Univerza na Primorskem

Fakulteta za matematiko, naravoslovje in informacijske tehnologije

Matematične znanosti - 2. stopnja

Peter Muršič

Particija grafa, odkrivanje skupnosti in maksimalen prerez

Magistrsko delo

Koper, 2012

Mentor: prof. dr. Dragan Stevanović
Somentor: doc. dr. Martin Milanič

Ključna dokumentacijska informacija

Ime in PRIIMEK: Peter MURŠIČ

Naslov magistrskega dela: Particija grafa, odkrivanje skupnosti in maksimalen prerez

Kraj: Koper

Leto: 2012

Število listov: 56

Število slik: 13

Število referenc: 21

Mentor: prof. dr. Dragan Stevanović

Somentor: doc. dr. Martin Milanič

UDK: 519.17(043.2)

Ključne besede: particija grafa, odkrivanje skupnosti, algoritem Kernighan-Lin, maksimalen prerez, spektralna particija, modularnost, enostavna maksimizacija modularnosti, spektralna maksimizacija modularnosti, bisekcija grafa, hierarhično grozdenje, hevristike, aproksimacijski algoritmi, dendrogram

Math. Subj. Class. (2010): 05C50, 05C85, 90C35, 90C59, 65F30, 68W25

Izvleček:

Problemi *particije grafa*, *odkrivanja skupnosti* in *maksimalnega prereza* so problemi delitve točk grafa na več skupin, pri čemer optimiziramo vrednost neke kriterijske funkcije. Pri problemu particije grafa množico točk grafa delimo na l skupin vnaprej določenih velikosti in minimiziramo število povezav med skupinami. Pri problemu odkrivanja skupnosti delimo graf glede na njegovo "naravno" delitev (strukturo) in zato je število skupin in njihova velikost odkrita tekom reševanja problema in ne na začetku. Ponavadi uporabimo mero za kakovost delitve in jo optimiziramo. Problem maksimalnega prereza sprašuje po delitvi točk grafa na dve skupini tako, da bo število povezav med skupinama maksimalno. V magistrskem delu bom podrobneje opisal probleme ter njihove podprobleme, pogledal kako jih rešiti, ter (zaradi NP-težkosti problemov) opisal hevristike in aproksimacijske algoritme zanje.

Key words documentation

Name and SURNAME: Peter MURŠIČ

Title of Master's degree: Graph partitioning, community detection and max cut

Place: Koper

Year: 2012

Number of pages: 56

Number of figures: 13

Number of references: 21

Supervisor: Prof. Dragan Stevanović

Co-supervisor: Assist. Prof. Martin Milanič

UDC: 519.17(043.2)

Key words: graph partitioning, community detection, Kernighan-Lin algorithm, max cut, spectral partitioning, modularity, simple modularity maximization, spectral modularity maximization, graph bisection, hierarchical clustering, heuristics, approximation algorithm, dendrogram

Math. Subj. Class. (2010): 05C50, 05C85, 90C35, 90C59, 65F30, 68W25

Abstract:

Graph partitioning, community detection and max cut problems are problems of dividing the vertices of a graph into more groups in such a way that the value of a certain objective function is optimized. The graph partitioning problem is asking for a division of the vertices of graph into l groups of given sizes, such that the cut size (the number of edges between groups) is minimized. The community detection problem asks for a "natural" division of a graph, and thus the number of groups and their sizes is given by the network structure. Usually we use a measure for a network division and we optimize it. The max cut problem asks for a division of the vertices of a graph into two groups, such that the number of edges between groups is maximized. We will illustrate the problems and their subproblems, show how they can be solved and (because they are NP-hard) describe heuristics and approximation algorithms for them.

Kazalo

Kazalo slik

ZAHVALA

Iskreno se zahvaljujem mentorju Draganu Stevanoviću, dekanji Klavdiji Kutnar, Bojanu Kuzmi in vsem zaposlenim na UP FAMNIT, ki so kakorkoli pripomogli k nastanku magistrskega dela.

Zahvaljujem se somentorju Martinu Milaniču za odlične popravke in napotke pri izdelavi magistrskega dela. Zahvaljujem se predvsem za celoletno zavzemanje in pomoč pri pripravah na študij v ZDA.

Tudi malenkosti kot so strpnost, podpora in pozitivna pričakovanja so bistveno pripomogle k učinkovitejšemu spoprijemanju z učenjem in delom. Zahvaljujem se družini za večletno spodbudo in pomoč pri izobraževanju.

Zahvaljujem se Alenki za vselejšnjo podporo pri izbiri študija v tujini, za požrtvovalnost in ljubezen.

Poglavje 1

UVOD

V magistrskem delu se bom dotaknil teme o *particiji grafa*. Particija grafa je klasični problem v računalništvu. Problem particije grafa sprašuje po particiji točk grafa, pri čemer imamo število skupin (podgrafov) ter njihovo velikost v naprej podano in kjer je število povezav med skupinami minimalno. Včasih imamo namesto fiksnega števila in velikosti skupin podane meje, med katerimi naj se ta števila gibljejo. Najbolj osnoven primer takega problema je problem bisekcije grafa, tj., problem optimalne razdelitve grafa na dve skupini.

Žal za rešitev problema ni znan noben algoritem polinomske časovne zahtevnosti, obstajajo pa t.i. hevristike (algoritmi, ki izračunajo približek pravega rezultata). Poglobil se bom v dve hevristiki za problem bisekcije grafa:

- *Algoritem Kernighan-Lin* deluje tako, da iz neke začetne razdelitve grafa permutira po eno točko iz vsake skupine med seboj tako, da med možnimi permutacijami poskuša dobiti takšno, ki bo trenutno število povezav med skupinama najbolj zmanjšala.

- *Spektralna particija* je algoritem, ki na osnovi lastnih vrednosti Laplaceove matrike grafa dobi približek za problem bisekcije grafa.

Problem, ki je podoben particiji grafa, se imenuje *odkrivanje skupnosti*. Skupno particiji grafa je pri problemu odkrivanja skupnosti to, da tudi ta problem sprašuje po particiji točk grafa, vendar tokrat ni določeno, na koliko skupin delimo ter njihova velikost. Tudi težnja po minimalnosti povezav med skupinami je nekoliko prirejena, saj bi sicer dobili trivialno rešitev, pri kateri so v prvi skupini vse točke grafa in nobene ni v drugi skupini.

Za odkrivanje skupnosti bomo spoznali več hevristik, ena od teh je prirejeni zgornji algoritem Kernighan-Lin. Med njimi je tudi algoritem, prirejen spektralni particiji. Slednja algoritma uporabljamo za delitev grafa v dve skupini; spoznali bomo še algoritme, ki razdelijo graf na več skupin. Za delitev grafa na več kot dve skupini se lahko uporabi večkratno bisekcijo, obstajajo pa tudi drugi načini.

Problem *maksimalnega prereza* je podoben problemu bisekcije grafa. Sprašuje po delitvi točk grafa na dve skupini poljubne velikosti, pri čemer je število povezav med skupinama največje možno. Spoznali bomo dva zelo preprosta aproksimacijska algoritma za problem maksimalnega prereza, ter enega bolj kompleksnega.

