SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 819

RELAKSACIJE SEMIDEFINITNOG PROGRAMIRANJA ZA PROBLEME BOJANJA GRAFOVA I MAKSIMALNE KLIKE

Lana Šprajc

Zagreb, lipanj, 2025.

Ovdje dolazi tekst zadatka diplomskog rada na hrvatskom jeziku.

Hvala Branimiru na podsjetnicima da trebam pisati diplomski...

Sadržaj

1. Uvod **3**

2. Opis problema **4**

2.1. Bojanje grafova 4

2.1.1. $\delta + 1$ bojanje grafa 4

2.2. Traženje maksimalne klike 4

2.2.1. Pohlepno traženje lokalno maksimalne klike 5

3. Semidefinitno programiranje **6**

3.1. Teorija dualnosti 6

3.2. Metode rješavanja SDP 7

3.2.1. Metoda unutarnje točke 7

4. Lovászov broj **8**

5. Primjene SDP-a u bojanju grafova **9**

5.1. Važnost vektorskog bojanja grafa 9

5.2. Formulacija SDP 10

5.3. Pretvorba vektorskog bojanja u bojanje grafa 11

5.3.1. Semibojanja 11

5.3.2. Metoda odsijecanja ravninama 11

5.3.3. Zaokruživanje vektorskim projekcijama 12

5.4. Wigdersonova tehnika 12

6. Primjene SDP-a u traženju maksimalne klike **13**

6.1. SDP formulacija problema maksimalnog stabilnog skupa 13

7. Programska ostvarenja **15**
7.1. Bojanje grafova 15
7.1.1. Primjer bojanja grafa 16
7.2. Traženje maksimalne klike 18
7.2.1. Primjer traženja maksimalne klike 18
8. Rezultati i rasprava **21**
8.1. Bojanje grafova 21
8.2. Pronalazak maksilne klike 23
9. Upute za prevođenje i pokretanje **25**
9.1. Upute za prevođenje 25
9.2. Glavni program 25
9.3. Program za vizualizaciju 26
10. Zaključak **27**
Literatura **28**
Sažetak **30**
Abstract **31**

1. Uvod

U teoriji grafova, problemi bojanja grafa i traženja maksimalne klike poznati su NP-teški problemi s brojnim primjenama. Bojanje grafa svakom vrhu grafa pridružuje jednu boju na način da su svaka dva susjedna vrha različitih boja. Svaki planar graf je moguće obojati u četiri boje [1]. Poznate primjene ovog problema su alociranje registara [2], raspoređivanje zadataka [3]...

Problem traženja maksimalne klike je problem pronalaska potpuno povezanog podgrafa. Problem ima česte primjene u kemiji i bioinformatici, primjerice za pronalaženje sličnih struktura u molekulama [4].

Za oba ova problema često je dovoljno pronaći rješenje koje je blizu optimalnom. Semidefinitno programiranja daje dobre aproksimacije rješenja problema koje se mogu izračunati u polinomnoj složenosti.

Rad je podijeljen u 10 poglavlja. U drugom poglavlju dan je opširniji opis problema bojanja grafova i traženja maksimalne klike kao i opis jednostavnih metoda rješavanja ovih problema. U trećem poglavlju dan je pregled semidefinitnog programiranja i metoda rješavanja takvih problema. U četvrtom poglavlju opisana je važnost Lovászovog broja i SDP formulacija za pronalazak njegove vrijednosti. U petom poglavlju opisan je algoritam za bojanje grafova uz pomoć semidefinitnog programiranja. U šestom poglavlju opisano je rješavanje problema pronalaska maksimalne klike uz primjenu semidefinitnog programiranja. U sedmom poglavlju dan je pregled programskih ostvaranja i opis rada programa kroz primjere. U osmom poglavlju prikazani su rezultati rada i mogućnosti daljnjeg rada u ovom području. U devetom poglavlju dane su upute za prevođenje i pokretanje programa. U desetom poglavlju iznesen je zaključak.

2. Opis problema

Graf se sastoji od n vrhova i m bridova. Pišemo $G = (V, E)$. Stupanj vrha je definiran kao broj vrhova s kojima je taj vrh susjedan. δ je oznaka za najveći stupanj vrha.

2.1. Bojanje grafova

Bojanje grafa je postupak pridruživanja boje svakom vrhu grafa tako da su susjedni vrhovi različitih boja. Graf je k-obojiv ako postoji bojanje grafa s k ili manje boja. Kromatski broj $\chi(G)$ grafa je najmanji broj boja potreban za obojati graf. U većini primjena, dovoljno je obojati graf s dovoljno malim brojem boja $k > \chi(G)$.

2.1.1. $\delta + 1$ bojanje grafa

$\delta + 1$ je jednostavan algoritan bojanja grafa koji boja graf s najviše $\delta + 1$ bojom. Algoritam je složenosti $O(n^2)$. Neka je $\{1, \dots, \delta + 1\}$ skup boja. Svakom vrhu pridružuje se najmanji broj kojim nije obojan ni jedan od njegovih susjeda. Uvijek će postojati barem jedna takva boja jer svaki vrh ima najviše δ susjeda, a postoji $\delta + 1$ boja. Problem ovog algoritma je što je u većini slučajeva kromatski broj grafa puno manji od $\delta + 1$.

2.2. Traženje maksimalne klike

Klika u grafu je potpuno povezani podgraf. Maksimalna klika (engl. *maximum clique*) je najveći takav podgraf. Broj vrhova u maksimalnoj kliki označavamo s $\omega(G)$. Ovaj broj predstavlja donju granicu kromatskog broja grafa, $\omega(G) \leq \chi(G)$.

Lokalno maksimalna klika (engl. *maximal clique, inclusion-maximal*) je klika koja nije podgraf veće klike. Takvoj kliki nije moguće dodati ni jedan vrh grafa koji nije dio klike, na način da dobiveni podgraf ostaje klika.

2.2.1. Pohlepno traženje lokalno maksimalne klike

Jednostavan algoritam za traženje lokalno maksimalne klike započinje proizvoljno odabranim vrhom, tj. skupom $S = \{v\}$. Algoritam zatim iterira kroz preostale vrhove grafa i za svaki vrh w provjerava čini li podgraf $S \cup \{w\}$ kliku. Ako čini, vrh w dodaje se u skup S; u suprotnom se odbacuje.

