slavná jména, která stojí za zmínku v souvislosti s lineárním programováním jsou John von Neumann, Albert Tucker, Harold Kuhn a spousta dalších. Relativně nová oblast optimalizace se nazývá semidefinitní programování, kterou dobře charakterizuje název článku z roku 1981 s názvem *Linear Programming with Matrix Variables* od Cravena a Monda. Pokud máme optimalizační problém, ve kterém řešení jsou vyjádřena pomocí diskrétních proměnných, pak hovoříme o problému kombinatorické optimalizace. K řešení těchto problémů se v posledních cca 30 letech rozšířilo semidefinitní programování, které se využívá např. u Shannonovy kapacity grafu, studia řezů, problému obchodního cestujícího a dalších. Pro další historické informace související s optimalizací doporučuji zdroj [19], ze kterého bylo čerpáno.

Použité značení

$G = (V, E)$ – neorientovaný graf s množinou vrcholů V a množinou hran E

$\bar{G} = (V, \bar{E})$ – doplněk grafu G

$uv \in E$ – hrana mezi vrcholy u a v

$\alpha(G)$ – nezávislost grafu G

$\chi(G)$ – chromatické číslo grafu G

$\|x\| = \sqrt{x_1^2, \ldots, x_n^2}$ – norma vektoru x

$\langle x, y \rangle = x^T y = x_1 y_1 + \cdots + x_n y_n$ – skalární součin vektorů x a y v $\mathbb{R}^n$

Tr $A = \sum_i a_{ii}$ – stopa matice A

dom f – definiční obor funkce f

span $\{x_1, \ldots, x_n\}$ – lineární obal vektorů $x_1, \ldots, x_n$

rank A – hodnost matice A

$S_{n-1} = \{x \in \mathbb{R}^n \mid \|x\| = 1\}$ – jednotková sféra v $\mathbb{R}^n$

$A \otimes B$ – Kroneckerův součin matic A a B

Část I

Optimalizace

Kapitola 1

Základní geometrické pojmy

Kapitola je zpracována z [2] a [15].

1.1 Přímky a úsečky

Mějme dva body $x_1, x_2 \in \mathbb{R}^n$ takové, že $x_1 \neq x_2$ a parametr $\theta \in \mathbb{R}$. Potom výraz

$$y = \theta x_1 + (1 - \theta)x_2 \tag{1.1}$$

popisuje **přímku** procházející body x_1 a x_2. Pro $\theta = 0$ dostáváme bod x_2 a pro $\theta = 1$ bod x_1. Omezíme-li θ na interval $\langle 0, 1\rangle$, dostaneme **úsečku** s koncovými body x_1 a x_2. Výraz 1.1 lze přepsat do tvaru

$$y = x_2 + \theta(x_1 - x_2),$$

který můžeme interpretovat jako součet počátečního bodu x_2 a nějakého násobku směrového vektoru $x_1 - x_2$.

1.2 Afinní prostory

Říkáme, že $C \subseteq \mathbb{R}^n$ je **afinní prostor**, jestliže přímka procházející libovolnými dvěma různými body z C leží celá v C. Tedy C obsahuje lineární kombinace libovolných dvou bodů z C, jestliže součet koeficientů lineární kombinace je roven jedné. To lze zobecnit i pro více než dva body. Lineární kombinace $\theta_1 x_1 + \cdots + \theta_k x_k$ bodů $x_1, \ldots, x_k$ taková, že $\theta_1 + \cdots + \theta_k = 1$, se nazývá **afinní kombinace** bodů $x_1, \ldots, x_k$. Indukcí z definice afinního prostoru lze snadno ukázat, že pokud C je afinní množina, $x_1, \ldots, x_k \in C$ a $\theta_1 + \cdots + \theta_k = 1$, potom bod $\theta_1 x_1 + \cdots + \theta_k x_k \in C$.

Nechť C je afinní prostor a $x_0 \in C$, potom množina

$$V = C - x_0 = \{x - x_0 \mid c \in C\}$$

je **lineární vektorový prostor**, tj. množina, která je uzavřená na sčítání a násobení skalárem.
Afinní prostor C lze vyjádřit jako

$$C = V + x_0 = \{v + x_0 \mid v \in V\},$$

kde V je vektorový prostor a x_0 je počátek. Poznamenejme, že vektorový prostor V asociovaný s afinním prostorem C nezávisí na volbě počátku x_0.
Dimenze afinního prostoru $C = V + x_0$ je definována jako dimenze vektorového prostoru $V = C - x_0$, kde x_0 je libovolný prvek z C. Množina všech afinních kombinací bodů množiny $C \subseteq \mathbb{R}^n$ se nazývá **afinní obal** množiny C. Afinní obal množiny C budeme značit

$$\textbf{aff}\ C = \{\theta_1 x_1 + \cdots + \theta_k x_k \mid x_1, \ldots, x_k \in C, \theta_1 + \cdots + \theta_k = 1\}.$$

Afinní obal je nejmenší afinní prostor, který obsahuje množinu C. Tedy, jestliže S je afinní prostor takový, že $C \subseteq S$, potom $\textbf{aff}\ C \subseteq S$. Definujeme **relativní vnitřek** množiny C tak, že

$$\textbf{relint}\ \ \{x \in C \mid B(x, r) \cap \textbf{aff}\ C \subseteq C \text{ pro nějaké } r > 0\},$$

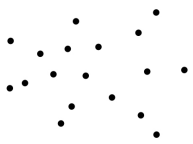
kde $B(x, r) = \{y \mid \|y - x\| \leq r\}$.

1.3 Konvexní množiny

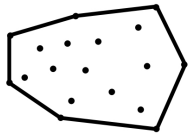
Říkáme, že množina C je **konvexní**, jestliže úsečka mezi libovolnými dvěma body z C leží také celá v C. Jinak řečeno, jestliže pro libovolné dva body $x_1, x_2 \in C$ a libovolné $\theta \in \langle 0, 1\rangle$ platí, že $\theta x_1 + (1-\theta)x_2 \in C$. Poznamenejme, že každý afinní prostor je zároveň konvexní množinou. Podobně jako afinní kombinaci definujeme **konvexní kombinaci** bodů $x_1, \ldots, x_k$ jako $\theta_1 x_1 + \cdots + \theta_k x_k$, kde $\theta_1 + \cdots + \theta_k = 1, \theta_i \geq 0$ pro $i = 1, \ldots, k$. **Konvexní obal** množiny C je množina všech konvexních kombinací bodů z množiny C, značíme

$$\textbf{conv}\ C = \{\theta_1 x_1 + \cdots + \theta_k x_k \mid x_i \in C, \theta_i \geq 0, i = 1, \ldots, k, \theta_1 + \cdots + \theta_k = 1\}.$$

Analogicky, konvexní obal množiny C je nejmenší konvexní množina, která obsahuje množinu C. Pro představu viz obrázek 1.1.



(a) Množina bodů C



(b) **conv** C

Obrázek 1.1: Konvexní obal množiny

1.4 Kužely

Množina C se nazývá **kužel**, jestliže pro každé $x \in C$ a $\theta \geq 0$ platí, že $\theta x \in C$. Je-li C navíc konvexní, pak se C nazývá **konvexní kužel**. Tedy C je konvexní kužel, jestliže pro libovolné $x_1, x_2 \in C$ a $\theta_1, \theta_2 \geq 0$ platí, že $\theta_1 x_1 + \theta_2 x_2 \in C$. Říkáme, že bod ve tvaru $\theta_1 x_1 + \cdots + \theta_k x_k$, kde $\theta_1, \ldots, \theta_k \geq 0$ je **kuželovou kombinací** bodů $x_1, \ldots, x_k$. Dále, pokud x_i leží v konvexním kuželu množiny C, potom libovolná kuželová kombinace bodu x_i leží rovněž v konvexním kuželu množiny C. Platí, že množina C je konvexní kužel právě tehdy, když C obsahuje všechny kuželové kombinace svých bodů. **Kuželový obal** množiny C je množina, která obsahuje všechny kuželové kombinace množiny C, tj.

$$\textbf{cone}\ C = \{\theta_1 x_1 + \cdots + \theta_k x_k \mid x_i \in C, \theta_i \geq 0, i = 1, \ldots, k\}\,.$$

Kuželový obal množiny C je zároveň nejmenší konvexní kužel, který obsahuje množinu C. Pro představu viz obrázek 1.2.

Kužel C nazýváme **bodový kužel**, jestliže

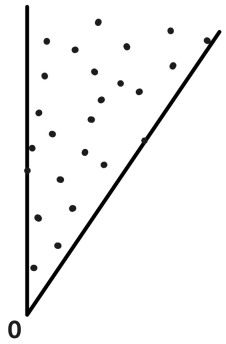
$$(x \in C \wedge -x \in C) \implies x = 0.$$

Duální kužel C^* ke kuželu C je množina

$$C^* = \{y \mid \langle x, y\rangle \geq 0 \text{ pro } x \in C\}\,.$$

Říkáme, že kužel C je **samoduální**, jestliže $C = C^*$.

(a) Množina bodů C



(b) **cone** C

Obrázek 1.2: Kuželový obal množiny

1.5 Nadroviny a poloprostory

Nadrovina je množina ve tvaru

$$\left\{x \mid a^T x = b\right\},$$

kde $a \in \mathbb{R}^n$, $a \neq 0$ a $b \in \mathbb{R}$. Analyticky na nadrovinu nahlížíme jako na množinu všech řešení lineární rovnice. Geometricky zase jako na množinu všech bodů takových, že mají konstantní skalární součin s normálovým vektorem a. Konstanta b značí posunutí nadroviny od počátku. Nadrovinu také

můžeme vyjádřit jako

$$\left\{x \mid a^T(x - x_0) = 0\right\} = x_0 + \left\{v \mid a^T v = 0\right\},$$

kde x_0 je libovolný bod této nadroviny a $\left\{v \mid a^T v = 0\right\}$ je množina všech vektorů, které jsou kolmé k normálovému vektoru a. Nadrovina je tedy množina, která obsahuje bod x_0 a libovolný bod ve tvaru $x_0 + v$, kde v je vektor, který je kolmý k normálovému vektoru a. Pro ilustraci v $\mathbb{R}^2$ viz obrázek 1.3a.

Nadrovina dělí $\mathbb{R}^n$ na dva poloprostory. Množina

$$\left\{x \mid a^T x \leq b\right\}, \text{ resp. } \left\{x \mid a^T x < b\right\},$$

kde $a \neq 0$ se nazývá (uzavřený) **poloprostor**, resp. **otevřený poloprostor**. Je to tedy množina všech řešení lineární nerovnice. Podobně jako nadrovinu, můžeme poloprostor vyjádřit ve tvaru

$$\left\{x \mid a^T(x - x_0) \leq 0\right\}, \text{ resp. } \left\{x \mid a^T(x - x_0) < 0\right\},$$

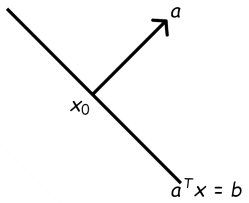
kde $a \neq 0$ a x_0 je libovolný bod z nadroviny $\left\{x \mid a^T x = b\right\}$. Poloprostor tedy obsahuje bod x_0 a libovolný bod $x_0 + v$, kde v je vektor, který s vnějším normálovým vektorem svírá tupý nebo pravý úhel. Tato interpretace je v $\mathbb{R}^2$ ilustrována na obrázku 1.3b. Ještě poznamenejme, že poloprostory jsou konvexní množiny, ale samozřejmě nejsou afinní.

1.6 Polyedry a polytopy

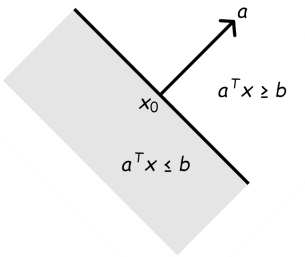
Polytopy jsou zobecněním konvexních mnohoúhelníků v rovině do více dimenzí. Polytop v $\mathbb{R}^3$ je konvexní množina, která je ohraničena konečně mnoha konvexními mnohoúhelníky (příkladem polytopů v $\mathbb{R}^3$ jsou např. Platónská tělesa). Na takovou množinu je možné nahlížet dvěma způsoby. **H-polyedr** je průnik konečně mnoha uzavřených poloprostorů v $\mathbb{R}^n$. **H-polytop** je omezený H-polyedr. **V-polytop** je konvexní obal konečně mnoha bodů v $\mathbb{R}^n$. Následující věta říká, že H-polytop a V-polytop jsou matematicky ekvivalentní množiny.

Věta. *[15] Každý V-polytop je H-polytop. Každý H-polytop je V-polytop.*

Poznamenejme, že V-polytop a H-polytop jsou sice ekvivalentní množiny, ale z algoritmického hlediska je velký rozdíl, zda pracujeme s bodovou množinou, nebo s uzavřenými poloprostory. Pro ilustraci: mějme lineární funkci, kterou chceme minimalizovat na daném polytopu. Pro V-polytop se jedná o triviální problém, protože stačí pro každý bod z množiny V určit hodnotu



(a) Nadrovina



(b) Poloprostor

Obrázek 1.3: Nadrovina a poloprostor v $\mathbb{R}^2$.

dané funkce a vybrat minimum. Na druhou stranu pro H-polytop se jedná o netriviální problém, kterým se zabývá lineární programování. Dále nebudeme rozlišovat mezi V-polytop a H-polytop a budeme mluvit jen o **polytopu**. A o H-polyedru budeme mluvit jen jako o **polyedru**.