Magistrsko delo temelji predvsem na Newmanovi knjigi [16] in delno na članku [6] Goemansa in Williamsona. Osnove so povzete po knjigi [10] ter zapiskih predavanj [13] in [14], ki temeljijo na [2], [3], [7], [8], [11], [12], [19] in [21]. Slike so bile izdelane s programoma GeoGebra [4] in GNU Octave [5].

Poglavje 2

PONOVITEV OSNOV

V tem poglavju bomo ponovili nekatere osnovne matematične pojme, ki jih bomo uporabili v magistrskem delu. Pogledali si bomo grafe, nekaj linearne algebre in verjetnosti ter osnove algoritmov na grafih.

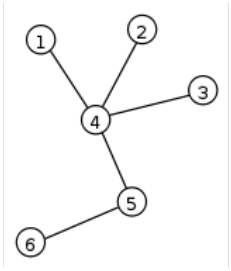
2.1 Graf

Definicija 2.1. *Graf* Γ *je urejen par* (V, E)*, kjer je* V *neprazna končna množica (*$V = \{v_1, v_2, \ldots, v_n\}$*) in* $E \subseteq \{\{x, y\} | x \in V \wedge y \in V \wedge x \neq y\}$*. Poleg tega je množica* E *simetrična (tj. če je* $\{x, y\} \in E$*, potem je tudi* $\{y, x\} \in E$*). Uporabljali bomo zapis* (x, y) *za povezave* $\{x, y\} \in E$*.*

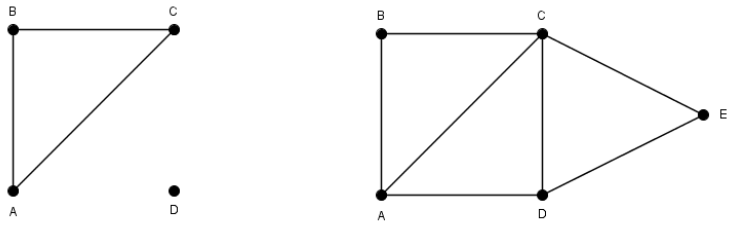
Množici V bomo rekli tudi množica vozlišč (točk) grafa Γ, njenim elementom pa vozlišča (točke). Množico E bomo poimenovali množica povezav grafa Γ, njene elemente pa povezave. V kolikor ne bo izrecno izpostavljeno, iz katerih množic je graf $\Gamma = (V, E)$ sestavljen, potem bomo z $V(\Gamma)$ imeli v mislih množico V, z $E(\Gamma)$ bomo pa imeli v mislih množico E. Z n običajno označimo število točk v grafu $n = |V|$, ter z m označujemo število povezav v grafu $m = |E|$. Število n bomo imenovali red grafa Γ. Če sta točki v_i in v_j povezani, bomo to označili z $v_i \sim v_j$ in ju poimenovali krajišči povezave (v_i, v_j). Graf si ponavadi predstavljamo tako, da za vsako vozlišče narišemo točko v ravnini ter jo, v kolikor je potrebno, označimo. Za vsaki dve vozlišči preverimo, ali obstaja povezava med njima v množici povezav, in ju, v kolikor povezava obstaja, povežemo. Kako točke in povezave narišemo, je nepomembno. Pomembno je le, da lahko iz slike vidimo, kateri pari vozlišč tvorijo povezavo in kateri ne. Včasih graf tudi utežimo, tj. vsaki povezavi (v_i, v_j) pripišemo realno število $w_{i,j}$, ki ji pravimo utež.

Definicija 2.2. *Naj bosta* $\Gamma = (V, E)$ *in* $\Gamma' = (V', E')$ *grafa. Če velja, da je* $V \subseteq V'$ *in* $E \subseteq E'$*, potem pravimo, da je graf* Γ *podgraf grafa* Γ'*. Pišemo tudi* $\Gamma \subseteq \Gamma'$*.*

Z drugimi besedami, če graf Γ dobimo tako, da grafu Γ' odstranimo nekaj vozlišč in povezav, potem je graf Γ podgraf grafa Γ'.



Slika 2.1: Primer grafa.

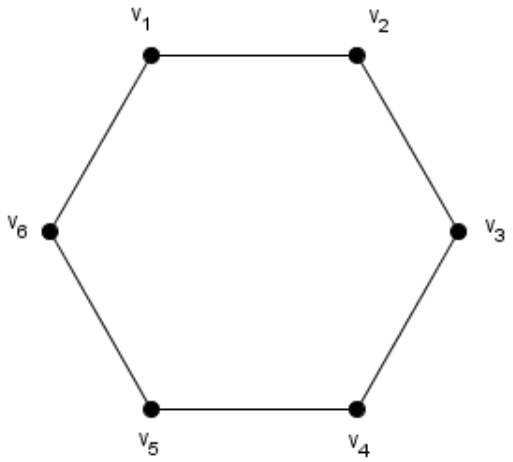


Slika 2.2: Graf na levi je podgraf grafa na desni.

Definicija 2.3. *Naj bo $\Gamma = (V, E)$ graf in $i \in V$. Stopnja k_i točke i je število sosedov točke i oz. število povezav v E, ki vsebujejo točko i (pri tem povezavi (j, i) in (i, j) štejemo samo enkrat).*

Primer 2.4. *Na sliki (2.1) je $k_1 = 1, k_2 = 1, k_3 = 1, k_4 = 4, k_5 = 2, k_6 = 1$.*

Definicija 2.5. *Naj bo $n \geq 3$ naravno število.* ***Cikel*** *C_n je graf $\Gamma = (V, E)$ na n vozliščih, kjer je $V = \{v_1, v_2, \ldots, v_n\}$ in $E = \{(v_i, v_{i+1}) | i \in \{1, 2, \ldots, n-1\}\} \cup \{(v_n, v_1)\}$.*



Slika 2.3: Primer cikla C_6.

Definicija 2.6. ***Pot*** P_n *je graf* $\Gamma = (V, E)$ *na* n *vozliščih, kjer je* $V = \{v_1, v_2, \ldots, v_n\}$ *in* $E = \{(v_i, v_{i+1}) | i \in \{1, 2, \ldots, n-1\}\}$. *Vozliščema* v_1 *in* v_n *pravimo krajišči poti, številu* $n-1$ *pa rečemo dolžina poti.*

Slika 2.4: Primer poti P_6.

Ko govorimo o ciklih oz. poteh v grafu Γ, imamo v mislih cikle oz. poti kot podgrafe grafa Γ. Za n-cikel gledamo podgraf C_n in za poti dolžine $n-1$ gledamo podgraf P_n v grafu Γ.

Definicija 2.7. *Komplement grafa* Γ *je graf* Γ^c *z lastnostma:*

- $V(\Gamma) = V(\Gamma^c)$ *in*
- *za vse* $u, v \in V(\Gamma)$ *z* $u \neq v$ *velja:* $(u, v) \in E(\Gamma) \Leftrightarrow (u, v) \notin E(\Gamma^c)$.

Definicija 2.8. *Graf* $\Gamma = (V, E)$ *je* ***povezan****, če za vsaki dve točki* $i, j \in V$ *obstaja pot v grafu* Γ*, ki ima točki* i *in* j *za krajišči. Maksimalnim povezanim podgrafom grafa* Γ *pravimo komponente grafa* Γ.