3. Semidefinitno programiranje

Semidefinitno programiranje (krat. *SDP*) klasa je optimizacijskih problema. Standardni oblik problema je:

$$\begin{aligned} minf &= \langle CX \rangle \\ t.d.\langle A_i X \rangle &= b_i \\ X &= X^T \succeq 0 \end{aligned} \tag{3.1}$$

pri čemu $\langle A \rangle = \sum_i^n a_{i,i}$ označava trag kvadratne matrice. $X \succeq 0$ označava da je matrica pozitivno semidefinitna.

Dualni problem zadan je jednadžbama:

$$\begin{aligned} min \sum_i^m b_i y_i \\ t.d. \sum_i^m A_i y_i + Z = C \\ Z \succeq 0 \end{aligned} \tag{3.2}$$

Semidefinitno programiranje ima primjene u rješavanju problema makimalnog reza (engl. *max cut problem*) [5], u rješavanju SAT problema [6] ...

3.1. Teorija dualnosti

U optimizacijskim problemima, razmak dualnosti (engl. *duality gap*) je razlika između optimuma primarnog p^* i dualnog d^* problema, $p^* - d^*$. Kažemo da vrijedi slaba dualnost ako je razmak dualnosti veći ili jednak od 0. Jaka dualnost vrijedi ako su optimum primarnog i dualnog problema jednaki, to jest ako je razmak dualnosti jednak 0 [7].

Za SDP razmak dualnosti jednak je:

$$
\begin{aligned}
\langle CX \rangle - b^T y &= \langle (\sum_{i=1}^{m} A_i y_i + Z) X \rangle - b^T y \\
&= \langle ZX \rangle + \sum_{i=1}^{m} (\langle A_i X \rangle - b_i) y_i \\
&= \langle ZX \rangle \geq 0
\end{aligned}
\tag{3.3}
$$

Iz ovoga je vidljivo da za SDP vrijedi slaba dualnost, ali ne nužno i jaka. Prema Slaterovom uvjetu, jaka dualnost je zadovoljena ako postoji rješenje primarnog problema u kojem je zadovoljen uvjet $X \succ 0$ i rješenje dualnog problema takvo da vrijedi $Z \succ 0$ [8].

3.2. Metode rješavanja SDP

Postoje razne metode rješavanja SDP-a, primjerice elipsoidna metoda i metoda unutarnje točke. Ove metode numerički pronalaze rješenje, obično u polinomnoj složenosti.

3.2.1. Metoda unutarnje točke

Kao što ime kaže, metoda unutarnje točke započinje točkom unutar prostora koji zadovoljava uvjete semidefinitnog programa i postupno se priližava optimumu. Algoritam definira zamjensku funkciju za optimizaciju $f^*(x) = f(x) + b(x)$, gdje je $b(x)$ takozvana granična funkcija. $b(x)$ teži beskonačnosti kako se x približava granici dopuštenog prostora, dok unutar prostora problema ima malu vrijednost blizu 0. Optimizacijski algoritmi izbjegavaju rubni prostor problema i pronalaze optimum unutar dozvoljenog prostora.

4. Lovászov broj

Lovászov broj (Lovászova theta funkcija) rješenje je sljedećeg semidefinitnog programa:

$$\begin{aligned} \theta &= \max\langle JX\rangle \\ t.d.\langle X\rangle &= 1 \\ x_{i,j} &= 0, \forall(i,j) \in E(G) \\ X &= X^T \succeq 0 \end{aligned} \tag{4.1}$$

Ovdje J predstavlja $n \times n$ matricu ispunjenu jedinicama.

Za Lovászov broj, $\theta(G)$ vrijedi da je veći od ili jednak broju klike, a manji od ili jednak kromatskom broju grafa, $\omega(G) \leq \theta(G) \leq \chi(G)$. [9]

Ovaj program se pretvora u standardnu formu tako da se postavi $C = -J$, $A_1 = I$, $b_1 = 1$, $A_i = E_{j,k}, \forall(j,k) \in E(G)$, $b_i = 0, i \in \{2, \dots, m\}$. Ovdje $E_{i,k}$ predstavlja matricu koja sadrži 0 na svim mjestima osim na $e_{i,j}$ i $e_{j,i}$ gdje sadrži 1.

5. Primjene SDP-a u bojanju grafova

U formulaciji bojanja grafova pomoću SDP-a, vrhovima se pridružuju jedinični vektoru. Vektori pridruženi susjednim vrhovima moraju biti udaljeni jedan od drugog.

Definicija 1 *(iz [10]) Za graf G=(V, E) s n vrhova i m bridova i realni broj $k \geq 1$, vektorsko k-bojanje je pridruživanje jediničnih vektora $v_i \in \mathfrak{R}^n$ svakom vrhu i grafa G takvih da za svaka dva susjedna vrha $v_{i,j} \in E$ vrijedi:*

$$v_i \cdot v_j \leq -\frac{1}{k-1} \tag{5.1}$$

Definicija 2 *(iz [10]) Za graf G=(V, E) s n vrhova i m bridova i realni broj $k \geq 1$, matrično k-bojanje je određivanje simetrične pozitivno semidefinitne matrice M, takve da je $m_{i,i} = 1, i \in \{1, \dots, n\}$ i $m_{i,j} = m_{j,i} \leq -\frac{1}{k-1}, (i, j) \in E$.*

Matrično k-bojanje i vektorsko k-bojanje su ekvivalentni. Ako je poznato vektorsko k-bojanje, matrica M određuje se tako da se element $m_{i,j}$ postavi na skalarni produkt vektora v_i i v_j. Ako je poznato matrično k-bojanje, matricu M moguće je faktorizirati pomoću dekompozicije svojstvenim vrijednosti (engl. *eigendecomposition*) tako da vrijedi $M = VV^T$. Sada su stupci matrice V jedinični vektori koji čine vektorsko k-bojanje grafa. [10]

5.1. Važnost vektorskog bojanja grafa

Teorem 5..1 *Za $k \leq n$, postoji k jediničnih vektora $v_i \in \mathfrak{R}^n$ takvih da je $v_i \cdot v_j = -\frac{1}{k-1}, \forall i \neq j$*

Dokaz *(iz [10]) Dovoljno je dokazati da ovo vrijedi za k = n, ako je k manji od n $v_i[j]$*

se postavlja na 0 za sve $j > k$.