Důležitý fakt, že každý polyedr je konečně generovaný, říká Minkowského-Weylova věta.

Věta (Minkowski-Weyl)**.** *[1] $P \subseteq \mathbb{R}^n$. Potom*

$$P = \boldsymbol{conv}(u_1, \ldots, u_r) + \boldsymbol{cone}(v_1, \ldots, v_s),$$

kde u_i, v_i jsou extremální vrcholy P právě tehdy, když $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$.

Kapitola 2

Konvexní optimalizace

Kapitola je zpracována z [2].

2.1 Obecná podmíněná úloha

$$\begin{aligned} &\inf\ f(x) \\ &g_i(x) \leq 0, i = 1, \ldots, m \\ &h_i(x) = 0, i = 1, \ldots, p \end{aligned} \tag{2.1}$$

Hledáme $x \in \mathbb{R}^n$, které minimalizuje $f(x)$ (pro maximalizaci minimalizujeme $-f(x)$), vzhledem k omezením $g_i(x)$ a $h_i(x)$. Proměnné x říkáme **optimalizační proměnná**, funkci $f(x)$ říkáme **cenová** nebo **účelová funkce**. Výrazy $g_i(x) \leq 0$ jsou **omezení typu nerovnosti** a $h_i(x) = 0$ jsou **omezení typu rovnosti**. Říkáme, že problém 2.1 je **úloha nepodmíněné optimalizace**, jestliže $m = p = 0$. Jinak se jedná o **úlohu podmíněné optimalizace**. **Definiční obor** $\mathcal{D}$ úlohy 2.1 je

$$\mathcal{D} = \bigcap_{i=1}^{m} \mathbf{dom}\ g_i \cap \bigcap_{i=1}^{p} \mathbf{dom}\ h_i.$$

Říkáme, že bod $x \in \mathcal{D}$ je **přípustný**, jestliže splňuje všechna omezení $g_i(x) \leq 0$ a $h_i(x) = 0$. Úloha 2.1 je **přípustná**, jestliže existuje alespoň jeden bod $x \in \mathcal{D}$, který je přípustný. Množina všech přípustných bodů $x \in \mathcal{D}$ se nazývá **přípustná množina**. **Optimální hodnota** f^* úlohy 2.1 je definována jako

$$f^* = \inf \{f(x) \mid g_i(x) \leq 0, i = 1, \ldots, m, h_i(x) = 0, i = 1, \ldots, p\}.$$

2.2 Konvexní podmíněná úloha

$$\begin{aligned} &\inf\ f(x) \\ &g_i(x) \leq 0, i = 1, \ldots, m \\ &a_i^T x = b_i, i = 1, \ldots, p \end{aligned} \tag{2.2}$$

Oproti obecné úloze 2.1 jsou funkce $f(x), g_i(x)$ konvexní a funkce $h_i(x) = a_i^T x - b_i$ jsou afinní. Přípustná množina takové úlohy je konvexní množinou.

2.3 Lagrangeova dualita

Mějme úlohu 2.1 s $\mathcal{D} \neq \emptyset$. Zobrazení $L:\ \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ takové, že

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \mu_i h_i(x) \tag{2.3}$$

se nazývá **Lagrangeova funkce**. Definiční obor **dom** $L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. Vektory $\lambda = (\lambda_1, \ldots, \lambda_m)$ a $\mu = (\mu_1, \ldots, \mu_p)$ nazýváme **duální proměnné** a prvkům těchto vektorů říkáme **Lagrangeovy multiplikátory**. Dále definujeme **duální funkci**

$$d:\ \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$$

jako infimum Lagrangeovy funkce L přes všechna $x \in \mathcal{D}$. Tedy

$$d(\lambda, \mu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \mu) = \inf_{x \in \mathcal{D}} \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \mu_i h_i(x) \right). \tag{2.4}$$

Poznamenejme, že duální funkce je konkávní bez ohledu na to, zda je úloha konvexní a je-li L zdola neomezená v proměnné x, potom duální funkce nabývá hodnoty $-\infty$.

Dolní odhad na f^*

Nechť $\tilde{x}$ je přípustný bod. Pro $\lambda \geq 0$ je

$$\sum_{i=1}^{m} \lambda_i g_i(\tilde{x}) + \sum_{i=1}^{p} \mu_i h_i(\tilde{x}) \leq 0.$$

Potom pro Lagrangeovu funkci můžeme psát

$$L(\tilde{x}, \lambda, \mu) = f(\tilde{x}) + \sum_{i=1}^{m} \lambda_i g_i(\tilde{x}) + \sum_{i=1}^{p} \mu_i h_i(\tilde{x}) \leq f(\tilde{x}).$$

A tedy pro duální funkci platí

$$d(\lambda, \mu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \mu) \leq L(\tilde{x}, \lambda, \mu) \leq f(\tilde{x}).$$

Duální úloha

V části 2.3 jsme si ukázali, že duální funkce dává dolní odhad na optimální hodnotu f^* úlohy 2.1. Stále jsme si ale neřekli, jaký je nejlepší dolní odhad, který pomocí duální funkce jsme schopni dostat. To nás dostává k následující optimalizační úloze.

$$\begin{aligned} &\sup d(\lambda, \mu) \\ &\lambda \geq 0 \end{aligned} \tag{2.5}$$

Úloze 2.5 se říká **Lagrangeova duální úloha** příslušná k úloze 2.1, kterou nazýváme **primární úlohou**.

Slabá dualita

Optimální řešení Lagrangeovy duální úlohy označíme d^*, které je už z definice nejlepší dolní odhad na optimální řešení primární úlohy f^*. Tato nerovnost platí i pokud primární úloha není konvexní. Této nerovnosti říkáme **slabá dualita**. Rozdíl optimálních řešení $f^* - d^*$ označujeme jako **optimální dualitní rozdíl** primární úlohy. Poznamenejme, že optimální dualitní rozdíl je vždy nezáporný.

Silná dualita a Slaterova podmínka

Pokud je optimální dualitní rozdíl $f^* - d^* = 0$, pak říkáme, že platí silná dualita. Silná dualita obecně neplatí, ale pro primární úlohu, která splňuje nějaké další podmínky to možné je. Těmto podmínkám se říká **podmínky kvalifikace omezení**. Jednou takovou je **Slaterova podmínka**

$$\exists x \in \textbf{relint } \mathcal{D} : \ g_i(x) < 0, i = 1, \ldots, m, Ax = b.$$

Bodu $x \in \mathcal{D}$, který splňuje Slaterovu podmínku, říkáme, že je **striktně přípustný**, protože omezení typu nerovnosti jsou ostré. Pokud jsou některé funkce g_i afinní, můžeme Slaterovu podmínku modifikovat. Nechť tedy $g_1, \ldots, g_k$ pro $k \leq m$, jsou afinní funkce. Potom **modifikovaná Slaterova podmínka** má tvar

$$\exists x \in \textbf{relint } \mathcal{D} : \ g_i(x) \leq 0, i = 1, \ldots, k, g_i(x) < 0, i = k+1, \ldots, m, Ax = b.$$

Pro úlohu 2.2 platí následující věta.

Věta (Slaterova)**.** *Nechť primární úloha je konvexní a platí (modifikovaná) Slaterova podmínka, potom $f^* = d^*$.*

Kapitola 3

Lineární programování

Kapitola je zpracována z [1], [6] a [24].

3.1 Primární úloha

Úlohou lineárního programování rozumíme minimalizaci nebo maximalizaci lineární **účelové funkce** vzhledem k afinním **omezením**, kde tato omezení jsou dána soustavou lineární rovnic a nerovnic. Úlohu lineárního programování lze formulovat v několika ekvivalentních tvarech, které se liší zadáním omezení. Úloha v **kanonickém tvaru** má svá omezení dána soustavou lineárních nerovnic $Ax \leq b$. Tedy

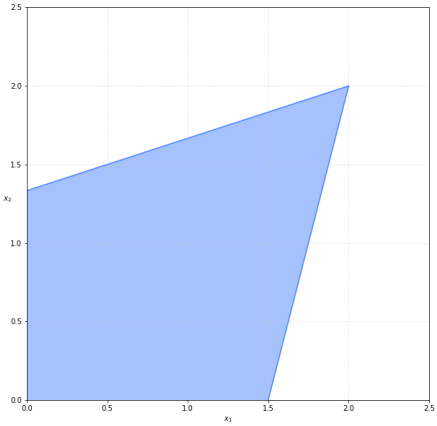
$$\max \left\{ c^T x \mid Ax \leq b, x \geq 0 \right\}, \tag{LP-P}$$

kde $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ a $c \in \mathbb{R}^n$. **Přípustná množina řešení** je průnikem poloprostorů, které jsou definovány soustavou nerovnic $Ax \leq b$ a **nezáporného ortantu**, tj. množiny $\{x \in \mathbb{R}^n \mid x_i \geq 0, i = 1, \ldots, n\}$. Obě tyto množiny jsou konvexní a tedy i jejich průnik je rovněž konvexní množina. Dále, protože přípustnou množinu máme popsanou soustavou konečně mnoha lineárních nerovnic, geometricky se na úlohu LP-P díváme jako na maximalizaci lineární funkce přes polyedr, který je definován touto soustavou.

Příklad. Mějme následující maximalizační úlohu.

$$\begin{aligned} \max x_1 + x_2 & \\ -x_1 + 3x_2 &\leq 4 \\ 4x_1 - x_2 &\leq 6 \\ x &\geq 0. \end{aligned} \tag{P1}$$

Přípustná množina řešení je zobrazena na obrázku 3.1. Řešením úlohy je vektor $x^* = (2, 2)$ s cenou 4.



Obrázek 3.1: Přípustná množina řešení k úloze P1.

3.2 Dualita

Pomocí Lagrangeovy duality odvodíme duální úlohu k primární úloze LP-P. Máme tedy optimalizační úlohu

$$\min \left\{ -c^T x \mid Ax \leq b, x \geq 0 \right\}.$$

Pro ní vytvoříme Lagrangeovu funkci

$$\begin{aligned} L(x, \lambda) &= -c^T x + \lambda^T (Ax - b) - \lambda^T x \\ &= -b^T \lambda + \left(A^T \lambda - c - \lambda \right)^T x. \end{aligned}$$

Z Lagrangeovy funkce přejdeme k duální funkci

$$\begin{aligned} d(\lambda) &= \inf_x L(x, \lambda) \\ &= \inf_x -b^T \lambda + \left(A^T \lambda - c - \lambda \right)^T x \\ &= \begin{cases} -b^T \lambda & \text{pokud } A^T \lambda - c - \lambda = 0, \\ -\infty & \text{jinak.} \end{cases} \end{aligned}$$

Tu nakonec použijeme v duální úloze

$$\max \left\{ -b^T \lambda \mid A^T \lambda - c - \lambda = 0 \right\}$$

$$\max\left\{-b^T\lambda \mid A^T\lambda \geq c, \lambda \geq 0\right\}$$

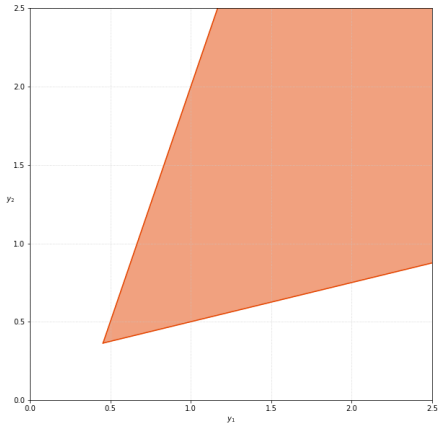
$$\min\left\{b^T\lambda \mid A^T\lambda \geq c, \lambda \geq 0\right\} \tag{LP-D}$$

Dostáváme tedy duální úlohu LP-D k primární úloze LP-P.

Příklad. Duální úloha k P1 je v následujícím tvaru.

$$\begin{aligned} \min\, & 4y_1 + 6y_2 \\ -y_1 + 4y_2 &\geq 1 \\ 3y_1 - y_2 &\geq 1 \\ y &\geq 0. \end{aligned} \tag{P2}$$

Přípustná množina řešení je zobrazena na obrázku 3.2. Řešením úlohy je vektor $y^* \approx (0.4546, 0.3636)$ s cenou 4.



Obrázek 3.2: Přípustná množina řešení k úloze P2.

Všimněme si, že v příkladech P1 a P2 mají řešení x^* i y^* stejnou cenu. To není náhoda a tento fakt je obsahem silné věty o dualitě lineárního programování, kterou dokázal v roce 1947 John von Neumann. Začneme slabou větou o dualitě lineárního programování.

Věta (Slabá věta o dualitě.)**.** *Nechť $\tilde{x}$ je přípustné řešení LP-P a $\tilde{y}$ je přípustné řešení LP-D. Potom $c^T\tilde{x} \leq b^T\tilde{y}$.*

Tedy každé přípustné řešení $\tilde{y}$ duální úlohy LP-D nám dává horní odhad na maximum účelové funkce primární úlohy LP-P. Graficky můžeme slabou větu o dualitě interpretovat jako na obrázku 3.3. Zatím tedy nevíme, zda vždy existují přípustná (optimální) řešení x^* pro úlohu LP-P a y^* pro úlohu LP-D, pro která platí $c^T x^* = b^T y^*$. Kladnou odpověď dostaneme z již zmíněné silné věty od dualitě.