Definicija 2.9. *Graf* Γ *je* ***drevo****, če je povezan in ne vsebuje ciklov kot pografe.*

Definicija 2.10. *Graf* Γ *je dvodelen, če obstaja taka razdelitev množice točk* $V(\Gamma) = X \cup Y$*, da ima vsaka povezava eno krajišče v množici* X *in drugo v množici* Y.

2.2 Nekaj linearne algebre

Definicija 2.11. *Matrika je pravokotna shema števil (lahko tudi simbolov ali izrazov), ki jim pravimo elementi matrike in so razporjeni v vrstice in stolpce. Števila so elementi polja* $\mathbb{F}$*, katero bo v magistrskem delu vselej množica realnih števil* $\mathbb{F} = \mathbb{R}$*. Matrika* B *je velikosti* $n \times m$*, če ima matrika* B n *vrstic in* m *stolpcev.*

Množico vseh matrik velikosti $n \times m$, ki imajo koeficiente v množici realnih števil, bomo označili z $\mathbb{R}^{n \times m}$. Naj bo $B \in \mathbb{R}^{n \times m}$ matrika ter $i \in \{1, 2, 3, ..., n\}, j \in \{1, 2, 3, ..., m\}$. Element na (i, j)-ti koordinati matrike B bomo označili z $B_{i,j}$. Z $B_{(i)}$ bomo označili i-to vrstico matrike B ter z $B^{(j)}$ j-ti stolpec matrike B. Z $d_0(B)$ bomo označili glavno diagonalo matrike B, z $d_1(B)$ diagonalo nad glavno diagonalo, z $d_{-1}(B)$ diagonalo pod glavno diagonalo, itd. Naj bosta C, D matriki. Potem operacijo $+ : \mathbb{R}^{n \times m} \times \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m}$, ki slika s predpisom $(C + D)_{i,j} = C_{i,j} + D_{i,j}$, imenujemo *običajno seštevanje matrik*. Operacijo $\cdot : \mathbb{R}^{n \times m} \times \mathbb{R}^{m \times p} \to \mathbb{R}^{n \times p}$, s predpisom $(C \cdot D)_{i,j} = \sum_{k=1}^{n} C_{i,k} \cdot D_{k,j}$, pa imenujemo *običajno množenje matrik*. Naj bo $\lambda \in \mathbb{R}$. Množenje matrike C s skalarjem λ je definirano kot $(\lambda \cdot C)_{i,j} = \lambda \cdot C_{i,j}$. Odštevanje matrik $C - D$ $(C, D \in \mathbb{R}^{n \times m})$ je definirano kot $C - D = C + (-1) \cdot D$.

Z 0_n bomo označili matriko velikosti $n \times n$ s samimi ničelnimi koeficienti, z I_n identično matriko velikosti $n \times n$ in z J_n matriko s samimi enicami velikosti $n \times n$. Transponirana matrika $A^\top$ matrike A je definirana kot $(A^\top)_{i,j} = A_{j,i}$. Inverz matrike $A \in \mathbb{R}^{n\times n}$ je taka matrika A^{-1}, za katero velja $AA^{-1} = A^{-1}A = I_n$ (če matrika A^{-1} obstaja, potem je enolično določena). Potenciranje matrik lahko definiramo rekurzivno. Naj bo $k \in \mathbb{N}$ ter $A \in \mathbb{R}^{n\times n}$, potem je $A^0 = I_n$, $A^n = A \cdot A^{n-1}$ in $A^{-n} = (A^{-1})^n$ (če A^{-1} obstaja).

Vektor a dolžine n si lahko predstavljamo kot $n \times 1$ matriko (tudi $1 \times n$, pišemo tudi $a \in \mathbb{R}^n$). Operacije z vektorji so definirane kot operacije z matrikami. Skalarni produkt vektorjev a in b dolžine n je definiran kot $\langle a, b\rangle = \sum_{i=1}^{n} a_i b_i$. Vektorja a in b sta pravokotna (ortogonalna), če je njun skalarni produkt enak 0. Vektor a je enotski (unitaren, normiran), če je $\langle a, a\rangle = 1$. Vektor a lahko naredimo enotski (normalizacija vektorja), tako da delimo vektor a z njegovo dolžino, ki je enaka $\sqrt{\langle a, a\rangle}$. Matriko lahko tudi normaliziramo, in sicer tako, da jo delimo z njeno determinanto (v kolikor ni enaka 0). Matrikam, ki imajo enako število vrstic in stolpcev, pravimo kvadrane matrike. Matriki A z lastnostjo $A^\top = A$ pravimo simetrična matrika. Velikokrat produkt matrik, ki vrne matriko velikosti 1×1 enačimo z realnim številom x. V tem primeru imamo v mislih, da je edini element matrike enak številu x.

Primer 2.12. *Spodaj je primer matrike $B \in \mathbb{R}^{3\times 3}$:*

$$B = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 1 \\ \pi & 0 & -1 \end{bmatrix}.$$

Definicija 2.13. *Determinanta* $\det(B)$ *matrike $B \in \mathbb{R}^{n\times n}$ je enaka:*

$$\det(B) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} B_{i,\sigma_i}.$$

S_n je simetrična grupa na n točkah, $sgn(\sigma) = 1$, če je σ soda permutacija in -1, če je liha. Spremenljivka σ_i predstavlja element v katerega se i preslika s permutacijo σ.

Primer 2.14. *Determinanta matrike B iz primera (2.12) je* $\det(B) = -\pi$.

Formula navedena v definiciji (2.13) je precej zamudna in zato običajno uporabimo drugačen pristop. Determinanta 2×2 matrike se poračuna enostavno po formuli

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

Za $n \times n$ matrike ($n \geq 3$) se uporabi postopek, ki determinanto $n \times n$ matrike spremeni v vsoto determinant $(n-1) \times (n-1)$ matrik. Ta postopek se ponavlja, dokler nimamo le še 2×2 matrike, kar že znamo enostavno izračunati. Naj bo A matrika velikosti $n \times n$ in $A^{i,j}$ matrika velikosti $(n-1) \times (n-1)$, ki jo dobimo tako, da matriki A odstranimo i-to vrstico in j-ti stolpec. Eden od postopkov je t.i. razvoj po j-tem stolpcu, ki se izračuna po formuli:

$$\det(A) = \sum_{i=1}^{n} A_{i,j} \cdot (-1)^{i+j} \cdot \det(A^{i,j}).$$

Razvoj po i-ti vrstici je drugi postopek, podoben prejšnjemu:

$$\det(A) = \sum_{j=1}^{n} A_{i,j} \cdot (-1)^{i+j} \cdot \det(A^{i,j}) .$$

Ni se treba strogo držati enega ali drugega načina, lahko ju uporabljamo mešano. Najlažje je, če se odločimo za postopek glede na to, ali imamo kako vrstico ali stolpec s precej ničelnimi elementi in razvijemo po njej ali njem.