Konstruirani su vektori $v_1, \dots, v_k$ na sljedeći način:

$$v_i[j] = \begin{cases} -\sqrt{\frac{1}{k(k-1)}} & j \neq i \\ \sqrt{\frac{k-1}{k}} & j = i \end{cases} \tag{5.2}$$

Teorem 5..2 *Svaki k-obojiv graf je i vektorski k-obojiv.*

***Dokaz** (iz [10]) Graf je uvijek obojiv s $k \leq n$ boja jer je u najgorem slučaju svaki vrh obojan u različitu boju. Svakom vrhu obojanom s i-tom bojom, pridružuje se vektor v_i iz 5..1*

Iz ovog teorema vidljivo je da je minimalni k za koji je graf vektorski k-obojiv donja granica za kromatski broj grafa.

5.2. Formulacija SDP

Slijedi SDP formulacija vektorskog bojanja grafa koja minimizira k za koji je graf vektorski k-obojiv.

$$\begin{aligned} &\min \alpha \\ t.d.&x_{ii} = 1 \\ &x_{ij} \leq \alpha, \forall (i,j) \in E(G) \\ &X = X^T \succeq 0 \end{aligned} \tag{5.3}$$

Optimalni α jednak je $-\frac{1}{k-1}$. Kako bi problem bio u standardnoj formi, u matrici X uvodi se novi element na dijagonalu tako da $x_{n+1,n+1} = \alpha$. Dodatno, uvode se *slack* varijable za uvjete povezane s bridovima. Konačna formulacija problema u standardnoj formi je:

$$
\begin{aligned}
&\min\langle E_{n+1,n+1}X\rangle \\
t.d.&\langle E_{ii}X\rangle = 1 \\
&\langle E_{ij}X\rangle - 2\langle E_{n+1,n+1}X\rangle + \langle E_{n+1+k,n+1+k}X\rangle = 0, \forall(i,j) \in E(G), k \in \{1,\dots,m\} \\
&X = X^T \succeq 0
\end{aligned}
\tag{5.4}
$$

5.3. Pretvorba vektorskog bojanja u bojanje grafa

Rješavanjem SDP-a dobiva se vektorsko bojanje grafa koje je potrebno pretvoriti u bojanje grafa. Postoji nekoliko algoritma za ovaj problem. Neki od njih su metoda odsijecanja ravninama (engl. *cutting plane method*) i zaokruživanje vektorskim projekcijama.

5.3.1. Semibojanja

Definicija 3 *k-semibojanje (engl. k-semicoloring) grafa G je pridruživanje k boja barem polovici vrhova grafa G tako da dva susjedna vrha nisu jednake boje.*

Ako je poznato k-semibojanje grafa, tada je moguće obojati cijeli graf u $k \log n$ boja. Ovo je moguće rekurzivnim bojanjem barem polovice preostalog grafa, sve dok ne ostanu jedan ili nijedan neobojan vrh.

5.3.2. Metoda odsijecanja ravninama

Motivacija algoritma je činjenica da su vektori koji predstavljaju susjedne vrhove udaljeni u prostoru, to jest velik je kut između njih. Generira se N slučajnih hiperravnina koje prolaze kroz ishodište koordinatnog sustava. One dijele prostor na $2N$ dijelova. Svakom dijelu je pridružena jedna boja te su svi vektori u tom prostoru obojani tom bojom. Ukoliko su dva susjedna vrha obojana istom bojom, jednom od tih vrhova treba maknuti boju.

Vjerojatnost da su 2 susjedna vrha s različitih strana jedne hiperravnine je β/Π, gdje je β kut između vektorskih reprezentacija tih vektora. Za veći broj hiperravnina, smanjuje se vjerojatnost da su vrhovi u jednakim podprostorima. Budući da susjedni vektori imaju velike kuteve između vektorskih reprezentacija, oni imaju manju vjerojatnost da se nalaze u jednakim podprostorima. Promijenom parametra N bira se između točnog bojanja većeg broja vrhova za veći N i bojanja s manjim brojem boja za manji N. [10]

5.3.3. Zaokruživanje vektorskim projekcijama

U ovom algoritmu, traže se veliki podgrafi u kojima ni jedan vrh nije međusobno povezan. Zadovoljavajuće je pronaći podgraf s N vrhova i m bridova, gdje je $N >> m$ te izbaciti m vrhova koji su imaju susjedne vrhove u podgrafu. Ovih $N-m$ vrhova možemo obojati jednom bojom.

Algoritam 1 *Bojanje grafa vektorskim projekcijama*

1. *generiranje slučajnog vektora* ***u*** *po standardnoj distribuciji i normaliziracija*
2. *ako je* $|v_i \cdot u| > \epsilon$*, dodavanje vrha* ***i*** *u trenutni skup vrhova*
3. *provjera nalaze li se bridovi u dobivenom podgrafu i izbacivanje potrebnih vrhova iz skupa*
4. *bojanje trenutnog skupa vrhova u jednu boju*
5. *ponavljanje dok nisu obojani svi vrhovi*

Intuitivno, ako je jedan od vektora v_i blizu slučajnom vektoru u, tada su njegovi susjedni vektori v_j koji su udaljeni od vektora v_i, vjerojatno udaljeni i od vektora u. Promijenom parametra ϵ utječemo na broj obojenih vektora u jednom koraku, ali i broj *uhvaćenih* bridova.

5.4. Wigdersonova tehnika

Algoritmi bojanja grafova mogu se poboljšati korištenjem Wigdersonove tehnike. Algoritam određuje konstantu β i ako postoji vrh stupnja većeg od β, on i njegovi susjedi bojaju se s 2 boje koje se dalje više ne koriste. [11] Ovo smanjuje maksimalni stupanj bojanog podgrafa, a broj boja koje algoritmi koriste ovisi o maksimalnom stupnju grafa pa se primjenom ove tehnike smanjuje taj broj.

6. Primjene SDP-a u traženju maksimalne klike

Stabilan skup (engl. *stable set*) je podskup vrhova grafa G koji ne sadrži dva susjedna vrha. Inverzni graf $\bar{G}$ grafu G sastoji se od jednakog skupa vrhova kao i graf G. Vrhovi v_i i v_j u grafu $\bar{G}$ su povezani ako i samo ako nisu povezani u grafu G.

Skup vrhova S je klika u grafu G ako i samo ako je stablian skup u grafu $\bar{G}$. Dakle, problem maksimalne klike grafa G ekvivalentan je problemu maksimalnog stabilnog skupa grafa $\bar{G}$. [12]

U nastavku je opisana metoda pronalska maksimalnog stabilnog skupa.