Obrázek 3.3: Slabá věta o dualitě.

Věta (Silná věta o dualitě.)**.** *Jestliže úlohy LP-P a LP-D mají přípustná řešení. Potom*

$$\max\left\{c^T x \mid Ax \leq b, x \geq 0\right\} = \min\left\{b^T y \mid A^T y \geq c, y \geq 0\right\}.$$

Se znalostí silné věty o dualitě můžeme obrázek 3.3 upravit na obrázek 3.4.

Obrázek 3.4: Ceny přípustných řešení primární a příslušné duální úlohy.

3.3 Komplementární skluzovost

Pro odvození tzv. podmínky komplementární skluzovosti nejprve převedeme úlohy LP-P a LP-D do jiných tvarů. V primární úloze povolíme $x \in \mathbb{R}^n$. Tedy primární úloha je ve tvaru

$$\max\left\{c^T x \mid Ax \leq b\right\}. \tag{LP-P2}$$

A příslušná duální úloha je ve tvaru

$$\min\left\{b^T y \mid A^T y = c, y \geq 0\right\}. \tag{LP-D2}$$

Nechť $\tilde{x}$ je připustné řešení a x^* je optimální řešení úlohy LP-P2, $\tilde{y}$ je přípustné řešení a y^* je optimální řešení úlohy LP-D2. **Dualitní rozdíl** $\tilde{x}$ a $\tilde{y}$ je číslo $b^T\tilde{y} - c^T\tilde{x} \geq 0$. Ze silné věty o dualitě samozřejmě plyne, že

pro optimální řešení x^* a y^* je dualitní rozdíl roven 0. Vyjdeme z dualitního rozdílu optimálních řešení.

$$b^T y^* - c^T x^* = {y^*}^T b - {y^*}^T A x^* = {y^*}^T (b - A x^*) = 0.$$

Poslední rovnost přepíšeme maticově

$$[y_1^*, \ldots, y_m^*] \left(\begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} - \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \ldots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1^* \\ \vdots \\ x_n^* \end{bmatrix} \right) = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Dostáváme tedy soustavu rovnic $y_i^* (b_i - a_{i\_} x^*) = 0$, kde $i = 1, \ldots, m$. Tedy buď $y_i^* = 0$ nebo $b_i - a_{i\bullet} x^* = 0$ ($a_{i\bullet}$ značí i-tý řádek matice A). **Podmínka komplementární skluzovosti** je splněna, jestliže pro přípustná řešení $\tilde{x}, \tilde{y}$ platí buď $\tilde{y}_i = 0$ nebo $b_i - a_{i\bullet} \tilde{x} = 0$, $i = 1, \ldots, m$. Pokud nastane $b_i - a_{i\bullet} \tilde{x} = 0$, potom říkáme, že **vazba** $a_{i\_} \tilde{x} \leq b_i$ **je aktivní**.

Věta. *Nechť $\tilde{x}$ je přípustné řešení LP-P2 a $\tilde{y}$ je přípustné řešení LP-D2. Potom $\tilde{x}, \tilde{y}$ jsou optimální právě tehdy, když platí podmínka komplementární skluzovosti.*

Kapitola 4

Semidefinitní programování

Kapitola je zpracována z [1], [3], [20], [21] a [24].

Na semidefinitní programování se můžeme dívat jako na zobecnění lineárního programování, kde proměnné jsou symetrické matice, namísto vektorů. Jedná se tedy o optimalizaci lineární funkce vzhledem k tzv. lineárním maticovým nerovnostem (viz dále).

4.1 Vsuvka z lineární algebry

Pozitivně definitní matice

Pracujeme s reálnými symetrickými maticemi $S = S^T$. Ty mají všechna vlastní čísla reálná a některé z nich mají zajímavou vlastnost, že všechna jejich vlastní čísla jsou kladná. Takovým maticím říkáme, že jsou pozitivně definitní. Alternativní definicí je, že matice S je pozitivně definitní, jestliže $x^T Sx > 0$ pro všechny nenulové vektory x.

Příklad.

$$x^T Sx = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2x_1^2 + 8x_1x_2 + 9x_2^2$$

Je pro všechny x nenulové $x^T Sx > 0$? Ano, protože můžeme výraz přepsat na součet čtverců

$$x^T Sx = 2x_1^2 + 8x_1x_2 + 9x_2^2 = 2(x_1 + 2x_2)^2 + x_2^2.$$

Ukážeme si několik kritérií, jak otestovat pozitivní definitnost dané matice.

Věta. *[20] $S = S^T$ je pozitivně definitní, jestliže ji lze napsat jako $S = A^T A$ pro nějakou matici A, která má lineárně nezávislé sloupce.*

Důkaz.

$$x^T S x = x^T A^T A x = (Ax)^T (Ax) = \|Ax\|^2 \geq 0$$

$\|Ax\|^2 > 0$, jestliže sloupce matice A jsou lineárně nezávislé. □

Příklad.

$$S = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 5 & 7 \\ 4 & 7 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} = A^T A$$

A má lineárně závislé sloupce. Vezměme vektor $x = (1, -2, 1)$. Platí

$$x^T S x = x^T A^T A x = (Ax)^T Ax = 0,$$

tj. existuje nenulový vektor x, pro který $x^T S x = 0$. Tedy S není pozitivně definitní.

Dalším testem je tzv. Sylvesterovo kritérium. **Vedoucí hlavní minor** čtvercové matice A je determinant podmatice, která vznikne z matice A vynecháním posledních několika jejích řádků a sloupců.

Věta. *[20] $S = S^T$ je pozitivně definitní, jestliže všechny vedoucí hlavní minory S jsou kladné.*

Příklad.

$$S = \begin{bmatrix} 3 & 4 \\ 4 & 6 \end{bmatrix}, D_1 = 3, D_2 = 3 \cdot 6 - 4 \cdot 4 = 2$$

Vedoucí hlavní minory $D_1, D_2 > 0$, tj. matice S je pozitivně definitní.

A poslední, které si uvedeme, souvisí s Gaussovou eliminací.

Věta. *[20] $S = S^T$ je pozitivně definitní, jestliže jsou všechny pivoty při Gaussově eliminaci kladné.*

Příklad.

$$S = \begin{bmatrix} 3 & 4 \\ 4 & 6 \end{bmatrix} \sim \begin{bmatrix} 3 & 4 \\ 0 & \frac{2}{3} \end{bmatrix}, p_1 = 3, p_2 = \frac{2}{3}$$

Pivoty $p_1, p_2 > 0$, tj. matice S je pozitivně definitní.

Pozitivně semidefinitní matice

Pro pozitivní semidefinitnost modifikujeme předcházejí definice a tvrzení pro pozitivně definitní matice. Všechna tvrzení by se samozřejmě měla dokázat, ale uvedeme je jen bez důkazů.

1. $S = S^T$ je pozitivně semidefinitní, jestliže všechna její čísla jsou nezáporná.
2. $S = S^T$ je pozitivně semidefinitní, jestliže $x^T S x \geq 0$ pro všechny nenulové vektory x.
3. [20] $S = S^T$ je pozitivně semidefinitní, jestliže lze napsat jako $S = A^T A$ pro nějakou matici A.
4. [20] $S = S^T$ je pozitivně semidefinitní, jestliže všechny vedoucí hlavní minory S jsou nezáporné.
5. [20] $S = S^T$ je pozitivně semidefinitní, jestliže jsou všechny pivoty při eliminaci nezáporné.

Pozitivně semidefinitní kužel

Množinu všech symetrických matic řádu n značíme S^n, množinu všech pozitivně semidefinitních matic řádu n značíme S^n_+ a množinu všech pozitivně definitních matic řádu n značíme S^n_{++}. S^n_+ je uzavřený, konvexní, bodový kužel, který je navíc samoduální a jako vnitřek má S^n_{++} (více viz [11]). Množinu S^n_+ nazýváme **pozitivně semidefinitní kužel**.

Spektraedry

Definujeme tzv. **Löwnerovo částečné uspořádání**

$$A \succeq B \iff A - B \in S^n_+,$$

tj. matice $A - B$ je pozitivně semidefinitní. **Lineární maticová nerovnost** (LMI) je ve tvaru

$$A_0 + \sum_{i=1}^{n} A_i x_i \succeq 0,$$

kde $A_i \in S^n$.

Množina $S \subset \mathbb{R}^n$, která je definována pomocí konečně mnoha LMI, se nazývá **spektraedr**. Tedy

$$S = \left\{ (x_1, \ldots, x_m) \in \mathbb{R}^m \mid A_0 + \sum_{i=1}^{m} A_i x_i \succeq 0 \right\}$$

pro nějaké symetrické matice $A_0, \ldots, A_m \in S^n$.

Můžeme si všimnout analogie s definicí polyedru, který je přípustnou množinu pro lineární program. Podobně spektraedr je přípustnou množinou pro semidefinitní program.

Geometricky je spektraedr definován jako průnik pozitivně semidefinitního kuželu S^n_+ a afinního podprostoru $\mathbf{span}\{A_1, \ldots, A_m\}$ posunutého do A_0.

Spektraedry jsou uzavřené množiny, neboť LMI je ekvivalentní nekonečně mnoha skalárním nerovnostem ve tvaru $v^T(A_0 + \sum_{i=1}^m A_i x_i)v \geq 0$, jednu pro každou hodnotu $v \in \mathbb{R}^n$.

Vždy můžeme několik LMI „scucnout“ do jedné. Stačí zvolit matice A_i blokově diagonální. Odtud snadno vidíme, že polyedr je speciálním případem spektraedru. Polyedr bude mít všechny matice A_i diagonální.

Příklad.

$$\left\{ (x,y) \in \mathbb{R}^2 \mid A(x,y) = \begin{bmatrix} x+1 & 0 & y \\ 0 & 2 & -x-1 \\ y & -x-1 & 2 \end{bmatrix} \succeq 0 \right\}$$

4.2 Primární úloha

Semidefinitní program je lineární optimalizační problém přes spektraedr. Primární úlohu ve standardním tvaru můžeme napsat jako

$$\inf\left\{ \langle C, X \rangle \mid \langle A_i, X \rangle = b_i, i = 1, \ldots, m; X \succeq 0 \right\}, \tag{SDP-P}$$

kde $C, A_i \in S^n$, $\langle X, Y \rangle = \mathbf{Tr}(X^T Y) = \sum_{ij} X_{ij} Y_{ij}$ a $X \in S^n$ je proměnná, nad kterou provádíme minimalizaci.

Příklad.

$$\inf\left\{ \left\langle \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{bmatrix} \right\rangle \middle| \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{bmatrix} \right\rangle = 1, \begin{bmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{bmatrix} \succeq 0 \right\} \tag{P3}$$

Po úpravě

$$\inf\left\{ 2x_{11} + 2x_{12} \middle| x_{11} + x_{22} = 1, \begin{bmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{bmatrix} \succeq 0 \right\}.$$

Jak vypadá přípustná množina? Použijeme Sylvesterovo kritérium, tj.

$$x_{11} \geq 0, x_{11}x_{22} - x_{12}^2 \geq 0.$$

Z LMI vyjádříme x_{22}, tj.

$$x_{22} = 1 - x_{11}.$$

Dosadíme do přechozího a dostaneme

$$x_{11} \geq 0, x_{11}\left(1 - x_{11}\right) - x_{12}^2 \geq 0.$$

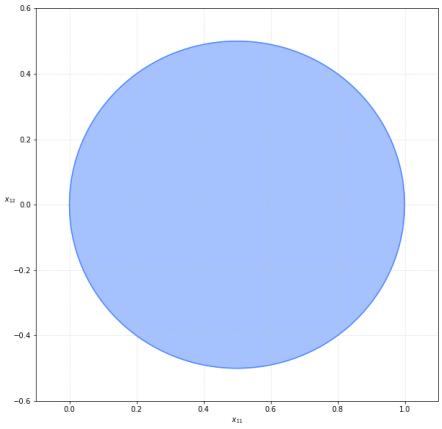
Po úpravě

$$x_{11} \geq 0, \left(x_{11} - \frac{1}{2}\right)^2 + x_{12}^2 \leq \frac{1}{4}.$$

Vidíme tedy, že přípustná množina (zobrazena na obrázku 4.1) je kruh s poloměrem $\frac{1}{2}$ a se středem v bodě $(x_{11}, x_{12}) = (\frac{1}{2}, 0)$. Řešením úlohy je matice

$$X^* \approx \begin{bmatrix} 0.1464 & -0.3536 \\ -0.3536 & 0.8536 \end{bmatrix}$$

s cenou ≈ -0.4142.



Obrázek 4.1: Přípustná množina řešení k úloze P3.

4.3 Dualita

Duální úloha

Podobně jako u lineárního programování použijeme Lagrangeovu dualitu k odvození duální úlohy k SDP-P. Lagrangeova funkce je ve tvaru

$$L(X, \lambda, Z) = \langle C, X \rangle - \sum_{i=1}^{m} \lambda_i \left(\langle A_i, X \rangle - b_i\right) - \langle Z, X \rangle.$$

K ní duální funkce

$$d(\lambda, Z) = \inf_{X \in S^n} L(X, \lambda, Z) = \begin{cases} \lambda^T b & \dots \ C - \sum_{i=1}^m \lambda_i A_i - Z = 0, \\ -\infty & \dots \text{ jinak.} \end{cases}$$

Duální funkci použijeme v duální úloze

$$\sup \left\{ \lambda^T b \mid C - \sum_{i=1}^m \lambda_i A_i \succeq 0 \right\}, \tag{SDP-D}$$

kde $\lambda = (\lambda_1, \dots, \lambda_m)$ je duální proměnná.
Dostali jsme duální úlohu SDP-D k úloze SDP-P.