Definicija 2.15. *Realno število α imenujemo **lastna vrednost** za matriko $A \in \mathbb{R}^{n\times n}$, če obstaja tak neničelen vektor $v \in \mathbb{R}^n$, da je $Av = \alpha v$. Vektor v imenujemo **lastni vektor** za A pri lastni vrednosti α.*

Primer 2.16. *Recimo da imamo matriko*

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

Vektor

$$x = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$$

je lastni vektor matrike A z lastno vrednostjo 1, saj je

$$A \cdot x = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \end{bmatrix} = \begin{bmatrix} 2 \cdot 3 + 1 \cdot (-3) \\ 1 \cdot 3 + 2 \cdot (-3) \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \end{bmatrix} = 1 \cdot \begin{bmatrix} 3 \\ -3 \end{bmatrix} = 1 \cdot x.$$

Definicija 2.17. *Naj bo $A \in \mathbb{R}^{n\times n}$ matrika. Karakteristični polinom matrike A je definiran kot:*

$$p_A(\lambda) = \det(A - \lambda I_n).$$

Trditev 2.18. *Ničle karakterističnega polinoma $p_A(\lambda)$ so ravno lastne vrednosti matrike A.*

Trditev 2.19. *Produkt lastnih vrednosti matrike A je enak determinanti matrike A.*

Definicija 2.20. *Spekter grafa Γ je nabor lastnih vrednosti njegove matrike sosednosti $A(\Gamma)$ (ki bo definirana v naslednjem podpoglavju). Od tod izvira ime za nekatere algoritme, ki jih bom kasneje obravnaval.*

Definicija 2.21. *Sled $\mathrm{tr}(A)$ matrike $A \in \mathbb{R}^{n\times n}$ je definirana kot*

$$\mathrm{tr}(A) = \sum_{i=1}^{n} A_{i,i}.$$

Trditev 2.22. *Naj bosta matriki $A, B \in \mathbb{R}^{n\times n}$ in $\lambda \in \mathbb{R}$. Sled matrike ima naslednje lastnosti:*

- $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$,
- $\mathrm{tr}(\lambda A) = \lambda \mathrm{tr}(A)$,
- $\mathrm{tr}(A^\top) = \mathrm{tr}(A)$.

2.3 Matrika sosednosti

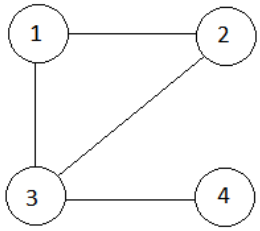
Definicija 2.23. *Naj bo n red grafa Γ ter $V(\Gamma) = \{v_1, v_2, \ldots, v_n\}$. Matrika sosednosti $A(\Gamma)$ grafa Γ je matrika velikosti $n \times n$ definirana z:*

$$A(\Gamma)_{ij} = \begin{cases} 1, & \text{če } (v_i, v_j) \in E(\Gamma), \text{ in} \\ 0, & \text{sicer.} \end{cases}$$

V primeru, ko imamo utežen graf, je matrika sosednosti A enaka $A_{i,j} = w_{i,j}$, če je $v_i \sim v_j$ in 0 sicer.

Primer 2.24. *Glede na to v kakšnem vrstnem redu izberemo vozlišča, lahko dobimo različne matrike sosednosti za isti graf. Če sta matriki A_1 in A_2 matriki sosednosti za isti graf, potem obstaja taka permutacijska matrika P, da je $A_2 = PA_1P^{-1}$. V tem primeru pravimo, da sta si matriki A_1 in A_2 podobni.*

Oglejmo si to na sledečem primeru:



Slika 2.5: Primer označenega grafa.

Če izberemo vozlišča za zgornji graf v vrstnem redu $1, 2, 3, 4$, dobimo sledečo matriko sosednosti:

$$A_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Če izberemo vozlišča v vrstnem redu $1, 4, 2, 3$ dobimo naslednjo matriko matriko sosednosti:

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}.$$

V tem primeru je permutacijska matrika P, ki ima lastnost $A_2 = PA_1P^{-1}$ enaka:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Ena od lastnosti determinante je, da imata podobni matriki $A_1, A_2 \in \mathbb{R}^{n\times n}$ enako determinanto:

$$\det(A_1) = \det(PA_2P^{-1}) = \det(P)\cdot\det(A_2)\cdot\det(P^{-1}) = \det(P)\cdot\det(P^{-1})\cdot\det(A_2) = \det(A_2).$$

Prav tako imata podobni matriki enak karakteristični polinom in s tem enake lastne vrednosti:

$$\begin{aligned} p_{A_1}(\lambda) &= \det(A_1 - \lambda I_n) = \det(PA_2P^{-1} - \lambda I_n) = \\ &= \det(P(A_2 - \lambda I_n)P^{-1}) = \det(P)\cdot\det(A_2 - \lambda I_n)\cdot\det(P^{-1}) = \\ &= \det(P)\cdot\det(P^{-1})\cdot\det(A_2 - \lambda I_n) = \det(A_2 - \lambda I_n) = p_{A_2}(\lambda). \end{aligned}$$

Trditev 2.25. *Naj bo A matrika sosednosti grafa Γ. Za matriko A veljajo naslednje lastnosti:*

- $A^\top = A$ *oz.* $A_{i,j} = A_{j,i}$,
- $\sum_{i=1}^{n} A_{i,j} = \sum_{i=1}^{n} A_{j,i} = k_j$,
- $\sum_{j=1}^{n} \sum_{i=1}^{n} A_{i,j} = \sum_{j=1}^{n} k_j = 2m$.

Trditev 2.26. *Naj bo A matrika sosednosti grafa Γ reda n in B matrika sosednosti komplementa Γ^c grafa Γ. Potem lahko matriko B zapišemo s pomočjo matrike A na sledeč način:*

$$B = J_n - I_n - A.$$

2.4 Verjetnost

Ponovili bomo osnove verjetnosti. Verjetnost sloni na teoriji mere, v katero se ne bomo poglobili.

Definicija 2.27. *Naj bo Ω množica (izidov). Množici $\mathcal{F}$, ki je podmnožica potenčne množice $\mathcal{P}(\Omega)$ množice Ω, z lastnostmi:*

- *$\mathcal{F}$ je neprazna,*
- *$\mathcal{F}$ je zaprta za komplemente:*

$$A \in \mathcal{F} \Rightarrow \Omega \setminus A \in \mathcal{F},$$

- *$\mathcal{F}$ je zaprta za števne unije:*

$$A_1, A_2, ... \in \mathcal{F} \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{F},$$

pravimo prostor dogodkov.

Definicija 2.28. *Dana je množica izidov Ω in prostor dogodkov $\mathcal{F}$. Verjetnost (verjetnostna mera) na $(\Omega, \mathcal{F})$ je preslikava $P : \mathcal{F} \to [0, 1]$, za katero velja:*

- *$P(\Omega) = 1$ in $P(\emptyset) = 0$,*
- *za paroma disjunktne dogodke $A_1, A_2, ... \in \mathcal{F}$ velja:*

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i).$$

Definicija 2.29. *Verjetnostni prostor je trojica $(\Omega, \mathcal{F}, P)$, kjer*

- *je Ω množica (izidov),*
- *je $\mathcal{F}$ prostor dogodkov,*
- *je P verjetnost na $(\Omega, \mathcal{F})$.*

Definicija 2.30. *Naj bo $(\Omega, \mathcal{F}, P)$ verjetnostni prostor.* ***Slučajna spremenljivka*** *na $(\Omega, \mathcal{F}, P)$ je taka funkcija $X : \Omega \to \mathbb{R}$, za katero velja:*

$$\{w \in \Omega : X(w) \leq x\} \in \mathcal{F} \text{ za vse } x \in \mathbb{R}.$$

Poznamo diskretne in zvezne slučajne spremenljvike.