6.1. SDP formulacija problema maksimalnog stabilnog skupa

Svakom vrhu grafa pridjeljuje se vrijednost -1 ili 1. Jedan od ovih skupova predstavlja stabilan skup. Grafu se dodaje jedan vrh v_{n+1} koji nije povezan ni s jednim vrhom grafa G. Ovaj vrh se očito nalazi u maksimalnom stabilnom skupu. Vrh služi prepoznavanju koji od skupova je stabilan skup.

Problem je definiran na sljedeći način (iz [12]):

$$\begin{aligned} &\max \frac{1}{2}\sum_{i=1}^{n}(v_i^2 + v_{n+1}v_i) \\ t.d.\, &v \in \{-1, 1\}^{n+1} \\ &|v_i + v_j + v_{n+1}| = 1 \forall (i, j) \in E \end{aligned} \tag{6.1}$$

Izraz $v_i^2 + v_{n+1}v_i$ je jednak 0 ako je v_i različitog predznaka kao i v_{n+1} ili jednak 2 ako je v_i jednakog predznaka kao i v_{n+1}. Što znači da je vrijednost optimizacijske funkcije jednaka broju vrhova u maksimalnom stabilnom skupu.

Problem se relaksira u SDP na sljedeći način (iz [12]):

$$\begin{aligned} &\max\left\langle \begin{bmatrix} .5 & & & .25 \\ & \ddots & & \vdots \\ & & .5 & .25 \\ .25 & \cdots & .25 & 0 \end{bmatrix} X \right\rangle \\ t.d.diag(X) &= e \\ \langle (e_{ijk} e_{ijk}^T) X \rangle &= 1 \\ X = X^T &\succeq 0 \end{aligned} \tag{6.2}$$

Iz relaksiranog problema dobiva se stabilni skup prateći sljedeći algoritam:

Algoritam 2 *Određivanje maksimalnog stabilnog skupa*

1. *Iz optimalnog* X^* *dobivenog iz 6.2 određuje se* V *za koji vrijedi* $X^* = VV^T$
2. *Za slučajan jedinični vektor* $u \in \Re^{n+1}$ *računa se* $v = sign(Vu)$
3. *Za svaki* $(i, j) \in E$, *ako ne vrijedi* $|v_i + v_j + v_{n+1}| = 1$, *mijenja se prednjak od* v_i *ili* v_j, *onom kojem je vrijednost* $V_k u$ *udaljenija od* $V_{n+1} u$

Stabilan skup vrhova je onaj skup vrhova koji imaju jednaki predznak kao i v_{n+1}.

7. Programska ostvarenja

U sklopu rada implementiran je algoritam za bojanje grafova i traženje maksimalne klike. Implementiran je program za vizualizaciju grafa i program za generiranje slučajnih grafova.

Sav kod napisan je u C++-u uz korištenje biblioteke DSDP [13] i OpenBLAS implementacije BLAS/LAPACK paketa [14].

Definirana je klasa `Graph` koja učitava graf spremljen u datoteci u obliku matrice susjedstva. U ovom zapisu, jedinica predstavlja da postoji brid između dva vrha, dok nula predstavlja da vrhovi nisu povezani. Definirane su metode `color` koja boja graf algoritmom zaokruživanja vektorskim projekcijama, `greedy_color` koja boja graf $\delta+1$ algoritmom, `find_max_clique` koja pronalazi veliku kliku SDP relaksacijom i `greedy_clique` koja pronalazi lokalno maksimalnu kliku.

Kod je moguće pronaći na [15].

7.1. Bojanje grafova

Implementirana su dva algoritma za bojanje grafova: SDP relaksacijom uz zaokruživanje vektorskim projekcijama i $\delta + 1$ algoritmom.

$\delta + 1$ algoritam je implementiran kao što je objašnjeno u poglavlju 2.1.1.

Za bojanje zaokruživanjem vektorskih projekcijama, u problemu 5.3 uvjeti nejednakosti pretvoreni su u jednakosti kako bi se ubrzao rad programa. Problem je pretvoren u standardnu formu na sljedeći način:

$$
\begin{aligned}
&\min\langle E_{n+1,n+1}X\rangle \\
t.d.&\langle E_{ii}X\rangle = 1 \\
&\langle E_{ij}X\rangle - 2\langle E_{n+1,n+1}X\rangle = 0, \forall(i,j) \in E(G) \\
&X = X^T \succeq 0
\end{aligned}
\tag{7.1}
$$

Problem 7.1 riješen je korištenjem biblioteke DSDP [13]. Korištenjem ove biblioteke dobivena je optimalna matrica X koja je faktorizirana u oblik VV^T korištenjem OpenBLAS-a [14].

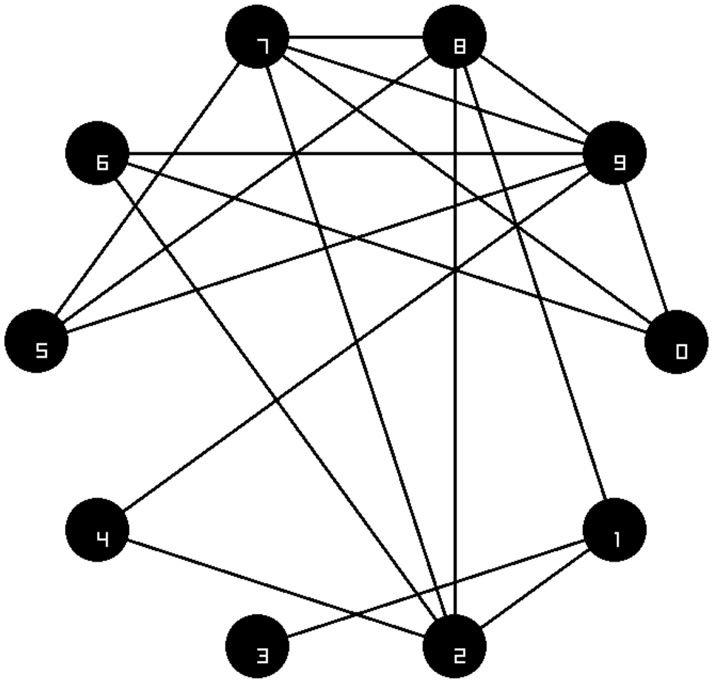
Graf je dalje obojan kako je opisano u poglavlju 5.3.3. Nije korištena Wigdersonova tehnika.