Slabá dualita semidefinitního programování

Vztah mezi primární a duální úlohou je stejně jako u lineárního programování takový, že řešení jedné úlohy lze použít jako odhad na úlohu druhou. Nechť X je libovolné přípustné řešení primární úlohy a y je libovolné přípustné řešení duální úlohy. Potom

$$\langle C, X \rangle - b^T y = \langle C, X \rangle - \sum_{i=1}^m y_i \langle A_i, X \rangle = \left\langle C - \sum_{i=1}^m A_i y_i, X \right\rangle \geq 0. \tag{4.1}$$

Za pozornost stojí poslední nerovnost, která plyne z toho, že skalární součin dvou pozitivně semidefinitních matic je nezáporný. Odvození je následující. Mějme dvě matice $S, T \succeq 0$. Matici S můžeme napsat jako součet „rank one“ matic. Označme $r_S = \mathbf{rank}\ S$, tj.

$$S = \sum_{i=1}^{r_S} S_i = \sum_{i=1}^{r_S} \lambda_i s_i s_i^T.$$

Dále se podíváme na součin $T \cdot S_i$. Tedy pro $i = 1, \dots, r_S$ platí

$$T \cdot S_i = \lambda_i s_i^T T s_i \geq 0,$$

kde nerovnost plyne z toho, že matice T je pozitivně semidefinitní.

O nerovnosti 4.1 se mluví jako o slabé dualitě semidefinitního programování.

Silná dualita semidefinitního programování

Věta (podmínka optimality)**.** *Nechť X je přípustné řešení úlohy SDP-P a y je přípustné řešení úlohy SDP-D taková, že splňují podmínku (komplementární skluzovosti)*

$$\left(C - \sum_{i=1}^{m} A_i y_i\right) X = 0.$$

Potom X je optimální řešení úlohy SDP-P a y je optimální řešení úlohy SDP-D.

Obracená implikace sama o sobě neplatí, což znamená, že obecně dualitní rozdíl u semidefinitního programování není nulový. Musíme přidat podmínku kvalifikace omezení, kterou je například již zmíněná Slaterova podmínka. Ta je pro úlohu SDP-P ve tvaru $X \succ 0$ a pro úlohu SDP-D ve tvaru $C - \sum_i A_i y_i \succ 0$.

Věta (silná dualita semidefinitního programování)**.** *Nechť úloha SDP-P a úloha SDP-D jsou striktně přípustné. Potom dualitní rozdíl jejich optimálních řešení je nulový.*

4.4 Vektorové programování

Mějme n vektorových proměnných $v_1, \ldots, v_n$ v $\mathbb{R}^n$. **Vektorový program** je problém optimalizace lineární funkce skalárních součinů $\langle v_i, v_j \rangle$, vzhledem k lineárním omezením na tyto skalární součiny.

Nyní ukážeme, že vektorové programy jsou ekvivalentní semidefinitním programům. Nechť tedy $\mathcal{V}$ je vektorový program s vektorovými proměnnými $v_1, \ldots, v_n$ v $\mathbb{R}^n$ a $\mathcal{S}$ je příslušný semidefinitní program s n^2 proměnnými y_{ij}, kde hodnota y_{ij} odpovídá skalárnímu součinu $\langle v_i, v_j \rangle$. Matice Y je navíc pozitivně semidefinitní. Potom platí následující věta.

Věta. *[21] Vektorový program $\mathcal{V}$ je ekvivalentní semidefinitnímu programu $\mathcal{S}$.*

Část II

Kombinatorické úlohy

Kapitola 5

Shannonova kapacita

Kapitola je zpracována z [7] a citovaných článků, ze kterých jsou „zřejmá“ tvrzení více rozepsána.

5.1 Formulace úlohy

Představme si komunikační kanál, kterým posíláme zprávy. Tyto zprávy jsou složeny ze symbolů nějaké konečné abecedy. Vlivem šumu mohou být některé symboly druhou stranou špatně interpretovány a naším cílem je vybrat co největší množinu slov délky k tak, aby žádná dvě odeslaná slova nebyla vlivem tohoto šumu zaměnitelná.

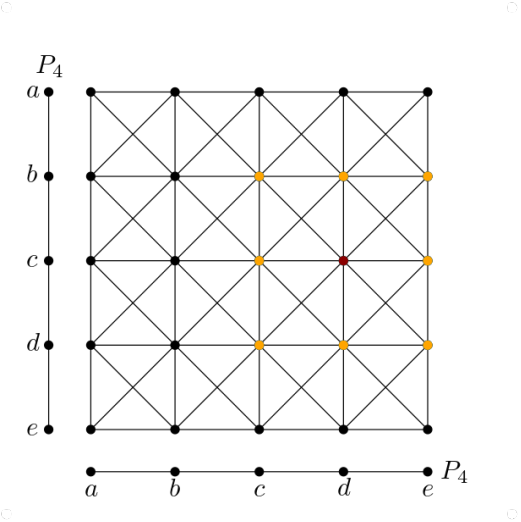
Problém si formalizujeme v řeči teorie grafů. Mějme neorientovaný graf $G = (V, E)$, kde množina vrcholů představuje symboly z konečné abecedy a dva vrcholy x, y jsou spojeny hranou, pokud vrchol x může být vlivem šumu zaměněn za y.

Maximální počet nezaměnitelných zpráv délky 1 je roven $\alpha(G)$, kde $\alpha(G)$ značí velikost největší nezávislé množiny v grafu G. Pro popis delších zpráv definujeme **silný součin** $G \cdot H$ grafů G a H následovně

$$V(G \cdot H) = V(G) \times V(H),$$

$$\begin{aligned} E(G \cdot H) = &\{(i,u)(j,v) \mid ij \in E(G) \wedge uv \in E(H)\} \cup \\ &\{(i,u)(j,v) \mid ij \in E(G) \wedge u = v\} \cup \\ &\{(i,u)(j,v) \mid i = j \wedge uv \in E(H)\}. \end{aligned}$$

Příklad. Pro graf $P_4 = a - b - c - d - e$ je silný součin $P_4 \cdot P_4$ zobrazen na obrázku 5.1, ze kterého je hezky vidět, že např. zpráva cd (na obrázku červeně) může být zaměněna s bc, bd, be, cc, ce, dc, dd a de (na obrázku oranžově). Podobně pro další zprávy.



Obrázek 5.1: $P_4 \cdot P_4$

Pro jednoduchost budeme silný součin k kopií grafu G značit G^k. Tedy $\alpha(G^k)$ je maximální počet nezaměnitelných zpráv délky k. **Shannonova kapacita** grafu G je definována jako

$$\Theta(G) = \sup\left\{\alpha(G^k)^{1/k} \mid k = 1, 2, \dots\right\}.$$

Neví se, zda pro libovolný graf G existuje vůbec nějaký algoritmus, kterým bychom určili hodnotu $\Theta(G)$. Přesto je alespoň něco známo. Pro perfektní grafy Claude E. Shannon [18] ukázal, že $\Theta(G) = \alpha(G)$. To také znamená, že pro perfektní grafy lze $\Theta(G)$ určit v polynomiálním čase. Poznamenejme, že perfektní graf je takový, jehož chromatické číslo každého indukovaného podgrafu je stejné jako velikost největší kliky v tomto podgrafu. Příklady perfektních grafů jsou například bipartitní grafy a hranové grafy. Dalším kdo se problémem zabýval byl László Lovász [14], který velmi hezkým způsobem ukázal, že kružnice délky 5 má kapacitu $\sqrt{5}$. Na Lovászův postup se dále podíváme, protože vede k obecnému hornímu odhadu na $\Theta(G)$.

5.2 $\Theta(C_5) = \sqrt{5}$

Nejprve potřebujeme zavést několik pojmů. **Tenzorový součin** vektorů $u = (u_1, \dots, u_n)$ a $v = (v_1, \dots, v_m)$ je

$$u \circ v = (u_1 v_1, \dots, u_1 v_m, u_2 v_1, \dots, u_n v_m).$$

Užitečné bude následující pozorování, které dává do souvislosti skalární a tenzorový součin.

Pozorování. *Nechť x, u jsou vektory délky n a y, v jsou vektory délky m. Potom platí*

$$(x \circ y)^T (u \circ v) = \left(x^T u\right) \left(y^T v\right). \tag{5.1}$$

Důkaz. Levá strana:

$$(x_1y_1, x_1y_2, \ldots, x_1y_m, \ldots, x_ny_m)^T (u_1v_1, u_1v_2, \ldots, u_1v_m, \ldots, u_nv_m) =$$
$$x_1y_1u_1v_1 + x_1y_2u_1v_2 + \cdots + x_1y_mu_1v_m + \cdots + x_my_mu_nv_m$$

Pravá strana:

$$(x_1u_1 + \cdots + x_nu_n) \cdot (y_1v_1 + \cdots + y_nv_m) =$$
$$x_1y_1u_1v_1 + x_1y_2u_1v_2 + \cdots + x_1y_mu_1v_m + \cdots + x_my_mu_nv_m$$

□

Mějme graf $G = (V, E)$, kde $V = \{1, \ldots, n\}$. Systém $(v_1, \ldots, v_n)$ jednotkových vektorů v Euklidovském prostoru takový, že

$$\forall ij \notin E \implies v_i \perp v_j$$

nazýváme **ortonormální reprezentace** grafu G. Poznamenejme, že každý graf má nějakou ortonormální reprezentaci, např. $1 \mapsto e_1, \ldots, n \mapsto e_n$.

Lemma 1. *[14] Nechť $(u_1, \ldots, u_n)$ je ortonormální reprezentace grafu G a $(v_1, \ldots, v_m)$ je ortonormální reprezentace grafu H. Potom $u_i \circ v_j$ je ortonormální reprezentace grafu $G \cdot H$.*

Důkaz. Použijeme vztah 5.1. $(u_i \circ v_j)^T (u_k \circ v_l) = \left(u_i^T u_k\right) \left(v_j^T v_l\right) = 0 \iff ik \notin E(G) \vee jl \notin E(H)$. □

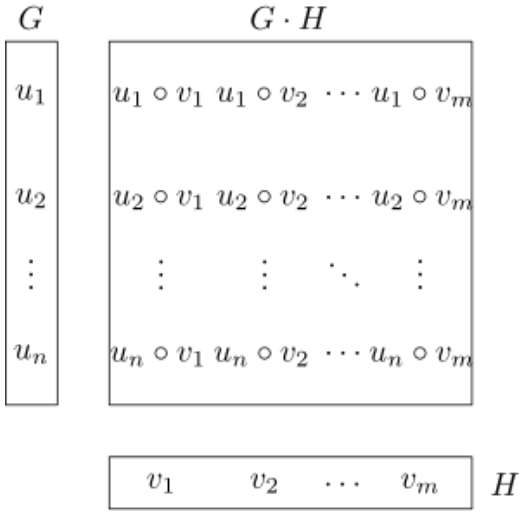
Hodnotu ortonormální reprezentace $(u_1, \ldots, u_n)$ definujeme jako

$$\min_c \max_{i=1,\ldots,n} \frac{1}{\left(c^T u_i\right)^2}.$$

Vektoru c, pro který nastává minimum říkáme **handle** dané ortonormální reprezentace.

Dále definujeme funkci $\vartheta(G)$ jako minimální hodnotu přes všechny ortonormální reprezentace grafu G. Ortonormální reprezentaci, pro kterou nastává minumum nazýváme **optimální**. Funkci $\vartheta(G)$ se říká **Lovászova theta funkce** a ona je právě již zmíněným horním odhadem na $\Theta(G)$. Podívejme se na některé její vlastnosti.