Definicija 2.31. *Vsako slučajno spremenljivko X lahko podamo z njeno* ***porazdelitveno funkcijo*** *F_X, definirano kot:*

$$F_X : \mathbb{R} \to [0, 1],$$
$$F_X(x) := P(X \leq x) := P(\{w \in \Omega : X(w) \leq x\}).$$

Definicija 2.32. *Naj bo $(\Omega, \mathcal{F}, P)$ verjetnostni prostor.* ***Diskretna slučajna spremenljivka*** *$X : \Omega \to \mathbb{R}$, za katero velja:*

- *slika X, tj. $\{X(w) : w \in \Omega\}$ je števna množica,*
- *za vsak $x \in \mathbb{R}$ velja:*

$$X^{-1}(x) := \{w \in \Omega : X(w) = x\} \in \mathcal{F}.$$

Definicija 2.33. *Slučajna spremenljivka X je* ***zvezna****, če njeno porazdelitveno funkcijo F_X lahko zapišemo v obliki:*

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(u)du,$$

kjer je $f_X \geq 0$ nenegativna funkcija. Taki funkciji f_X pravimo ***gostota*** *slučajne spremenljivke X.*

Porazdelitvenih funkcij je veliko tako za diskretne slučajne spremenljivke, kot za zvezne slučanje spremenljivke. Med znanimi diskretnimi porazdelitvami so:

- Bernoullijeva porazelitev,
- binomska porazdelitev in
- Poissonova porazdelitev.

Med znanimi zveznimi porazdelitvami so:

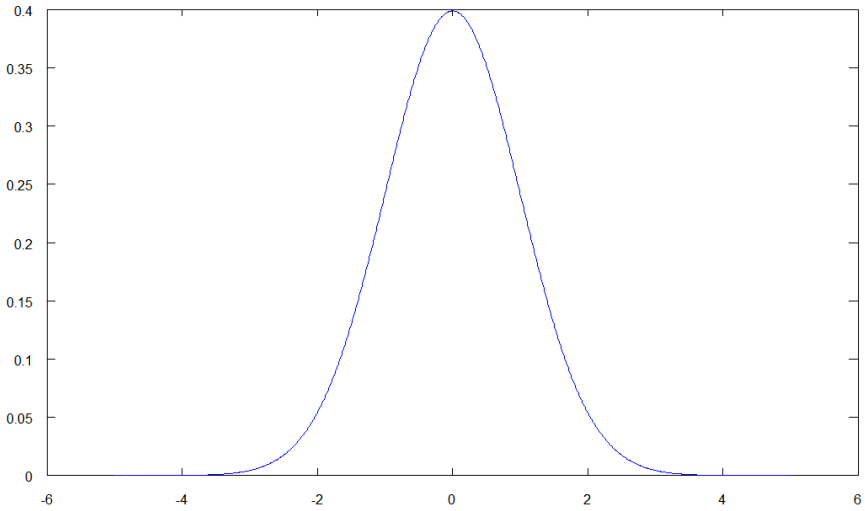
- enakomerna porazdelitev,
- normalna porazdelitev,
- porazdelitev hi-kvadrat,
- eksponentna porazdelitev in
- gama porazdelitev.

Opisal bom le normalno porazdelitev, saj je edina, ki pride v poštev kasneje.

Definicija 2.34. ***Normalna porazdelitev*** *s parametroma* μ *in* $\sigma^2 > 0$ *je definirana z gostoto:*

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \text{ za } x \in \mathbb{R}.$$

Normalno porazdelitev označimo z $N(\mu, \sigma^2)$*. V primeru ko je* $\mu = 0, \sigma = 1$*, jo imenujemo* ***standardna porazdelitev****.*



Slika 2.6: Graf gostote standardne normalne porazdelitve.

Definicija 2.35. *Matematično upanje diskretne slučajne spremenljivke* X *označimo z* $\mathbb{E}(X)$ *in ga definiramo kot:*

$$\mathbb{E}(X) = \sum_{x \in Slika(X)} xP(X = x),$$

če vrsta konvergira absolutno (tj. če velja $\sum_{x \in Slika(X)} |xP(X = x)| < \infty$*),* $Slika(X)$ *predstavlja množico* $\{X(w) : w \in \Omega\}$ *in* $P(X = x) := P(\{w \in \Omega : X(w) = x\})$*.*

Definicija 2.36. *Matematično upanje zvezne slučajne spremenljivke* X *z gostoto* f_X *označimo z* $\mathbb{E}(X)$ *in ga definiramo kot:*

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_X(x) dx,$$

če integral konvergira absolutno (tj. če velja $\int_{-\infty}^{\infty} |x f_X(x)| dx < \infty$*).*

2.5 Algoritmi in analiza algoritmov

Algoritem je vsako dobro definirano zaporedje ukazov, s katerim kaj izračunamo ali rešimo kak problem [2]. Algoritem vhodne podatke spremeni v izhodne podatke. Od algoritma zahtevamo, da se konča v končnem številu korakov (računskih operacij). Ko razvijemo algoritem za dani problem, je potrebno zagotoviti še:

- analizo časovne zahtevnosti algoritma,
- dokaz pravilnosti algoritma.

Algoritme lahko zapisujemo v naravnem jeziku ali v psevdokodi.

Dan je računski (optimizacijski, odločitveni, ...) problem P in konkretni vhodni podatki za problem P. Velikost naloge je število bitov, potrebnih za zapis naloge v računalniku. Pri tem moramo izbrati primeren način zapisa.

Naj bo A algoritem, ki reši problem P. Čas izvajanja algoritma A je število izvedenih osnovnih računskih operacij (seštevanje, odštevanje, množenje, primerjava dveh števil ...). Časovna zahtevnost algoritma je funkcija $T_A(n)$, ki meri čas izvajanja algoritma A v najslabšem primeru. Funkcija T_A preslika naravno število n v največji čas izvajanja, ki ga algoritem A porabi za rešitev naloge velikosti kvečjemu n.

Opomba:

- Poleg časovne včasih ocenjujemo tudi prostorsko zahtevnost (koliko pomnilnika potrebujemo za izvajanje).
- Če poznamo porazdelitev vhodnih podatkov, lahko ocenjujemo tudi pričakovano časovno zahtevnost.

Naj bosta dani funkciji $f, g : \mathbb{N} \to \mathbb{N}$. Funkcija f je (kvečjemu) reda $O(g)$, če obstaja taka konstanta $C > 0$, da je $f(n) \leq C \cdot g(n)$ za vse $n \in \mathbb{N}$ (več v [12]). Namesto pravilnega zapisa $f \in O(g)$, kjer $O(g)$ označuje množico funkcij f z zgornjo lastnostjo, je v ustaljeni rabi tudi zapis $f = O(g)$.

Če je $f = O(g)$, zapišemo tudi $g \in \Omega(f)$ in rečemo, da je g (vsaj) reda $\Omega(f)$. Funkciji f in g sta istega reda, če je $f = O(g)$ in $f = \Omega(g)$. Zapis: $f = \Theta(g)$.