7.1.1. Primjer bojanja grafa

Zaokruživanje vektorskim projekcijama objašnjeno je na grafu G s tablicom susjedstva 7.2 Na slici 7.1. prikazan je graf G s oznakama vrhova $0, \ldots, 9$.

$$
G = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\tag{7.2}
$$

U tablici 7.1. prikazano je vektorsko bojanje grafa G koje se dobiva rješavanjem pripadajućeg SDP-a. U tablici 7.2. prikazani su slučajni vektori generirani u postupku bojanja grafa G zaokruživanjem vektorskim projekcijama. U tablici 7.3. prikazani su skalarni umnošci iterativno generiranih slučajnih vektora s vektorima pripadajućih vrhova grafa



Slika 7.1. Prikaz grafa G

G. Vrijednost parametra $\epsilon = 0.0$. U prvom koraku vrhovi 6, 7 i 8 imaju skalarni produkt $> \epsilon$. Budući da su vrhovi 7 i 8 susjedni, vrh 8 nije obojan. Na slici 7.2.a, prikazan je graf nakon prve iteracije bojanja.

U drugom koraku, vrhovi 2, 3, 4, 8 i 9 imaju skalarni produkt $> \epsilon$. Vrhovi 4 i 8 nisu obojani jer postoje bridovi (2, 4), (4, 9) i (8, 9). Slika 7.2.b, prikazuje graf G nakon druge iteracije bojanja.

U trećoj iteraciji, vrhovi 0, 1, 4 i 8 imaju pozitivni skalarni produkt. Vrh 8 nije obojan zato što postoji brid (1, 8). Na slici 7.2.c prikazan je graf G nakon trećeg koraka algoritma bojanja grafa.

Ostali su vrhovi 5 i 8 koji su povezani pa su oni obojani različitim bojama u sljedeće dvije iteracije algoritma. Na slici 7.2.d prikazan je potpuno obojan graf G.

Tablica 7.1. Vektorsko bojanje grafa G

i	$v_{i,0}$	$v_{i,1}$	$v_{i,2}$	$v_{i,3}$	$v_{i,4}$	$v_{i,5}$	$v_{i,6}$	$v_{i,7}$	$v_{i,8}$	$v_{i,9}$
0	0.000	0.226	-0.253	0.113	-0.340	0.0680	0.286	0.342	0.627	0.401
1	0.000	-0.191	-0.174	0.475	-0.024	0.026	0.387	0.410	-0.364	-0.506
2	0.000	-0.174	-0.387	-0.276	0.069	-0.003	-0.232	0.153	-0.300	0.756
3	0.000	-0.021	-0.043	0.518	-0.041	0.273	-0.657	-0.327	0.218	0.259
4	0.000	0.037	-0.126	-0.126	0.539	0.570	0.084	0.075	0.404	-0.419
5	-0.010	-0.042	0.207	0.070	0.204	-0.116	-0.198	0.806	0.278	0.358
6	0.000	0.151	-0.236	0.057	0.258	-0.564	-0.355	0.056	0.046	-0.634
7	-0.010	0.017	-0.076	-0.223	-0.425	0.270	-0.301	0.107	-0.244	-0.728
8	-0.010	-0.174	-0.087	-0.003	0.006	-0.218	0.257	-0.591	0.705	-0.049
9	-0.010	0.192	-0.048	0.158	0.216	0.062	0.245	-0.320	-0.741	0.416

Tablica 7.2. Generirani slučajni vektori r_i

i	$r_{i,0}$	$r_{i,1}$	$r_{i,2}$	$r_{i,3}$	$r_{i,4}$	$r_{i,5}$	$r_{i,6}$	$r_{i,7}$	$r_{i,8}$	$r_{i,9}$
0	0.057	-0.219	0.846	0.061	0.000	-0.252	-0.052	-0.254	0.039	-0.312
1	-0.556	-0.436	0.246	-0.042	0.191	0.479	0.225	-0.270	0.124	0.186
2	-0.260	0.172	-0.272	0.320	0.250	0.643	0.159	-0.395	0.261	-0.039
3	0.195	-0.327	0.190	-0.060	-0.415	-0.060	-0.229	0.603	0.460	-0.105
4	-0.011	0.019	0.732	0.432	-0.015	-0.312	0.146	-0.000	0.289	0.274

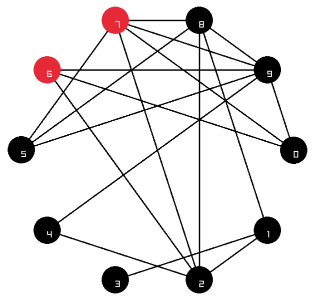
7.2. Traženje maksimalne klike

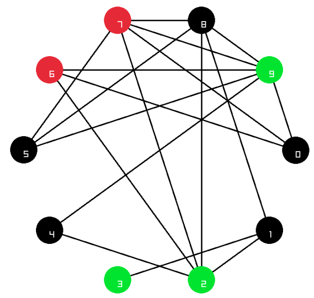
U sklopu rada implementirane su dvije metode traženja maksimalne klike grafa: pohlepno traženje lokalno maksimalne klike kao što je opisano u poglavlju 2.2.1. i SDP relaksacijom definiranom u 6.2 pomoću algoritma 2 Za graf je određen njegov inverzni graf kojem se određuje stabilni skup koji je zapravo klika u početnom grafu.

7.2.1. Primjer traženja maksimalne klike

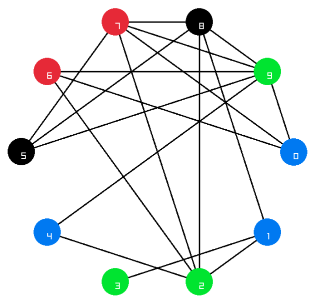
Na slici 7.3. prikazana je klika dobivena algoritmom 2 za graf G prikaz na slici 7.1. Klika sadrži vrhove 5, 7, 8 i 9. Ovo klika je ujedno i maksimalna klika grafa G.

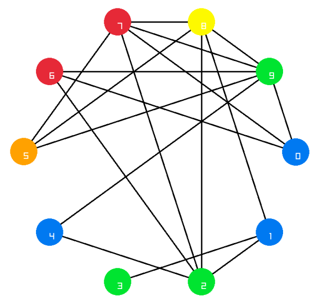
U tablici 7.4. prikaza je matrica V koja je dobivena faktorizacijom optimalnog rješenje SDP-a 6.2 za graf G. Prema algoritnu 2 generira se slučajan vektor r koju je zapisan u tablici 7.5. Zatim se računa umnožak matrice V i vektora r te se određuje predznak svakog elementa vektora umnoška. Ovi predznaci su zapisani u tablici 7.6. Za graf G elementi na mjestima 5, 7, 8 i 9 imaju jednaki predznak (-1) kao i dodatan element 10 te oni čine potencijalnu kliku. Provjerom vidimo da svi potrebni bridovi postoje te dobivamo kliku koja je prikazana na slici 7.3.