Lemma 2. *[14]* $\vartheta(G \cdot H) \leq \vartheta(G)\vartheta(H)$



Obrázek 5.2: Lemma 1

Důkaz. Nechť $(u_1, \ldots, u_n)$ je optimální ortonormální reprezentace grafu G s handle c a $(v_1, \ldots, v_m)$ je optimální ortonormální reprezentace grafu H s handle d. Pak $c \circ d$ je jednotkový vektor a platí

$$\vartheta(G{\cdot}H) \leq \max_{i,j} \frac{1}{\left((c \circ d)^T (u_i \circ v_j)\right)^2} = \max_i \frac{1}{(c^T u_i)^2} {\cdot} \max_j \frac{1}{(d^T v_j)^2} = \vartheta(G)\vartheta(H).$$

□

Lemma 3. *[14]* $\alpha(G) \leq \vartheta(G)$

Důkaz. Mějme maximální nezávislou množinu $I \subseteq V(G)$ v grafu G a optimální ortonormální reprezentaci $\mathcal{U} = (u_1, \ldots, u_n)$ grafu G s handle c. Platí

$$\forall i, j \in I : \; i \neq j \implies u_i \perp u_j.$$

Máme tedy systém ortonormálních vektorů $\{u_i \in \mathcal{U} \mid i \in I\}$ v $\mathbb{R}^n$. Ten rozšíříme na ortonormální bázi $\mathcal{B}$. Potom i-tá souřadnice vektoru c v bázi $\mathcal{B}$ je $c^T u_i$. Tedy

$$1 = \|c\|^2 = \sum_{i=1}^{n} \left(c^T u_i\right)^2.$$

Dále vynecháme přidáné vektory do ortonormální báze $\mathcal{B}$

$$\sum_{i=1}^{n} \left(c^T u_i\right)^2 \geq \sum_{i \in I} \left(c^T u_i\right)^2.$$

Poslední výraz přepíšeme

$$\sum_{i\in I}\left(c^T u_i\right)^2 \geq |I| \cdot \min_{i\in I}\left(c^T u_i\right)^2 = \alpha(G)\cdot \min_{i\in I}\left(c^T u_i\right)^2.$$

Předchozí výrazy dáme dohromady

$$1 \geq \alpha(G)\cdot \min_{i\in I}\left(c^T u_i\right)^2,$$

a dostáváme

$$\alpha(G) \leq \frac{1}{\min_{i\in I}\left(c^T u_i\right)^2} = \max_{i\in I}\frac{1}{\left(c^T u_i\right)^2} \leq \max_{i\in V(G)}\frac{1}{\left(c^T u_i\right)^2} = \vartheta(G).$$

□

Lemma 4. *[14]* $\Theta(G) \leq \vartheta(G)$

Důkaz. Pro každé k platí, že

$$\alpha(G^k) \leq \vartheta(G^k) \leq \vartheta(G)^k.$$

Odtud

$$\sqrt[k]{\alpha(G^k)} \leq \vartheta(G),$$

a limitním přechodem dostáváme požadovanou nerovnost

$$\Theta(G) = \lim_{k\to\infty}\sqrt[k]{\alpha(G^k)} \leq \vartheta(G).$$

□

Věta 1. *[14]* $\Theta(C_5) = \sqrt{5}$

Důkaz. Ukážeme konstrukci ortonormální reprezentace grafu C_5, ze které dostaneme horní odhad na $\Theta(C_5)$. Nechť $V(C_5) = \{v_1, \ldots, v_5\}$ a $E(C_5) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_1v_5\}$. Mějme vektory $\bar{u}_i$ ve tvaru

$$\bar{u}_i = \left(\cos\frac{2\pi i}{5}, \sin\frac{2\pi i}{5}, z\right), i = 1, \ldots, 5.$$

Každý vektor $\bar{u}_i$ je svázán s vrcholem v_i. Chceme, aby dva vektory, které jsou příslušné nesousedním vrcholům, byly ortogonální. Tedy například $\langle \bar{u}_2, \bar{u}_5\rangle = 0$. Dosadíme

$$\langle \bar{u}_2, \bar{u}_5\rangle = \left\langle (\cos\frac{4\pi}{5}, \sin\frac{4\pi}{5}, z), (1, 0, z)\right\rangle = \cos\frac{4\pi}{5} + z^2 = 0.$$

Dostáváme tedy

$$z = \sqrt{-\cos\frac{4\pi}{5}}.$$

Definujeme ortonormální reprezentaci $\mathcal{U}$ grafu C_5 (projekce do první a druhé souřadnice, viz Obrázek 5.3) tak, že

$$u_i = \frac{\bar{u}}{\|\bar{u}\|}, i = 1, \ldots, 5,$$

s handle $c = (0, 0, 1)$. Dostáváme

$$\vartheta(C_5) \leq \vartheta(\mathcal{U}) = \max_{i=1,\ldots,5} \frac{1}{(c^T u_i)^2} = \frac{1}{(c^T u_5)^2} = \frac{1 - \cos\frac{4\pi}{5}}{-\cos\frac{4\pi}{5}}.$$

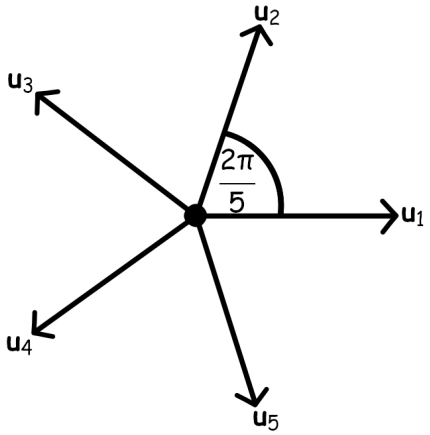
Do posledního výrazu dosadíme známou hodnotu pro $\cos 36°$.

$$\frac{1 - \cos\frac{4\pi}{5}}{-\cos\frac{4\pi}{5}} = \frac{1 + \frac{1+\sqrt{5}}{4}}{\frac{1+\sqrt{5}}{4}} = \frac{5 + \sqrt{5}}{1 + \sqrt{5}} = \sqrt{5}.$$

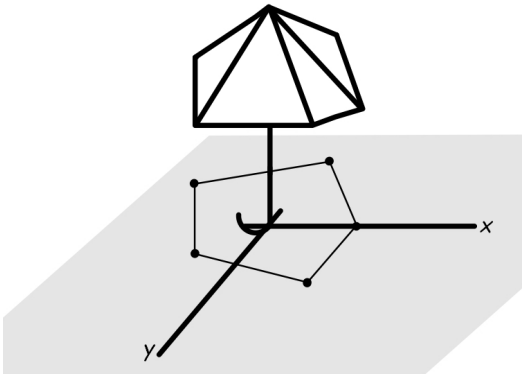
Dostáváme

$$\vartheta(C_5) \leq \sqrt{5}.$$

Této ortonormální reprezentaci se říká **Lovászův deštník**, viz Obrázek 5.4. Druhou nerovnost $\vartheta(C_5) \geq \sqrt{5}$ dostaneme snadno. Sice $\alpha(C_5) = 2$, ale $\alpha(C_5^2) = 5$. Z čehož plyne druhá nerovnost. □



Obrázek 5.3: Projekce u_i do první a druhé souřadnice.



Obrázek 5.4: Lovászův deštník.

5.3 Semidefinitní programy pro $\vartheta(G)$

Program pro $1/\sqrt{\vartheta}$

První formulací je semidefinitní program [7], jehož řešením je hodnota $1/\sqrt{\vartheta}$. Mějme graf $G = (V, E)$. Hodnota $\vartheta(G)$ je z definice

$$\vartheta(G) = \min_{\mathcal{U}} \vartheta(\mathcal{U}) = \min_{\mathcal{U}} \min_{\|c\|=1} \max_{i \in V(G)} \frac{1}{(c^T u_i)^2},$$

kde $\mathcal{U}$ probíhá přes všechny ortonormální reprezentace grafu G. Pokud se stane, že $c^T u_i \leq 0$, potom místo u_i budeme dále uvažovat vektor $-u_i$. Můžeme tedy předpokládat, že $\forall i \in V(G) : \; c^T u_i \geq 0$. Potom hodnotu $1/\sqrt{\vartheta(G)}$ můžeme vyjádřit jako

$$\frac{1}{\sqrt{\vartheta(G)}} = \max_{\mathcal{U}} \frac{1}{\sqrt{\vartheta(\mathcal{U})}} = \max_{\mathcal{U}} \max_{\|c\|=1} \min_{i \in V(G)} c^T u_i.$$

Z čehož dostaneme následující vektorový program

$$\begin{aligned} &\max t \\ &\forall ij \in E(\bar{G}) : \; \langle u_i, u_j \rangle = 0 \\ &\forall i \in V(G) : \; \langle c, u_i \rangle \geq t \\ &\forall i \in V(G) : \; \|u_i\| = 1 \\ &\|c\| = 1. \end{aligned} \tag{VP1}$$

Z vektorového programu VP1 dále odvodíme semidefinitní program. Budeme uvažovat matici

$$X = \begin{bmatrix} — & c^T & — \\ — & u_1^T & — \\ & \vdots & \\ — & u_n^T & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ c & u_1 & \dots & u_n \\ | & | & & | \end{bmatrix},$$

která je samozřejme pozitivně semidefinitní. Podmínkám $\forall i \in V(G) : \|u_i\| = 1$ a $\|c\| = 1$ odpovídá podmínka $x_{ii} = 1$ pro $i = 0, 1, \dots, n$ (pro teď budeme indexovat matici X od 0). Podmínku $\forall ij \in E(\bar{G}) : \langle u_i, u_j \rangle = 0$ přepíšeme na

$$\forall ij \in E(\bar{G}) : \ x_{ij} = 0.$$

A konečně poslední podmínku $\forall i \in V(G) : \langle c, u_i \rangle \geq t$ přepíšeme takto

$$\forall i \in V(G) : \ x_{0i} \geq t.$$

Dostáváme tedy semidefinitní program

$$\begin{aligned} &\max t \\ &x_{ii} = 1, i = 0, 1, \dots, n \\ &\forall ij \in E(\bar{G}) : \ x_{ij} = 0 \\ &\forall i \in V(G) : \ x_{0i} \geq t \\ &X \succeq 0. \end{aligned} \tag{SDP1}$$

Program pro ϑ

V původním článku od L. Lovásze [14] je další semidefinitní program, jehož řešením je přímo hodnota $\vartheta(G)$.

$$\begin{aligned} &\max \langle X, J \rangle \\ &\forall ij \in E(G) : \ x_{ij} = 0 \\ &\mathbf{Tr}\ X = 1 \\ &X \succeq 0, \end{aligned} \tag{SDP2}$$

kde J je matice samých jedniček.

5.4 ϑ a související grafové parametry

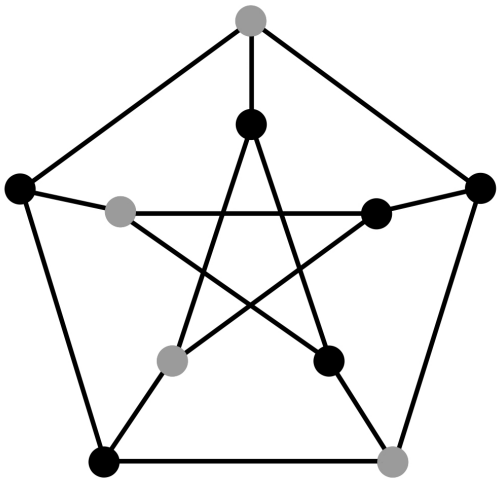
Začneme tzv. Sendvičovou větou a dále se zaměříme na grafy C_5, C_7, Petersenův graf, K_5 a S_5.

Věta 2. *[12] Mějme graf G a jeho doplněk $\bar{G}$. Potom*

$$\alpha(G) \leq \vartheta(G) \leq \chi(\bar{G}).$$

$\alpha(G)$ pro vybrané grafy

Je zřejmé, že pro $\alpha(C_5) = 2$, $\alpha(C_7) = 3$, $\alpha(K_5) = 1$ a $\alpha(S_5) = 5$. Na obrázku 5.5 je, pro Petersenův graf, nezávislá množina velikosti 4. Probírkou všech možností zjistíme, že větší nezávislou množinu se nám najít nepodaří.



Obrázek 5.5: Největší nezávislá množina v $GP_{5,2}$.

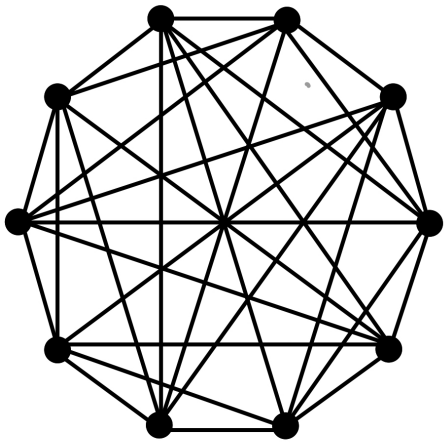
$\chi(\bar{G})$ pro vybrané grafy

Doplněk C_5 je opět C_5 a lichá kružnice má chromatické číslo 3. K_5 má jako svůj doplněk diskrétní graf, který má chromatické číslo 1. U hvězdy S_5 dostaneme jako doplněk graf s šesti vrcholy, kde pět z nich tvoří úplný graf a jeden vrchol nemá žádného souseda. Takový graf má chromatické číslo 5. Pro Petersenův graf je jeho doplněk T_5, který má $\chi(T_5) = 5$ (T_5 viz Obrázek 5.6). Nakonec doplněk k C_7 je na obrázku 5.7 a opět probírkou všech možností zjistíme, že chromatické číslo je 4.

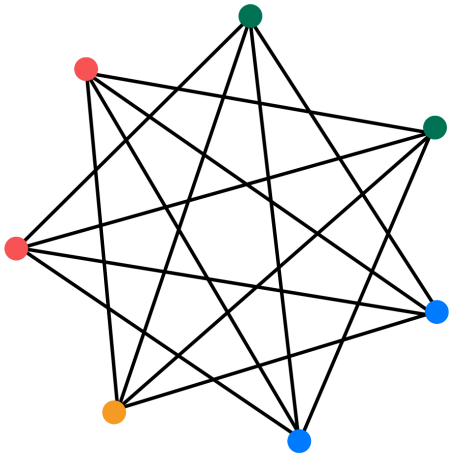
$\vartheta(G)$ pro vybrané grafy

Dříve jsme ukázali, že $\vartheta(C_5) = \sqrt{5} \approx 2.2361$. Navíc pro liché n [14] platí, že

$$\vartheta(C_n) = \frac{n \cdot \cos(\frac{\pi}{n})}{1 + \cos(\frac{\pi}{n})}. \tag{5.2}$$



Obrázek 5.6: Triangular graf T_5.