Nekaj lastnosti O (brez dokaza):

1. Konstantni faktor lahko ignoriramo: za vse $k > 0$ je $k \cdot f$ reda $O(f)$.
2. Višje potence rastejo hitreje kot nižje: n^r je $O(n^s)$, če je $r \leq s$.
3. Hitrost rasti vsote je hitrost najhitreje rastočega sumanda: če je f reda $O(g)$, potem je $f + g$ reda $O(g)$. Primer: Izraz $6n^3 + 9n^2$ je $O(n^3)$.
4. Red polinoma je enak redu vodilnega člena: polinom stopnje d je $O(n^d)$.
5. Tranzitivnost: če je f reda $O(g)$ in g reda $O(h)$, potem je f reda $O(h)$.
6. Eksponentne funkcije rastejo hitreje kot potence: za vse $k \geq 0$, $b > 1$ je n^k reda $O(b^n)$. Primer: n^4 je $O(2^n)$, pa tudi $O(e^n)$ in $O(1.0001^n)$.

7. Logaritmi rastejo počasneje kot potence: za vse $k > 0$, $b > 1$ je $\log_b n$ reda $O(nk)$. Primer: $\log_2 n$ je reda $O(n^{1/2})$.

8. Logaritmi rastejo enako hitro: za vse $b, d > 1$ je $\log_b n$ reda $O(\log_d n)$.

9. Če je f reda $O(g)$ in h reda $O(r)$, potem je $f \cdot h$ reda $O(g \cdot r)$. Primer: če je f reda $O(n^2)$ in g reda $O(\log_2 n)$, potem je $f \cdot g$ reda $O(n^2 \log_2 n)$.

Algoritem A je polinomski, če je njegova časovna zahtevnost $T_A(n)$ kvečjemu reda $O(n^k)$ za neki $k \in \mathbb{N}$. Polinomskim algoritmom pravimo tudi učinkoviti algoritmi.

Če je $m \in O(n)$ pravimo, da je graf z n točkami in m povezavami redek, če pa je $m \in \Omega(n^2)$, pravimo, da je graf gost.

2.6 Predstavitve grafov

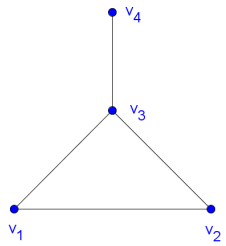
Kako predstaviti graf v pomnilniku računalnika? Poglejmo si dva glavna načina: matriko sosednosti in sezname sosedov [7]. Katero predstavitev izberemo, je zelo odvisno od tega, s kakšnimi grafi bomo delali in kakšne operacije hočemo izvajati na teh grafih.

- **Matrika sosednosti**

  Kot smo že povedali, je matrika sosednosti A grafa Γ enaka:

$$A_{i,j} = \begin{cases} 1, & \text{če } v_i \sim v_j, \text{ in} \\ 0, & \text{sicer.} \end{cases}$$

  Primer:



Slika 2.7: Primer grafa.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

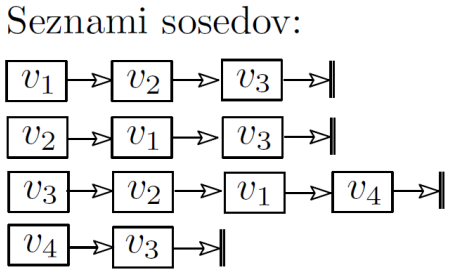
Slika 2.8: Matrika sosednosti za zgornji graf.

Velikostni red zapisa matrike A je $\Theta(n^2)$.

- **Seznami sosedov**

  Za vsako vozlišče grafa zapišemo seznam njegovih sosedov (v poljubnem vrstnem redu).

  Primer:



Slika 2.9: Seznami sosedov za zgornji graf.

  Velikostni red zapisa:

$$\sum_{i=1}^{n}[1+k_{v_i}] = n + \sum_{i=1}^{n} k_{v_i} = O(n+m).$$

  Ta predstavitev je še posebej uporabna za redke grafe (take z $O(n)$ povezavami). Mnogi zanimivi grafi so redki.

Primerjava predstavitev:

- Z matriko sosednosti lahko v konstantnem času $O(1)$:

  1. preverimo, ali je $v_i \sim v_j$,
  2. odstranimo povezavo,
  3. dodamo povezavo.

  Za 1. operacijo s seznami sosedov porabimo $O(k_{v_i})$ časa, saj moramo pregledati seznam vseh sosedov točke v_i (ali v_j). Za 2. operacijo s seznami sosedov porabimo $O(\max\{k_{v_i}, k_{v_j}\})$ časa, saj moramo pregledati oba seznama sosedov točk v_i in v_j.

- Prostorska zahtevnost matrike sosednosti je $\Theta(n^2)$, neodvisno od števila povezav. Prostorska zahtevnost seznamov sosedov pa je $O(n+m)$, kar je $O(n)$ za redke grafe.

Pravimo, da je grafovski problem rešljiv v linearnem času, če zanj obstaja algoritem časovne zahtevnosti $O(n+m)$. Pri tem predpostavimo predstavitev grafa s seznami sosedov.

2.7 NP-težki problemi, aproksimacijski algoritmi

Mnogi pomembni optimizacijski problemi so NP-težki [3] in zanje ne poznamo učinkovitih (polinomskih) algoritmov za natančno reševanje. V takih primerih se pogosto zadovoljimo s čim boljšo približno rešitvijo (aproksimacijo). O aproksimacijskih algoritmih običajno govorimo, ko imamo nekakšno "garancijo" za kvaliteto dobljene rešitve [19]. Če takšne "garancije" nimamo, govorimo o hevristikah. Hevristike so neformalne, intuitivne, spekulativne strategije, ki ne zagotovijo pravilnosti dobljene rešitve.

Definicija 2.37. *Optimizacijska naloga je trojka* (D, f, opt)*, kjer je:*

- D *množica dopustnih rešitev,*
- $f : D \to \mathbb{R}$ *kriterijska funkcija in*
- $opt \in \{\min, \max\}$ *vrsta naloge: minimizacija ali maksimizacija.*

Definicija 2.38. *Naj bo* (D, f, opt) *optimizacijska naloga, pri kateri ima kriterijska funkcija za vsako dopustno rešitev pozitivno vrednost. Algoritem za reševanje takega problema ima faktor aproksimacije* ρ *(je* ρ*-aproksimacijski), če je polinomske časovne zahtevnosti in velja: naj bo* I *poljubna naloga problema,* OPT *vrednost optimalne rešitve pri tej nalogi in* $f_A(I)$ *vrednost rešitve, ki jo vrne algoritem. Potem mora veljati:*

- *če je* $opt = \min$*, je* $\rho \geq 1$ *in velja* $\frac{f_A(I)}{OPT} \leq \rho$*,*
- *če je* $opt = \max$*, je* $\rho \leq 1$ *in velja* $\frac{f_A(I)}{OPT} \geq \rho$*.*

Ekvivalentno:

$$\rho \cdot OPT \geq f_A(I) \text{ za } opt = \min,$$
$$\rho \cdot OPT \leq f_A(I) \text{ za } opt = \max.$$

Obstajajo različne kategorije algoritmov, bolje rečeno strategij, ki jih uporabljamo za reševanje problemov [11]. Med bolj znanimi strategijami so:

- deli in vladaj,
- dinamično programiranje,
- požrešna metoda,
- linearno programiranje,
- verjetnostni algoritmi in
- lokalna optimizacija.