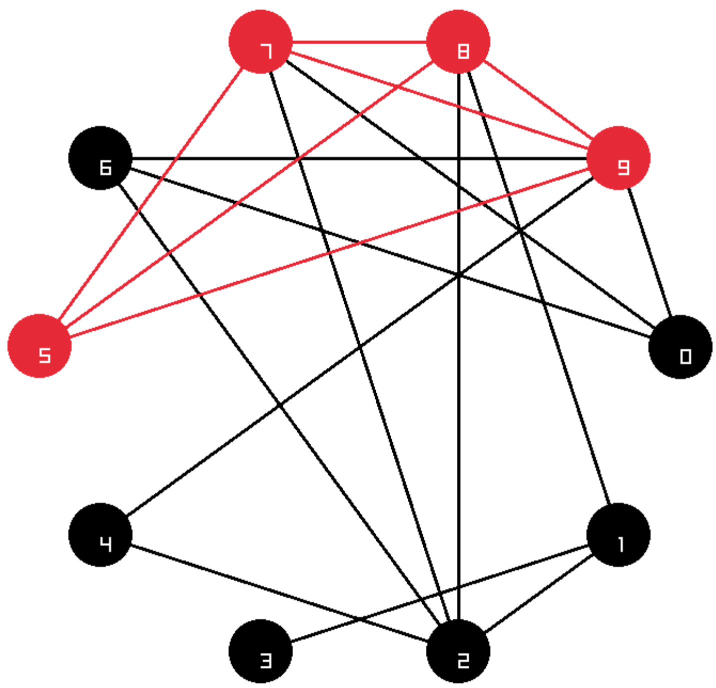
(a) Graf G nakon prve iteracije bojanja **(b)** Graf G nakon druge iteracije bojanja





(c) Graf G nakon trece iteracije bojanja **(d)** Konačno bojanje grafa G

Slika 7.2. Koraci bojanja grafa G



Slika 7.3. Klika graf G

Tablica 7.3. Umnošci slučajnih vektora s vektorima vrhova

i	$r_i \cdot v_0$	$r_i \cdot v_1$	$r_i \cdot v_2$	$r_i \cdot v_3$	$r_i \cdot v_4$	$r_i \cdot v_5$	$r_i \cdot v_6$	$r_i \cdot v_7$	$r_i \cdot v_8$	$r_i \cdot v_9$
0	-0.476	-0.063	-0.580	-0.024	-0.143	-0.078	0.117	0.056	0.199	-0.179
1	-0.073	-0.135	0.014	0.115	0.304	-0.107			0.252	0.102
2	0.161	0.002			0.607	-0.353			0.316	
3						0.591			-0.037	
4									0.228	

Tablica 7.4. Faktorizacija V optimalnog rješenja SDP relaksacije za graf G

i	$V_{i,0}$	$V_{i,1}$	$V_{i,2}$	$V_{i,3}$	$V_{i,4}$	$V_{i,5}$	$V_{i,6}$	$V_{i,7}$	$V_{i,8}$	$V_{i,9}$	$V_{i,10}$
0	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
1	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
2	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
3	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
4	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
5	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	-1.000	1.000	1.000	1.000	1.000
6	1.000	1.000	1.000	1.000	1.000	-1.000	1.000	-1.000	-1.000	-1.000	-1.000
7	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	-1.000	1.000	1.000	1.000	1.000
8	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	-1.000	1.000	1.000	1.000	1.000
9	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	-1.000	1.000	1.000	1.000	1.000
10	-1.000	-1.000	-1.000	-1.000	-1.000	1.000	-1.000	1.000	1.000	1.000	1.000

Tablica 7.5. Slučajan vektor r

r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9	r_{10}
0.265	0.335	0.636	0.362	0.176	0.064	-0.016	0.132	0.049	0.290	-0.378

Tablica 7.6. Predznak umnoška matrice V i vektora r

$r \cdot v_0$	$r \cdot v_1$	$r \cdot v_2$	$r \cdot v_3$	$r \cdot v_4$	$r \cdot v_5$	$r \cdot v_6$	$r \cdot v_7$	$r \cdot v_8$	$r \cdot v_9$	$r \cdot v_{10}$
1	1	1	1	1	-1	1	-1	-1	-1	-1

8. Rezultati i rasprava

Za potrebe testiranja napravljen je jednostavan generator slučajnih grafova. Za definirane parametre $n \in \mathbb{N}$ i $p \in [0, 1]$, generira se graf s n vrhova. Bridovi se generiraju slučajno, za svaki brid je vjerojatnost generiranja p. Graf je spremljen u datoteku u obliku matrice susjedstva. Generiran je po jedan graf za sve $n \in \{5, 10, \dots, 40, 45\}$ i sve $p \in \{0, 0.05, \dots, 0.9, 0.95\}$.

8.1. Bojanje grafova

Svi generirani grafovi obojani su $\delta + 1$ algoritmom i algoritmom zaokruživanja vektorskim projekcijama uz $\epsilon \in \{-0.2, 0, 0.2\}$. Za algoritam zaokruživanja vektorskim projekcijama, rezultati su izračunati 3 puta te je uzeta najbolja vrijednost. Za svaki algoritam su izračunati prosjeci po parametru n koji su prikazani u tablici 8.1. Iz tablice vidimo da algoritam zaokruživanja vektorskim projekcijama daje najbolje rezultate za $\epsilon = -0.2$, a najgore za $\epsilon = 0.2$. Algoritam zaokruživanja vektorskim projekcijama za $\epsilon = -0.2$ i $\delta + 1$ bojanje daju otprilike jednake rezultate. Zanimljivo je primjetiti da $\delta + 1$ algoritam boja graf s puno manje od δ boja, pogotovo za veliki n.

U tablici 8.2., prikazani su prosječni rezultati po parametru p. Vidimo da je SDP relaksacija bolja od $\delta + 1$ bojanja za veliki p što je za očekivati jer u grafovima generiranima s većim p ima veći broj vrhova koji imaju visoki stupanj.

Dobiveni rezultati za algoritam zaokruživanja vektorskim projekcijama mogli bi se poboljšati korištenjem Wigdersonove tehnike. Implementacijom ove metode mogli bi se ostvariti bolji rezultati nego korištenjem $\delta + 1$ algoritma.