Obrázek 5.7: Obarvení $\bar{C}_7$.

Dostáváme tedy, že $\vartheta(C_7) \approx 3.3177$. Zbylé hodnoty plynou ze sendvičové věty. V tabulce 5.1 jsou shrnuty všechny zmíněné hodnoty.

Co víme o $\Theta(G)$

Shannonova kapacita je známá jen pro několik málo grafů. V [18] Shannon dal dolní odhad na $\Theta(C_5)$ a až za 23 let Lovász ukázal pomocí konstrukce, kterou jsme si ukázali výše, že $\Theta(C_5) = \sqrt{5}$. Ve stejném článku je důkaz, že kapacita Petersenova grafu $GP_{5,2}$ je 4. Triviální případy S_5 a K_5 dostaneme ze sendvičové věty, tj. $\Theta(S_5) = 5$ a $\Theta(K_5) = 1$. Naopak pro C_7 hodnotu

G	$\alpha(G)$	$\vartheta(G)$	$\chi(\bar{G})$
C_5	2	2.2361	3
C_7	3	3.3177	4
$GP_{5,2}$	4	4	5
K_5	1	1	1
S_5	5	5	5

Tabulka 5.1: $\alpha(G)$, $\vartheta(G)$, $\chi(\bar{G})$ pro vybrané grafy.

Θ neznáme. Máme dolní odhad $\alpha(C_7) = 3$ a horní odhad $\vartheta(C_7) \approx 3.3177$. Lepší dolní odhad, než dává $\alpha(C_7)$, ukázali Polak a Schrijver [17] tak, že pomocí počítače našli nezávislou množinu v grafu C_7^5 velikosti 367. Z čehož dostaneme dolní odhad $\sqrt[5]{367} \approx 3.2579$. Hodnota $\Theta(C_7)$ je tedy někde mezi

$$3.2578 < \Theta(C_7) \leq 3.3177.$$

Poznamenejme, že vylepšit dolní odhad na $\Theta(C_7)$ je výpočetně náročná úloha. Už pro C_7^4 se pomocí formulace

$$\begin{aligned} &\max \sum_{i=1}^{n} x_i \\ &\forall ij \in E: \ x_i + x_j \leq 1 \\ &\forall i \in V: \ x_i \in \{0, 1\} \end{aligned}$$

nenajde užitečná nezávislá množina, která by měla velikost alespoň 108 [22]. K výpočtu byl použit framework **Gurobi** a program po 7 měsících našel pouze nezávislou množinu velikosti 102, která dává pouze dolní odhad $\sqrt[4]{102} \approx 3.1779$.

5.5 Experimenty

Porovnáme formulace SDP1 a SDP2, které byly naimplementovány ve frameworku **Mosek**, na lichých kružnicích s přesnou hodnotou a dále určíme hodnoty ϑ pro náhodné grafy řádu 30.

5.5.1 Liché kružnice

V tabulce 5.2 jsou napočítané hodnoty ϑ funkce pro liché kružnice pomocí formulací SDP1 a SDP2. Tučně jsou zvýrazněny číslice, které se shodují s přesnou hodnotou určenou ze vztahu 5.2.

m	$\vartheta(C_n)$	SDP1	SDP2
5	2.236067977	**2.23606**8103	**2.2360679**78
7	3.317667207	**3.317667**960	**3.317667207**
9	4.360089581	**4.360089**596	**4.360089**606
11	5.386302911	**5.386302**951	**5.3863029**14
13	6.404168562	**6.404168**697	**6.404168562**
15	7.417148247	**7.41714**9582	**7.417148247**
...	...	...	...
121	110.494417456	nedoběhlo	**110.494417456**

Tabulka 5.2: Porovnání implementací ϑ funkce na lichých kružnicích.

Jednodušší model SDP2 dává přesnější výsledky a je schopen upočítat i větší úlohy. Nepřesnosti jsou dány tím, že solver v modelu SDP1 má mnohem více omezení a dvě proměnné $X \succeq 0, t \geq 0$. Největší problém představuje omezení

$$\forall i \in V(G) : \; x_{0i} \geq t.$$

V implementaci se totiž od sebe odečítají dvě proměnné a to v tomto frameworku není doporučovaná operace.

5.5.2 Náhodné grafy řádu 30

U větších grafů s nepravidelnou strukturou se rozdíly prohlubují a je vidět, že ani pro úplný graf, který je v tabulce 5.3 uveden na posledním řádku, není hodnota u SDP1 přesná.

m	SDP1	SDP2
0	30.000000007	30.0
43	16.000005889	17.690577212
87	11.151050632	14.385152039
130	9.0000004168	11.557666814
174	7.0892127153	9.9856302253
217	6.1569911550	9.5632864871
261	5.0000131024	7.6828991181
304	4.0323146793	6.8219576402
348	3.2171989901	5.1733914919
391	3.0000021103	4.0513834517
435	1.0000007203	1.0

Tabulka 5.3: Výsledky implementací ϑ funkce na náhodných grafech řádu 30 s daným počtem hran.

Kapitola 6

Problém maximálníhu řezu

6.1 Formulace úloh

Mějme neorientovaný graf $G = (V, E)$ s nezáporným ohodnocením hran w. Cílem je rozložit množinu vrcholů V na nejvýše $k \geq 2$ disjunktních množin $V_1, \ldots, V_k$ tak, aby součet vah hran vedoucí mezi různými množinami byl maximální. Pokud $k = 2$ hovoříme o úloze **MAX CUT** a pro $k \geq 3$ o úloze **MAX k-CUT**. Máme-li navíc předepsané maximální počty vrcholů v jednotlivách množinách, tj.

$$|V_1| \leq s_1, \ldots, |V_k| \leq s_k,$$

kde $|V| = n \leq \sum_i s_i$, jedná se o kapacitní MAX k-CUT úlohu, kterou budeme značit **CMAX k-CUT**.

6.2 Úloha MAX CUT

Nejprve se podíváme na aproximační algoritmus z článku [9] pro úlohu MAX CUT. Bylo čerpáno také z [21].

6.2.1 Striktní kvadratický program pro MAX CUT

Kvadratický program je problém optimalizace kvadratické funkce celočíselných proměnných, vzhledem ke kvadratickým omezením těchto proměnných. Je-li navíc každý monom (jednočlen) cenové funkce i daných omezení stupně 0 nebo 2, potom mluvíme o **striktním kvadratickém programu**.

Pomocí striktního kvadratického programu můžeme formulovat úlohu MAX CUT. Postup je následující. Nechť $y_i \in \{1, -1\}$ je proměnná příslušná vr-

cholu i. Množiny S a $\bar{S}$ definujeme tak, že

$$S = \{i \in V \mid y_i = 1\} \text{ a } \bar{S} = \{i \in V \mid y_i = -1\}.$$

Pokud $i \in S$ a $j \in \bar{S}$, potom je součin $y_i y_j = -1$ a chceme, aby tato hrana přispívala hodnotou w_{ij} k cenové funkci. Ve zbylých dvou možnostech je $y_i y_j = 1$ a chceme, aby se hodnota cenové funce nezměnila. Dostáváme následující striktní kvadratický program.

$$\begin{aligned} OPT = &\max \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - y_i y_j) \\ &\forall i \in V : \ y_i^2 = 1, \\ &\forall i \in V : \ y_i \in \mathbb{Z}. \end{aligned} \tag{SQ-MAX-CUT}$$

6.2.2 Vektorový program pro MAX CUT

Poznamenejme jen, že úloha celočíselného programování je NP-těžká. Proto se dále budeme zabývat relaxací úlohy SQ-MAX-CUT, což znamená, že upustíme od podmínek celočíselnosti a původní úlohu aproximujeme vektorovým programem. Modifikujeme tedy program SQ-MAX-CUT tak, že každý součin $y_i y_j$ nahradíme skalárním součinem vektorů $\langle v_i, v_j \rangle$ v $\mathbb{R}^n$. Dostáváme následující vektorový program.

$$\begin{aligned} RELAX = &\max \frac{1}{2} \sum_{1 \le i < j \le n} w_{ij}(1 - \langle v_i, v_j \rangle) \\ &\forall i \in V : \ \langle v_i, v_i \rangle = 1, \\ &\forall i \in V : \ v_i \in \mathbb{R}^n. \end{aligned} \tag{V-MAX-CUT}$$

6.2.3 Semidefinitní program pro MAX CUT

Vektorový program V-MAX-CUT je ekvivalentní s příslušným semidefinitním programem. Nechť W je vážená matice sousednosti grafu G a w_{ij} je váha hrany ij, kde $i < j$. Matice J je matice $n \times n$ samých jedniček.

$$\begin{aligned} RELAX = &\max \frac{1}{4} \langle W, J - Y \rangle \\ &\forall i \in V : \ y_{ii} = 1, \\ &Y \succeq 0. \end{aligned} \tag{SDP-MAX-CUT}$$

6.2.4 Aproximační algoritmus

Vyřešením programu SDP-MAX-CUT dostaneme optimální řešení Y^*. Matice je samozřejmě pozitivně semidefinitní. Provedeme rozklad

$$Y^* = LL^T,$$

kde řádky matice L jsou optimální řešení vektorového programu V-MAX-CUT. Označme i-tý řádek matice L jako a_i. Dále budeme chtít nějakým způsobem separovat vektory, které jsou od sebe „daleko“ a shlukovat ty, které jsou „blízko“.

Označme Θ_{ij} úhel, který svírají vektory a_i a a_j. Z podmínky

$$\forall i \in V : \ \langle v_i, v_i \rangle = 1$$

dostáváme, že $\langle a_i, a_j \rangle = \cos \Theta_{ij}$ a příspěvek těchto vektorů k optimálnímu řešení je

$$\frac{w_{ij}}{2}(1 - \cos \Theta_{ij}).$$

Tedy čím „blíž“ je úhel Θ_{ij} hodnotě π, tím větší příspěvek mají tyto vektory k hodnotě optimálního řešení. Algoritmus je následující.

Algoritmus 1 (MAX-CUT)**.** *[9]*

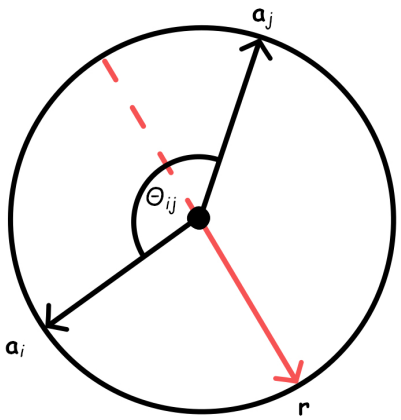
1. *Najdi řešení $a_1, \ldots, a_n$ programu V-MAX-CUT.*
2. *Zvol náhodně vektor r na jednotkové sféře S_{n-1}.*
3. *$S = \{i \in V \mid \langle a_i, r \rangle \geq 0\}$.*

Dále se budeme snažit objasnit kroky **1** a **2** v algoritmu 1.

Lemma 5. *[22] Nechť X_{ij} je jev takový, že vrcholy i a j jsou od sebe separovány, tj. jsou v různých množinách. Potom*

$$P[X_{ij}] = \frac{\Theta_{ij}}{\pi}.$$

Lemma 6. *[13] Nechť $x = (x_1, \ldots, x_n)$ je vektor, jehož prvky jsou zvoleny nezávisle z normálního normovaného rozdělení $\mathcal{N}(0, 1)$. Potom $r = \frac{x}{\|x\|}$ je náhodný vektor, který leží na jednotkové sféře S_{n-1}.*



Obrázek 6.1: Separace vrcholů i, j náhodným vektorem r.

Lemma 6 nám dává postup, kterým provedeme bod **2** v algoritmu 1. Nyní ukážeme, jak „dobrou“ aproximaci algoritmem 1 dostaneme. Označme

$$\alpha = \min_{0 \leq \Theta \leq \pi} \frac{2\Theta}{\pi(1 - \cos\Theta)}.$$

Snadno se ukáže, použitím derivace, že $\alpha \approx 0.87856$.

Lemma 7. *[9] Nechť Y je náhodná veličina, která označuje součet vah hran, které vedou z S do $\bar{S}$, nalezeny algoritmem 1. Potom*

$$E[Y] \geq \alpha \cdot RELAX.$$

Důkaz. Z definice čísla α, pro $0 \leq \Theta \leq \pi$, dostáváme

$$\frac{\Theta}{\pi} = \frac{2\Theta}{\pi(1 - \cos\Theta)} \cdot \frac{1 - \cos\Theta}{2} \geq \frac{\alpha}{2}(1 - \cos\Theta). \tag{6.1}$$

Použitím lemmatu 5 a nerovnosti 6.1 dostáváme

$$\begin{aligned} E[Y] &= \sum w_{ij} P[X_{ij}] \\ &= \sum w_{ij} \frac{\Theta_{ij}}{\pi} \\ &\geq \frac{\alpha}{2} \sum w_{ij}(1 - \cos\Theta_{ij}) \\ &= \alpha \cdot RELAX. \end{aligned}$$

□

Poznamenejme, že samozřejmě platí

$$OPT \geq E[Y] \geq \alpha \cdot RELAX. \tag{6.2}$$

Mezeru celočíselnosti relaxace (pro maximalizační problém) definujeme jako

$$\inf_I \frac{OPT(I)}{RELAX(I)},$$

kde infimum probíhá přes všechny instance I daného programu (pro minimalizační problém by se čitatel a jmenovat prohodily). Ze vztahu 6.2 dostáváme, že mezera celočíselnosti relaxace V-MAX-CUT je alespoň $\alpha \approx 0.87856$. Číslu α ze 6.2 se říká **aproximační poměr** a algoritmus 1 je tedy 0.87856-aproximační algoritmus.