Poglavje 3

LAPLACEOVA MATRIKA

Laplaceova matrika je tesno povezana z matriko sosednosti grafa Γ in nam prav tako pove določene informacije o grafu. Uporabna je pri naključnih sprehodih v grafu, električnih omrežjih z uporniki, particiji grafa, ugotavljanju povezanosti grafa itd. V tem poglavju si bomo pogledali, kaj je Laplaceova matrika, ter njene lastnosti.

Definicija 3.1. *Naj bo Γ graf ter A pripadajoča matrika sosednosti. Matrika*

$$D_{i,j} = \begin{cases} k_i, & \text{če je } i = j, \text{ in} \\ 0, & \textit{sicer} \end{cases}$$

je diagonalna matrika s stopnjami točk na glavni diagonali. Laplaceova matrika L pa je definirana kot

$$L = D - A,$$

oziroma

$$L_{i,j} = \begin{cases} k_i, & \text{če je } i = j, \\ -1, & \text{če je } (i,j) \in E(\Gamma), \text{ in} \\ 0, & \text{sicer.} \end{cases}$$

Laplaceova matrika je simetrična matrika in ima zato samo realne lastne vrednosti. Izkaže se celo, da so vse njene lastne vrednosti nenegativne. Recimo, da imamo graf Γ na n točkah in m povezavah. Brez škode za splošnost bomo eno od krajišč vsake povezave označili z 1 ter drugo krajšče z 2. Ni pomembno, katero krajišče označimo z 1 in katero z 2, važno je samo, da ju drugače označimo. Definirajmo $m \times n$ matriko B na naslednji način:

$$B_{i,j} = \begin{cases} 1, & \text{če je krajišče povezave } i \text{ označeno z 1 enako točki } j, \\ -1, & \text{če je krajišče povezave } i \text{ označeno z 2 enako točki } j, \text{ in} \\ 0, & \text{sicer.} \end{cases}$$

Vsaka vrstica matrike B ima natanko eno vrednost enako 1 in eno -1. Matriki B pravimo tudi incidenčna matrika povezav. Poglejmo si sedaj vsoto $\sum_k B_{k,i}B_{k,j}$.

Če $i \neq j$, potem so edini neničelni sumandi tisti, ki imajo tako $B_{k,i}$ kot $B_{k,j}$ različna od 0. V tem primeru bo produkt $B_{k,i}B_{k,j}$ enak -1, saj povezava k povezuje točki i in j. V grafu imamo lahko največ eno povezavo med točkama in zato je celotna vsota enaka -1, če obstaja povezava med točkama i in j, in 0 sicer.

Če je $i = j$, potem ima vsota $\sum_k B_{k,i}B_{k,j} = \sum_k B_{k,i}^2$ sumand 1 za vsako točko, ki je povezana z i. Torej je $\sum_k B_{k,i}^2 = k_i$.

Vsota $\sum_k B_{k,i}B_{k,j}$ je torej enaka elementom Laplaceove matrike $\sum_k B_{k,i}B_{k,j} = L_{i,j}$. V matrični obliki lahko to zapišemo kot:

$$L = B^\top B.$$

Naj bo v_i lastni vektor matrike L z lastno vrednostjo λ_i. Potem

$$v_i^\top B^\top B v_i = v_i^\top L v_i = \lambda_i v_i^\top v_i = \lambda_i,$$

kjer predpostavimo, da je lastni vektor v_i normaliziran.

Lastna vrednost λ_i Laplaceove matrike je enaka $(v_i^\top B^\top)(Bv_i)$, kar je enako skalarnemu produktu vektorja (Bv_i) s seboj. Od tod sledi, da λ_i ne more biti negativna:

$$\lambda_i \geq 0$$

za vsak i.

Izkaže se, da imamo vselej vsaj eno lastno vrednost Laplaceove matrike L enako 0. Poglejmo si produkt matrike L z vektorjem $1 = (1, 1, 1, ...)$. Element na i-ti poziciji je enak

$$\sum_j L_{i,j} \times 1 = \sum_j (\delta_{i,j}k_i - A_{i,j}) = k_i - \sum_j A_{i,j} = k_i - k_i = 0\,,$$

kjer je $\delta_{i,j}$ Krocknerjev delta

$$\delta_{i,j} = \begin{cases} 1, & \text{če je } i = j, \text{ in} \\ 0, & \text{sicer.} \end{cases}$$

Torej je $L \cdot 1 = 0$. Vektor 1 je zmeraj lastni vektor Laplaceove matrike L z lastno vrednostjo 0.

Trditev 3.2. *Naj bo Γ graf in L njegova Laplaceova matrika. Matrika L ima nenegativne lastne vrednosti in ima vselej vsaj eno ničelno lastno vrednost.*

Recimo, da ima graf Γ c komponent velikosti $n_1, n_2, ..., n_c$. Za lažjo notacijo označimo točke tako, da je prvih n_1 točk v prvi komponenti, naslednjih n_2 točk v drugi komponenti itd. S tako izbiro bo Laplaceova matrika bločno diagonalna in izgledala tako:

$$L = \begin{bmatrix} \square & 0 & \cdots \\ 0 & \square & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

Vsak blok je Laplaceova matrika pripadajoče komponente. Zato lahko takoj napišemo c različnih vektorjev, ki imajo 0 za lastno vrednost. Ti vektorji imajo enice na mestih, ki pripadajo točkam v izbrani komponenti grafa, in ničle drugod.

Sledi, da ima graf s c komponentami vsaj c ničelnih lastnih vrednosti. Izkaže se, da ima povezan graf natanko eno lastno vrednost enako 0 in zato ima graf s c komponentami natanko c lastnih vrednosti enakih 0. Drugi najmanjši lastni vrednosti Laplaceove matrike včasih pravimo tudi *algebraična povezanost* grafa, saj v kolikor je le-ta 0, je graf nepovezan.

Poglavje 4

PARTICIJA GRAFA, ODKRIVANJE SKUPNOSTI IN MAKSIMALEN PREREZ

Veliko je razlogov, zakaj bi razdelili graf na več skupin (podgrafov). Obstajajo tri splošne vrste delitev, ki posledično privedejo do treh različnih tipov optimizacijskih problemov in računalniških algoritmov. Te vrste delitev se imenujejo *particija grafa*, *odkrivanje skupnosti* in *maksimalni prerez*. Particija grafa in odkrivanje skupnosti se razlikujeta v tem, ali je velikost skupin v particiji fiksna ali ni. Pri obeh si želimo imeti čim manjše število povezav med skupinami. Pri problemu maksimalnega prereza delimo na dve skupini poljubne velikosti in želimo imeti čim večje število povezav med skupinama. Rešitve za dane probleme se uporablja za različne probleme v praksi.