Tablica 8.1. Broj boja po algoritmu po n

n	$\delta + 1$ bojanje	SDP bojanje $\epsilon = -0.2$	SDP bojanje $\epsilon = 0$	SDP bojanje $\epsilon = 0.2$	δ
5	2.85	2.95	3.2	3.6	3.05
10	4.25	4.5	4.7	5.8	6.45
15	5.95	5.85	6.35	7.95	9.65
20	7.2	7.2	7.9	10.4	12.65
25	8.1	8.3	9.65	12.4	15.5
30	9.6	9.95	11.25	17.5	19
35	10.6	11.25	12.65	18.65	21.5
40	12	12.45	13.85	21.3	24.35
45	12.65	13.55	15	24.15	27

Tablica 8.2. Broj boja po algoritmu po p

p	$\delta + 1$ bojanje	SDP bojanje $\epsilon = -0.2$	SDP bojanje $\epsilon = 0$	SDP bojanje $\epsilon = 0.2$	δ
0	1	2.22222	4.44444	11.7778	0
0.05	2.55556	2.55556	4.11111	10.6667	3.66667
0.1	3.44444	3.33333	4.66667	10.4444	5
0.15	3.88889	3.66667	5	11.3333	7.55556
0.2	4.66667	4.44444	5.77778	10.3333	9.33333
0.25	5.44444	6	6.66667	11.3333	11.7778
0.3	10.8889	12	13.3333	22.6667	23.5556
0.35	6.44444	6.66667	7.66667	12.3333	13.2222
0.4	6.66667	7.11111	8.33333	12.1111	15.2222
0.45	7.22222	7.44444	8.66667	12.6667	16.1111
0.5	8	7.88889	9.33333	13.2222	16.6667
0.55	9	9	10.2222	13.4444	18.7778
0.6	9.66667	10.1111	10.3333	14.3333	19.2222
0.65	9.66667	10.1111	10.7778	14.6667	20.2222
0.7	10.3333	11	11.8889	14.7778	21.2222
0.75	11.2222	12.4444	12.3333	15.6667	22.3333
0.8	12.3333	12.6667	13.5556	16.3333	23.6667
0.85	13.4444	13.3333	14.1111	16.5556	23.7778
0.9	15.1111	14.7778	15.2222	18.3333	24.6667
0.95	17.1111	16.3333	16.7778	17.8889	25

Tablica 8.3. Veličina pronađene klike po n

n	pohlepni algoritam	SDP relaksacija
5	2.5	2.8
10	3.5	4.05
15	4.45	5
20	4.7	5.6
25	5.4	6.55
30	6	7.6
35	6.15	7.75
40	6.85	7.65
45	6.9	7.55

8.2. Pronalazak maksilne klike

Za sve generirane grafove određen su klike pohlepnim pronalaskom lokalno maksimalne klike i pronalaskom SDP relaksacijom. Algoritam SDP relaksacije pokrenut je 3 puta i uzeta je vrijednost maksimalne od pronađenih klika. Prosječne veličine pronađenih klike za svaki n prikazane su u tablici 8.3. Iz tablice je vidljivo da algoritam baziran na SDP relaksaciji pronalazi veće klike od pohlepnog, otprilike za jedan vrh više po kliki.

U tablici 8.4. prikazane su prosječne veličine pronađenih klika za različite vrijednosti parametra p. Za gotovo sve vrijednosti parametra, SDP relaksacijom se pronalaze veća klika nego pohlepnim algoritmom. Najveća razlika između algoritama je za vrijednosti $p \sim 0.5$.

Tablica 8.4. Veličina pronađene klike po p

p	pohlepni algoritam	SDP relaksacija
0	1	0.888889
0.05	1.77778	2.11111
0.1	2	2.44444
0.15	2.33333	2.77778
0.2	2.33333	2.77778
0.25	2.66667	4
0.3	5.33333	8
0.35	3.44444	4.33333
0.4	3.44444	4.44444
0.45	3.77778	5.33333
0.5	4.22222	5.22222
0.55	4.77778	6.11111
0.6	5.66667	5.66667
0.65	5.55556	7
0.7	6.11111	7.66667
0.75	6.55556	8.22222
0.8	9	8.88889
0.85	9.44444	11.1111
0.9	11.6667	12.6667
0.95	14.7778	15.5556

9. Upute za prevođenje i pokretanje

9.1. Upute za prevođenje

Prije prevođenja pretpostavlja se da su na računalu instalirani DSDP [13], OpenBLAS [14] i raylib. Definiran je `Makefile` pa je moguće prevođenje pomoću naredbe `make`. Ovime nastaju programi `.out/main.out`, `.out/visualize.out`, `.out/generate.out` i `.out/tester.out`.

9.2. Glavni program

Program `.out/main.out` glavni je program kojem se putem opcija definira željeni ispis na ekran te putanja do datoteke u kojoj je spremljen graf. Moguće opcije su ispis bridova grafa, ispis boja grafa obojanog $\delta + 1$ algoritmom, ispis boja grafa obojanog algoritmom zaokruživanja vektorskim projekcijama, ispis klike pronađene pohlepnim algoritmom i ispis klike pronađene semidefinitnom relaksacijom.

```
usage: main.out [options] file
options:
--help         display help
-print         print graph
-delta_color   color the graph with delta+1 algorithm
-color         color the graph with SDP algorithm
-greedy_clique find clique using greedy algorithm
-clique        find clique using SDP algorithm
```

(a) Primjer vizualizacije obojanog grafa   **(b)** Primjer vizualizacije klike grafa

Slika 9.1. Primjeri programa vizualizacije

9.3. Program za vizualizaciju

Program `.out/visualize.out` program je za vizualizaciju grafa. Prilikom pokretanja upisuju se opcije vizualizacije i ime datoteke u kojoj je spremljen graf. Moguća je vizualizacija obojanog grafa, ili $\delta + 1$ algoritmom ili algoritmom zaokruživanja vektorskim projekcijama. Također je moguće vizualizirati kliku grafa, pronađenu ili pohlepnim algoritmom ili SDP relaksacijom. Na slici 9.1. prikazani su primjeri vizualizacije bojanja i maksimalne klike grafova.

```
usage: visualize.out option file
options:
--help          display help
-delta_color    color the graph with delta+1 algorithm
-color          color the graph with SDP algorithm
-greedy_clique find clique using greedy algorithm
-clique         find clique using SDP algorithm
```

10. Zaključak

U sklopu rada, dan je pregled problema bojanja grafova i pronalaska maksimalne klike grafa.