Předchozí odvození je založeno na střední hodnotě náhodné veličiny Y. Proto kroky **2** a **3**, v algoritmu 1, opakujeme vícekrát a jako výsledek zvolíme množinu S, která dává největší součet hran z S do $\bar{S}$. Dále jen specifikujeme kolikrát musíme tyto kroky opakovat. Kompletní odvození je v [21]. Zvolíme tedy $\varepsilon > 0$ (malé), nechť

$$c = \frac{\varepsilon\alpha}{2 + 2\varepsilon - \alpha},$$

a kroky **2**, **3** opakujeme $\lceil \frac{1}{c} \rceil$-krát.

6.3 Úloha MAX k-CUT

V této části shrneme semidefinitní (vektorové) formulace s aproximačními schématy z několika článků pro úlohu MAX k-CUT.

6.3.1 Frieze-Jerrum a MAX k-CUT

Začneme článkem [5]. Uvažme rovnostranný simplex Σ_k v $\mathbb{R}^{k-1}$ s vrcholy $b_1, b_2, \ldots, b_k$. Nechť $c = (b_1 + \cdots + b_k)/k$ je těžiště Σ_k a nechť $a_i = b_i - c$, kde $i = 1, \ldots, k$. Dále předpokládejme, že délka strany Σ_k je taková, že $\|a_i\| = 1$.

Lemma 8. *[5] Pro $i \neq j$, platí*

$$\langle a_i, a_j \rangle = -\frac{1}{k-1}.$$

Nyní můžeme formulovat úlohu MAX k-CUT následovně:

$$\begin{aligned} &\max \frac{k-1}{k} \sum_{1 \leq i < j \leq n} w_{ij}(1 - \langle y_i, y_j \rangle) \\ &y_i \in \{a_1, \ldots, a_k\}. \end{aligned} \tag{FJ}$$

Poznamenejme, že

$$1 - \langle y_i, y_j \rangle = \begin{cases} 0 & y_i = y_j, \\ k/(k-1) & y_i \neq y_j. \end{cases}$$

K získání vektorové relaxace programu FJ nahradíme vektor y_i vektorem v_i, kde v_i je vektor na S_{n-1}.

$$\begin{aligned} &\max \frac{k}{k-1} \sum_{1 \leq i < j \leq n} w_{ij}(1 - \langle v_i, v_j \rangle) \\ &\forall i \in V : \ \langle v_i, v_i \rangle = 1, \\ &\forall i \neq j \in V : \ \langle v_i, v_j \rangle \geq -\frac{1}{k-1}, \\ &\forall i \in V : \ v_i \in \mathbb{R}^n. \end{aligned} \qquad \text{(FJ-RELAX)}$$

Máme řešení $a_1, \ldots, a_n$ programu FJ-RELAX. Zvolíme k náhodných vektorů $z_1, \ldots, z_k$ na jednotkové sféře S_{n-1}. Pro každý vrchol $i \in V$ určíme k skalárních součinů $\langle a_i, z_1 \rangle, \ldots, \langle a_i, z_k \rangle$ a vrchol i přidáme do množiny V_j, jestliže $j = \arg\max \{\langle a_i, z_l \rangle \mid l = 1, \ldots, k\}$. Použitím tohoto postupu dostáváme následující algoritmus.

Algoritmus 2 (FJ MAX k-CUT)**.** *[5]*

1. *Najdi řešení $a_1, \ldots, a_n$ programu FJ-RELAX.*
2. *Zvol náhodně k vektorů $z_1, \ldots, z_k$ na jednotkové sféře S_{n-1}.*
3. *Pro každý vrchol $i \in V$ určit k skalárních součinů $\langle a_i, z_1 \rangle, \ldots, \langle a_i, z_k \rangle$.*
4. *Vrchol i přidej do množiny V_j, jestliže $j = \arg\max \{\langle a_i, z_l \rangle \mid l = 1, \ldots, k\}$.*

6.3.2 Goemans-Williamson a MAX 3-CUT

Algoritmus vycházející z článku [10] využívá komplexní semidefinitní programování, tj. každý prvek je reprezentován komplexním vektorem. Následující

vektorový program je jeho relaxací, viz [16].

$$
\begin{aligned}
&\max \frac{2}{3} \sum_{1 \le i < j \le n} w_{ij}(1 - \langle v_i^1, v_j^1 \rangle) \\
&\forall i \in V \ \forall a, b \in \{1,2,3\}, a \neq b : \ \langle v_i^a, v_i^b \rangle = -\frac{1}{2}, \\
&\forall i, j \in V \ \forall a, b, c \in \{1,2,3\} : \ \langle v_i^a, v_i^b \rangle = \langle v_i^{a+c}, v_i^{b+c} \rangle \qquad \text{(GW-RELAX)} \\
&\forall i, j \in V \ \forall a, b \in \{1,2,3\} : \ \langle v_i^a, v_j^b \rangle \ge -\frac{1}{2} \\
&\forall i \in V \ \forall a \in \{1,2,3\} : \ \langle v_i^a, v_i^a \rangle = 1 \\
&\forall i \in V \ \forall a \in \{1,2,3\} : \ v_i^a \in \mathbb{R}^{3n}
\end{aligned}
$$

Mějme $3n$ vektorů, které tvoří řešení GW-RELAX. Pro vrchol $i \in V$ leží vektory v_i^1, v_i^2, v_i^3 ve stejné rovině tak, že jsou otočeny o $\frac{2\pi}{3}$. Nejprve zvolíme vektor $g \in \mathbb{R}^{3n}$ takový, že každá složka je vybrána nezávisle z normálního normovaného rozdělení $\mathcal{N}(0,1)$. Pro každý vrchol $i \in V$ určíme projekci vektoru g do příslušné roviny. Odtud dostaneme úhel $\theta_i \in \langle 0, 2\pi)$ pro každý vrchol. Náhodně zvolíme úhel $\psi \in \langle 0, \pi)$ a vrchol $i \in V$ přidáme do množiny V_j, jestliže

$$\theta_i \in \psi + \frac{j2\pi}{3}, j \in \{0,1,2\},$$

kde úhly počítáme modulo 2π. Dostáváme algoritmus pro MAX 3-CUT.

Algoritmus 3 (GW MAX 3-CUT)**.** *[10]*

1. *Najdi řešení $a_1^1, a_1^2, a_1^3, \ldots, a_n^3$ programu GW-RELAX.*
2. *Zvol náhodně vektor $g \in \mathbb{R}^{3n}$ tak, že každá složka je vybrána nezávisle z normálního normovaného rozdělelní $\mathcal{N}(0,1)$.*
3. *Pro každý vrchol $i \in V$ urči projekci vektoru g do příslušné roviny a vypočítej úhel θ_i, který svírá projekce g a vektor a_i^3.*
4. *Zvol libovolně úhel $\psi \in \langle 0, \pi)$.*
5. *Vrchol i přidej do množiny V_j, jestliže $\theta_i \in \psi + \frac{j2\pi}{3}, j \in \{0,1,2\}$, kde úhly počítáme modulo 2π.*

6.3.3 Newman a MAX k-CUT

Cílem [16] je rozšířit přístup, pomocí komplexního semidefinitního programování, z [10] pro MAX 3-CUT na libovolné $k \geq 3$. Využívá se formulace FJ-RELAX, jejíž vyřešením dostaneme vektory $a_1, \ldots, a_n$. Pro každý vrchol $i \in V$ definujeme dva ortonormální vektory v $\mathbb{R}^{2n}$ tak, že

$$u_i = (a_i, 0) \text{ a } u_i^{\perp} = (0, a_i)\,.$$

Dále zvolíme náhodný vektor $g \in \mathbb{R}^{2n}$, kde každá složka je vybrána náhodně z normálního normovaného rozdělení $\mathcal{N}(0,1)$. Pro každý vrchol $i \in V$ určíme projekci vektoru g na 2-dimenzionální disk

$$\{u_i(\theta) \mid \theta \in \langle 0, \pi)\}\,,$$

kde $u_i(\theta) = u_i \cos\theta + u_i^{\perp} \sin\theta, \theta \in \langle 0, \pi)$ a určíme úhel mezi projekcí vektoru g a vektorem u_i. Nakonec náhodně zvolíme úhel $\psi \in \langle 0, 2\pi)$ a vrchol $i \in V$ přidáme do množiny V_j, jestliže

$$\theta_i \in \psi + \frac{j2\pi}{k}, j \in \{0, 1, \ldots, k-1\}\,,$$

kde úhly počítáme modulo 2π.

Algoritmus 4 (N1 MAX k-CUT)**.** *[16]*

1. *Najdi řešení $a_1, \ldots, a_n$ programu FJ-RELAX.*
2. *Pro každý vrchol $i \in V$ urči vektory $u_i = (a_i, 0)$ a $u_i^{\perp} = (0, a_i)$ v $\mathbb{R}^{2n}$.*
3. *Zvol náhodně vektor $g \in \mathbb{R}^{2n}$ tak, že každá složka je vybrána nezávisle z normálního normovaného rozdělení $\mathcal{N}(0,1)$.*
4. *Pro každý vrchol $i \in V$ urči úhel θ_i.*
5. *Zvol libovolně úhel $\psi \in \langle 0, 2\pi)$ a vrchol $i \in V$ přidej do množiny V_j, jestliže $\theta_i \in \psi + \frac{j2\pi}{k}, j \in \{0, 1, \ldots, k-1\}$, kde úhly počítáme modulo 2π.*

V závěru článku je navrhnuté ještě jedno aproximační schéma. Shrneme ho v následujícím algoritmu.

Algoritmus 5 (N2 MAX k-CUT)**.** *[16]*

1. *Najdi řešení $a_1, \ldots, a_n$ programu FJ-RELAX.*

2. *Zvol náhodně $k-1$ vektorů $g_1, \ldots, g_{k-1} \in \mathbb{R}^n$ tak, že každá složka je vybrána nezávisle z normálního normovaného rozdělení $\mathcal{N}(0,1)$.*

3. *Vygeneruj rovnostranný simplex Σ_k se středem v počátku.*

4. *Pro každý vrchol $i \in V$ urči vektor $s_i = (\langle g_1, a_i \rangle, \ldots, \langle g_{k-1}, a_i \rangle) \in \mathbb{R}^{k-1}$.*

5. *Každý vektor (a tedy i vrchol) přiřaď k nejbližšímu vrcholu simplexu Σ_k.*

6.3.4 de Klerk-Pasechnik-Warners a MAX k-CUT

Jako poslední ještě zmíníme algoritmus z [4], ve kterém nejprve vyřešíme následující semidefinitní program pro $\vartheta(\bar{G})$.

$$\begin{aligned} &\min t \\ &\forall ij \in E: \; U_{ij} = -\frac{1}{t-1} \\ &\forall i \in V: \; U_{ii} = 1 \\ &U \succeq 0, k \geq 2. \end{aligned} \qquad \text{(THETA-}\bar{G}\text{)}$$

Dostaneme optimální řešení $(U, \vartheta(\bar{G}))$, kde matici U použijeme k určení matice Y.

$$Y = U \otimes \frac{k}{k-1}\left(I_k - \frac{1}{k} e_k e_k^T\right),$$

kde I_k je jednotková matice $k \times k$ a e_k je vektor samých jedniček v $\mathbb{R}^k$. Rozkladem $Y = V^T V$ získáme matici $V = \begin{bmatrix} v_1^1 & v_1^2 & \ldots & v_1^k & \ldots & v_n^k \end{bmatrix}$. Zvolíme náhodný vektor $g \in \mathbb{R}^{kn}$ na sféře S_{kn-1} a určíme vektor x tak, že

$$x_i^p = \begin{cases} 1 & g^T v_i^p = \max\left\{\langle g, v_i^q \rangle \mid q = 1, \ldots, k\right\}, \\ -1 & \text{jinak.} \end{cases}$$

Vektor $i \in V$ jsme přiřadili do množiny V_j, jestliže $x_i^j = 1$.

6.3.5 Aproximační poměry

V tabulce 6.1, která je převzána z [16], jsou uvedeny aproximační poměry pro jednotlivé algoritmy.

k	MAX CUT	FJ MAX k-CUT	GW MAX 3-CUT	dKPW [4]	N1 MAX k-CUT
2	.878956				
3		.832718	.836008	.836008	
4		.850304		.857487	.846478
5		.874243		.876610	.862440
10		.926642		.926788	.915885

Tabulka 6.1: Aproximační poměry pro MAX k-CUT.

6.4 Úloha CMAX k-CUT

O obecné úloze CMAX k-CUT toho není mnoho známo. V [8] (a později v opravě [23]) je popsán algoritmus, který využívá lokální prohledávání, jehož aproximační poměr je

$$\frac{t(k-1)}{2(s-1)+t(k-1)},$$

kde $k \geq 2$, $t = \min_{i=1,\ldots,k} |V_i|$ a $s = \max_{i=1,\ldots,k} |V_i|$. Dále vyzkoušíme pro úlohu CMAX k-CUT přístup, ve kterém využijeme semidefinitní programování.

6.4.1 Lokální prohledávání

Nechť $G = (V, E)$ je neorientovaný graf s n vrcholy. Označíme $w(u, v)$ váhu hrany uv. Na vstupu máme zadáno $k \geq 2$ a maximální počty vrcholů v jednotlivých množinách

$$|V_1| \leq s_1, \ldots, |V_k| \leq s_k.$$

Algoritmus začíná **inicializací**, kde libovolně rozdělíme vrcholy do k množin tak, aby $|V_i| \leq s_i, i = 1, \ldots, k$.

V **iteračním kroku** hledáme dvojici vrcholů $u \in V_i$ a $v \in V_j, i \neq j$, pro který

$$\sum_{x \in V_i, x \neq u} w(u,x) + \sum_{x \in V_j, x \neq v} w(v,x) > \sum_{x \in V_j, x \neq u} w(u,x) + \sum_{x \in V_i, x \neq v} w(v,x) - 2w(u,v).$$

Pokud takové vrcholy najdeme, tak vrchol u přesuneme do množiny V_j a vrchol v přesuneme do množiny V_i a iterační krok opakujeme. Jestliže takové vrcholy neexistují algoritmus končí.

6.4.2 CMAX k-CUT pomocí SDP

V našem přístupu pro CMAX k-CUT použijeme algoritmus 5, ze kterého potřebujeme rozklad vrcholů do množin $V_1, \ldots, V_k$, vygenerovaný simplex

a $k-1$ náhodných vektorů pomocí, kterých byl určen rozklad vrcholů do množin $V_1, \ldots, V_k$.

Máme zadány kapacity $s_1 \geq \cdots \geq s_k$. Množiny, které dostaneme z algoritmu 5, přeznačíme tak, aby součet záporných rezerv $r_i = s_i - |V_i|$ množin $V_1, \ldots, V_k$ byl maximální.

Pro každou množinu $V_1, \ldots, V_k$ určíme její rezervu r_i. Pokud jsou všechny rezervy nezáporné, vrátíme množiny $V_1, \ldots, V_k$ jako výsledek. Jinak z množiny, která má zápornou rezervu vybereme vrchol, který má nejkratší vzdálenost do nějakého jiného „volného vrcholu“ simplexu Σ_k. Takto iterujeme dokud nejsou všechny rezervy nezáporné.

6.5 Experimenty

K implementaci algoritmů je použit framework **Mosek**. Jsou naimplementovány algoritmy 2, 3, 4, 5 pro úlohu MAX k-CUT a pro kapacitní verzi také algoritmus využívající lokální prohledávání a náš přístup využívající semidefinitní programování. Experimenty jsou realizovány na náhodných grafech řádu 30 s předepsaným počtem hran. Tučně jsou vždy vyznačeny nejlepší výsledky.

6.5.1 MAX k-CUT

Pro úlohu MAX 3-CUT byl otestován algoritmus 2, algoritmus 3 a algoritmus 5, viz tabulka 6.2. Srovnatelné jsou algoritmy 2 a 5. Algoritmus 3, který by teoreticky měl vycházet nejlépe (viz tabulka 6.1 s aproximačními poměry), je nejhorší. V semidefinitní formulaci, kterou využívá tento algoritmus je velké množství omezení a velikost úlohy (optimalizační proměnná) je třikrát větší než u formulace FJ-RELAX. Problémy s přesností byly indikovány tím, že při výpočtu úhlu pro vrchol na příslušném disku, skalární součin dvou vektorů, které by měly být jednotkové, vždy neležel v intervalu $\langle -1, 1\rangle$ a bylo potřeba zaokrouhlovat.

Pro úlohy MAX 4-CUT a MAX 5-CUT byl otestován algoritmus 2, algoritmus 4 a algoritmus 5, viz tabulky 6.3 – 6.4. Zde nejlépe vychází algoritmus 5, pro který není zatím známý aproximační poměr (aproximační schéma je jen zmíněno v závěru článku [16]).

6.5.2 CMAX k-CUT

Pro úlohu CMAX k-CUT jsou porovnány dva algoritmy. Jeden využívající lokální prohledávání a druhý přístup, který je založen na semidefinitním pro-

	FJ MAX k-CUT		GW MAX 3-CUT		N2 MAX k-CUT	
m	100 iterací	10000 iterací	100 iterací	10000 iterací		
43	41	**43**	37	40	41	**43**
87	80	**82**	72	77	80	81
130	116	**117**	104	115	114	**117**
174	150	**153**	140	147	152	152
217	178	**182**	166	175	178	**182**
261	208	**210**	194	203	206	**210**
304	239	**241**	226	240	237	**241**
348	263	**265**	251	261	263	**265**
391	287	**289**	281	287	288	**289**

Tabulka 6.2: MAX 3-CUT na náhodných grafech řádu 30.

	FJ MAX k-CUT		N1 MAX k-CUT		N2 MAX k-CUT	
m	100 iterací	10000 iterací	100 iterací	10000 iterací	100 iterací	10000 iterací
43	**43**	**43**	39	42	42	**43**
87	84	**87**	76	81	85	**87**
130	125	**129**	114	119	127	**129**
174	163	**164**	149	157	161	**164**
217	195	201	185	188	197	**202**
261	229	**233**	215	225	230	**233**
304	265	267	249	258	263	**268**
348	289	**294**	277	287	292	**294**
391	321	**323**	312	319	319	**323**

Tabulka 6.3: MAX 4-CUT na náhodných grafech řádu 30.

gramování. Byly provedeny 3 experimenty, viz tabulky 6.5 – 6.7. Jeden pro CMAX 3-CUT s kapacitami $(10, 10, 10)$ a dva pro CMAX 4-CUT s kapacitami $(8, 8, 8, 8)$ a $(20, 20, 20, 20)$. Výsledky obou algoritmů jsou srovnatelné, ale je nutné poznamenat, že výpočet využívající lokální prohledávání zabere mnohem méně strojového času.

	FJ MAX k-CUT		N1 MAX k-CUT		N2 MAX k-CUT	
m	100 iterací	10000 iterací	100 iterací	10000 iterací	100 iterací	10000 iterací
43	**43**	**43**	40	**43**	**43**	**43**
87	**87**	**87**	78	84	**87**	**87**
130	128	**130**	119	123	128	**130**
174	167	**170**	156	160	167	**170**
217	206	208	194	199	204	**209**
261	242	**244**	229	234	**244**	243
304	276	**282**	263	270	279	**282**
348	309	311	294	302	307	**312**
391	337	**343**	328	336	339	342

Tabulka 6.4: MAX 5-CUT na náhodných grafech řádu 30.

	lokální prohledávání		SDP	
m	nejhorší	nejlepší	nejhorší	nejlepší
43	40	**43**	34	**43**
87	77	**82**	73	**82**
130	110	**117**	105	**117**
174	142	**153**	142	**153**
217	174	**182**	175	**182**
261	202	**210**	203	**210**
304	231	**241**	235	**241**
348	258	**265**	259	**265**
391	282	**289**	284	**289**

Tabulka 6.5: CMAX 3-CUT na náhodných grafech řádu 30 s kapacitami $(10, 10, 10)$, 100 iterací.

	lokální prohledávání		SDP	
m	nejhorší	nejlepší	nejhorší	nejlepší
43	**43**	**43**	37	**43**
87	83	**87**	77	85
130	120	**129**	118	**129**
174	158	**166**	151	165
217	190	**201**	189	200
261	226	**233**	225	**233**
304	258	**268**	257	267
348	289	**295**	288	**295**
391	316	322	319	**323**

Tabulka 6.6: CMAX 4-CUT na náhodných grafech řádu 30 s kapacitami $(8, 8, 8, 8)$, 100 iterací.

	lokální prohledávání		SDP	
m	nejhorší	nejlepší	nejhorší	nejlepší
43	37	**43**	42	**43**
87	73	**87**	82	85
130	102	**129**	126	**129**
174	140	**166**	159	165
217	165	200	195	**201**
261	204	232	228	**233**
304	224	266	263	**268**
348	253	294	291	**295**
391	278	320	320	**323**

Tabulka 6.7: CMAX 4-CUT na náhodných grafech řádu 30 s kapacitami $(20, 20, 20, 20)$, 100 iterací.

Závěr

V prvních čtyřech kapitolách je shrnuta teorie, která se dále využívá při popisu úloh kombinatorické optimalizace.

V kapitole o Shannonově kapacitě se věnujeme Lovászově ϑ funkci a jejímu využití. Byly naimplementovány a následně porovnány dva modely s přesnou hodnotou pro liché kružnice. Jednodušší model, který dává rovnou hodnotu ϑ funkce byl přesnější. U druhého modelu, ze kterého dostaneme hodnotu $1/\sqrt{\vartheta}$, byly patrné chyby už pro malé grafy. Problémem je, že úloha není ve standardním tvaru a frameworky typu Mosek jsou optimalizovány hlavně na práci s úlohami ve standardním tvaru. Dále byl puštěn výpočet pro vylepšení dolního odhadu Shannonovy kapacity kružnice C_7, ale nezávislá množina, která by odhad vylepšila se nenašla.

Pro úlohu MAX k-CUT byly pro účely testování implementovány čtyři algoritmy a pro úlohu kapacitního MAX k-CUTu další dva. U experimentu pro úlohu MAX 3-CUT vyšel nejhůře algoritmus 3, který má nejlepší aproximační poměr, což je to dáno složitostí modelu založeného na komplexním semidefinitním programování, který je třikrát větší než model použitý v ostatních algoritmech a navíc obsahuje hodně omezení (vznikaly zaokrouhlovací chyby). Pro MAX 4-CUT a MAX 5-CUT vyšel nejlépe algoritmus 5, který je jen navrhnut v závěru článku [16] a není pro něj znám aproximační poměr. U kapacitní verze byly porovnány dva přístupy: algoritmus založený na lokálním prohledávání a náš algoritmus využívající semidefinitní programování. Oba algoritmy dávaly srovnatelné výsledky, ale rozdíl byl hlavně v době běhu, kde jasně vyhrává algoritmus využívající lokální prohledávání. Algoritmus založený na semidefinitní programování dával lepší výsledky, když součet kapacit jednotlivých množin byl větší než počet vrcholů grafu.

Dále by mohla být věnována pozornost analýze algoritmu 5, o kterém není známé nic.

Celá práce i s implementacemi je dostupná na `https://github.com/c0n73x7/D1PL0MK4`.

Literatura

[1] G. Blekherman, P. A. Parrilo, and R. R. Thomas. *Semidefinite Optimization and Convex Algebraic Geometry*. Siam, 2013.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2013.

[3] E. de Klerk. *Aspects of Semidefinite Programming*. Kluwer Academic Publishers, 2004.

[4] E. de Klerk, D. V. Pasechnik, and J. P. Warners. On approximate graph colouring and max-k-cut algorithms base od the ϑ-function. *Journal of Combinatorial Optimization*, 2004.

[5] A. Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica*, 1995.

[6] B. Gärtner and J. Matoušek. *Understanding and Using Linear Programming*. Springer, 2007.

[7] B. Gärtner and J. Matoušek. *Approximation Algorithms and Semidefinite Programming*. Springer, 2012.

[8] D. R. Gaur, R. Krishnamurti, and R. Kohli. The capacited max k-cut problem. *Mathematical Programming*, 2008.

[9] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for max cut and max 2sat. *Journal of the ACM*, 1994.

[10] M. X. Goemans and D. P. Williamson. Approximation algorithms for max-3-cut and other problems via complex semidefinite programming. *Journal of Computer and System Sciences*, 2001.

[11] R. D. Hill and S. R. Waters. On the cone of positive semidefinite matrices. *Linear Algebra and its Applications*, 1987.

[12] D. E. Knuth. The sandwich theorem. *The Electronic Journal of Combinatorics*, 1993.

[13] D. E. Knuth. *The Art of Computer Programming.* Addison-Wesley, third edition, 1997.

[14] L. Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information Theory*, 1979.

[15] J. Matoušek. *Lectures on Discrete Geometry.* Springer, 2002.

[16] A. Newman. Complex semidefinite programming and max-k-cut. 2018.

[17] S. Polak and A. Schrijver. New lower bound on the shannon capacity of c_7 from circular graphs. *Information Processing Letters*, 2018.

[18] C. E. Shannon. The zero error capacity of a noisy crannel. *IRE Transactions on Information Theory*, 1956.

[19] B. Singh and M. Eisner. A brief history of optimization and mathematical programming. `https://www.informs.org/Explore/History-of-O.R.-Excellence/O.R.-Methodologies/Optimization-Mathematical-Programming`.

[20] G. Strang. *Linear Algebra and Learning from Data.* Wellesley Cambridge Press, 2019.

[21] V. V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[22] A. Vesel and J. Žerovnik. Improved lower bound on the shannon capacity of c_7. *Information Processing Letters*, 2002.

[23] Jinglan Wu and Wenxing Zhu. On a local search algorithm for the capacited max-k-cut problem. *Kuwait Journal of Science*, 2014.

[24] R. Čada. Přednášky z předmětu kombinatorická optimalizace.