Implementirano je bojanje grafova pomoću SDP relaksacije uz algoritam zaokruživanja vektorskim projekcijama i pomoću $\delta + 1$ bojanja. Oba algoritma u polinomnom vremenu ispravno bojaju graf, no $\delta + 1$ algoritam eksperimentalno daje bolje rezultate. U danjem radu, bilo bi moguće poboljšati bojanje grafa algoritmom zaokruživanja uz primjenu Wigdersonove tehnike.

Mogle bi se istražiti i implementirati druge formulacije semidefinitnih relaksacija bojanja grafova koje bi se tada usporedile s trenutnim implementacijama. U danjem radu, trebalo bi implementirati izračun Lovászove theta funkcije kao ocjenu veličine broja boja i maksimalne klike.

Također, implementiran je pronalazak velike klike grafa pomoću semidefinitne relaksacije problema i pomoću pohlepnog algoritma traženja lokalno maksimalne klike. Algoritam semidefinitne relaksacija pronalazi veće klike.

Ove rezultate trebalo bi usporediti s drugim nekomercijalnim bibliotekama.

Literatura

[1] R. Fritsch i G. Fritsch, *The Four-Color Theorem*. New York: Springer-Verlag, 1998.

[2] P. Briggs, K. Cooper, K. Kennedy, i L. Torczon, "Coloring heuristics for register allocation", sv. 39, 07 1989., str. 275–274. https://doi.org/10.1145/74818.74843

[3] D. C. Wood, "A technique for colouring a graph applicable to large scale timetabling problems", *The Computer Journal*, sv. 12, br. 4, str. 317–319, 01 1969. https://doi.org/10.1093/comjnl/12.4.317

[4] N. R. Council, *Mathematical Challenges from Theoretical/Computational Chemistry*. Washington, DC: The National Academies Press, 1995. https://doi.org/10.17226/4886

[5] D. Orkia i L. Ahmed, "Solving the max-cut problem using semidefinite optimization", u *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, 2016., str. 768–772. https://doi.org/10.1109/CIST.2016.7804990

[6] M. F. Anjos, "Semidefinite optimization approaches for satisfiability and maximum-satisfiability problems", *Journal on Satisfiability, Boolean Modelling and Computation*, sv. 1, br. 1, str. 1–47, 2006. https://doi.org/10.3233/SAT190001

[7] M. Hintermüller i K. Papafitsoros, "Chapter 11 - generating structured nonsmooth priors and associated primal-dual methods", u *Processing, Analyzing and Learning of Images, Shapes, and Forms: Part 2*, ser. Handbook of Numerical Analysis, R. Kimmel i X.-C. Tai, Ur. Elsevier, 2019., sv. 20, str. 437–502. https://doi.org/https://doi.org/10.1016/bs.hna.2019.08.001

[8] H. Bauschke, M. Dao, D. Noll, i H. Phan, “On slater’s condition and finite convergence of the douglas-rachford algorithm for solving convex feasibility problems in euclidean spaces”, *Journal of Global Optimization*, sv. 65, str. 329–349, 06 2016. https://doi.org/10.1007/s10898-015-0373-5

[9] L. Lovasz, “On the shannon capacity of a graph”, *IEEE Transactions on Information Theory*, sv. 25, br. 1, str. 1–7, 1979. https://doi.org/10.1109/TIT.1979.1055985

[10] D. Karger, R. Motwani, i M. Sudan, “Approximate graph coloring by semidefinite programming”, 1998. [Mrežno]. Adresa: https://arxiv.org/abs/cs/9812008

[11] A. Wigderson, “Improving the performance guarantee for approximate graph coloring”, *Journal of the ACM*, sv. 30, br. 4, str. 729–735, jul 2002. https://doi.org/10.1145/2157.2158

[12] S. Benson i Y. Ye, “Approximating maximum stable set and minimum graph coloring problems with the positive semidefinite relaxation”, *Applications and Algorithms of Complementarity*, 07 2000. https://doi.org/10.1007/978-1-4757-3279-5_1

[13] S. J. Benson i Y. Ye, “DSDP5 user guide — software for semidefinite programming”, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, teh. izv. ANL/MCS-TM-277, September 2005., http://www.mcs.anl.gov/~benson/dsdp.

[14] Q. Wang, X. Zhang, Y. Zhang, i Q. Yi, “Augem: Automatically generate high performance dense linear algebra kernels on x86 cpus”, u *SC ’13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2013., str. 1–12. https://doi.org/10.1145/2503210.2503219

[15] L. Šprajc, 2025. [Mrežno]. Adresa: https://github.com/lspra/graphColoring

Sažetak

Relaksacije semidefinitnog programiranja za probleme bojanja grafova i maksimalne klike

Lana Šprajc

U sklopu rada, dan je opis problema bojanja grafova i pronalaska maksimalne klike grafa. Opisani su jednostavni načini rješavanja ovih problema. Opisano je formulacija semidefinitnih problema kao i njihova primjena kod bojanja grafova i traženja maksimalne klike. Objašnjena je važnost Lovászove theta funkcije i dana formulacija semidefinitnog problema za pronalazak njezine vrijednosti.

Implementirano je bojanje grafova pomoću SDP relaksacije uz algoritam zaokruživanja vektorskim projekcijama i pomoću $\delta + 1$ bojanja i uspoređeni su rezultati. Također, implementiran je pronalazak velike klike grafa pomoću semidefinitne relaksacije problema i pomoću pohlepnog algoritma traženja lokalno maksimalne klike.

Implementiran je i program za vizualizaciju grafa koji prikazuje obojani graf ili njegovu maksimalnu kliku.

Ključne riječi: bojanje grafa; semidefinitno programiranje; maksimalna klika; Lovászova fukcija

Abstract

Semidefinite programming relaxation for graph coloring and maximum clique number

Lana Šprajc

In this paper, graph coloring and maximum clique number are described. Simple procedures for solving these problems are explained. The formulation of semidefinite problems is given and so is their application in graph coloring and finding max clique. The importance of Lovász theta function is explained and semidefinite program for finding its value is given.

Graph coloring is implemented using SDP relaxing and rounding via vector projections. Graph coloring is also solved using $\delta + 1$ coloring and the results are compared. Finding a large clique is also implemented both via semidefinite relaxation and using a greedy algorithm of finding locally maximal clique.

Graph visualisation program which shows either a colored graph or its maximum clique is implemented.

Keywords: graph coloring; SDP; maximal clique number; Lovász function