4.1 Particija grafa

Definicija 4.1. *Particija P množice X je množica podmnožic X, tako da je vsak element množice X vsebovan v natanko eni izmed teh podmnožic.*

Particija grafa je klasičen problem računalniške znanosti že od 60ih dalje. Je problem delitve (particije) točk grafa v več disjunktnih skupin tako, da je število povezav (velikost prereza) med skupinami najmanjše možno. Tu je pomembno, da je število skupin in njihove velikosti že vnaprej določene. Včasih niso določene, so pa v določenih mejah.

Recimo, da imamo računalnik z dvemi (ali več) procesorji. Da pospešimo določene operacije, je smiselno nalogo razdeliti na oba procesorja. V tem primeru pride particija grafa v poštev, da nalogo pravilno razdeli na procesorje, ki potem določen račun izvajajo samo na manjšem delu grafa. Podobno se lahko nalogo razdeli tudi na več računalnikov. Ponavadi želimo vsakemu procesorju (računalniku) pripisati približno enako število točk grafa.

Problem nastane, v kolikor graf ni sestavljen iz več komponent, tako da imajo točke na enem procesorju tudi sosede na drugem procesorju. Za končno rešitev je potrebno izmenjevanje podatkov med procesorji in to lahko precej upočasni celoten račun. Zato je

pomembno, da poskusimo razdeliti graf tako, da bo čim manj povezav med skupinami. Tako bo medprocesorske komunikacije manj, kar bo pospešilo račun.

V naslednjih dveh poglavjih si bomo pogledali dve znani metodi za bisekcijo grafa. Prva metoda, algoritem Kernighan-Lin, ne temelji na matrični metodi. Druga metoda, spektralna particija, pa temelji na lastnostih Laplaceove matrike grafa.

Najbolj osnoven primer particije grafa je razdelitev grafa na dve skupini. Delitvi na dve skupini pravimo tudi *bisekcija grafa*. Večina algoritmov za particijo grafa so v bistvu algoritmi za bisekcijo grafa. To ni problem, saj se lahko bisekcijo ponavlja, dokler nimamo želenega števila delov.

Definicija 4.2. *Naj bo P particija točk grafa Γ. Velikost prereza particije P je število povezav med množicami v P. Označimo jo z R.*

Kot smo že povedali, želimo poiskati particijo, ki ima najmanjšo velikost prereza R. Ta naloga nima enostavne rešitve. Lahko bi pogledali vse možne delitve grafa in izbrali najboljšo. Temu principu pravimo *izčrpno iskanje* in deluje zadovoljivo le na zelo majhnih grafih. Največja slabost izčrpnega iskanja je, da je zelo počasno.

Število načinov, kako razdelimo graf na n točkah v dve skupini velikosti n_1 in n_2, je enako $n!/(n_1!n_2!)$. Če aproksimiramo fakulteto z uporabo Strilingove formule $n! \simeq \sqrt{2\pi n}(n/e)^n$ in uporabimo dejstvo, da je $n_1 + n_2 = n$, dobimo

$$\frac{n!}{n_1!n_2!} \simeq \frac{\sqrt{2\pi n}(n/e)^n}{\sqrt{2\pi n_1}(n_1/e)^{n_1}\sqrt{2\pi n_2}(n_2/e)^{n_2}} \simeq \frac{n^{n+1/2}}{n_1^{n_1+1/2}n_2^{n_2+1/2}}.$$

Če bi na primer želeli graf razdeliti na dva dela enake velikosti $\frac{1}{2}n$, potem bi morali preveriti približno

$$\frac{n^{n+1/2}}{(n/2)^{n+1}} = \frac{2^{n+1}}{\sqrt{n}}$$

načinov. Torej bo časovna zahtevnost eksponentna, kar ni praktično za algoritme. To nas prisili, da poiščemo bolj domiselno rešitev. Obstaja utemeljen sum, da ne bomo nikoli našli polinomskega algoritma, ki bi nam vrnil optimalno rešitev za vsak graf. Ostanejo nam samo algoritmi, ki delujejo hitro in nam vrnejo približek, in algoritmi, ki nam vrnejo optimalno rešitev, vendar potrebujejo nepraktično dolgo časa. Pametni algoritmi, ki nam vrnejo približen rezultat, so še vedno zelo koristni. Približen rezultat je vsekakor boljši, kot da bi vzeli neko naključno delitev.

4.2 Odkrivanje skupnosti

Odkrivanje skupnosti se od particije grafa razlikuje v tem, da število in velikost skupin ni fiksno, temveč je (implicitno) določeno z grafom samim. Cilj odkrivanja skupnosti je, da najde naravno delitev grafa. Velikost skupin je nedoločena in se lahko precej razlikuje od grafa do grafa. Določen graf se lahko razdeli na nekaj velikih skupin, več manjših ali oboje.

Odkrivanje skupnosti se uporablja za analizo in razumevanje grafov. Če imamo graf neke organizacije, nam lahko odkrivanje skupnosti pomaga pri razumevanju, kako je ta

organizacija zgrajena in kako deluje. V internetnem omrežju nam lahko pove, katere spletne strani so si med seboj tesno vsebinsko povezane.

Odkrivanje skupnosti je manj opredeljen problem od particije grafa. Njegov cilj je najti naravno delitev grafa v skupine tako, da je veliko povezav znotraj posamezne skupine in malo med skupinami. Kaj točno imamo v mislih kot "malo" in "veliko",

je relativno in obstaja več možnih definicij, ki posledično vodijo do veliko različnih algoritmov za odkrivanje skupnosti. V magistrskem delu se bomo osredotočili na najbolj razširjeno formulacijo problema, ki je *optimizacija modularnosti*. *Modularnost* je mera za dobro ali slabo delitev grafa.

Poglejmo si najbolj osnoven in najlažji problem odkrivanja skupnosti. To je delitev grafa na dve skupini oz. skupnosti (tokrat nedoločenih velikosti, disjunktnih in katerih vsota je enaka n). Torej je v najpreprostejšem primeru število skupin vseeno določeno, velikost pa ni. Naša prva ideja, kako rešiti problem je, da najdemo particijo tako, da bo velikost prereza najmanjša. Ta pristop ne bi deloval, saj bi algoritem dal vse točke v eno skupino in nobene v drugo. Slednja trivialna rešitev zagotovi, da je velikost prereza enaka 0. Ta rezultat pa seveda ni uporaben.

Boljši pristop bi bil, da bi nekoliko omejili velikosti skupnosti. Torej bi dovolili delno spreminjanje velikosti skupnosti do neke mere. Primer takega pristopa je *particija z razmerjem prereza* (angl. ratio cut partitioning), v katerem namesto minimiziranja velikosti prereza R, minimiziramo razmerje $R/(n_1 n_2)$, kjer sta n_1 in n_2 velikosti skupnosti. Imenovalec $n_1 n_2$ ima največjo vrednost, ko sta $n_1 = n_2 = \frac{1}{2}n$, in takrat najbolj zmanjša razmerje. To izključi rešitev, ko je velikost ene od skupin enaka 0 in teži bolj k rešitvi, ko sta velikosti skupnosti približno enaki.

Razmerje prereza, kot orodje za naravno delitev grafa, ni idealno. Čeprav dovoljuje različne velikosti skupnosti, še vedno teži k temu, da bosta velikosti skupnosti približno enaki. Ni nobene racionalne razlage, zakaj bi razmerje prereza delovalo pravilno. Vseeno deluje razumno dobro v nekaterih primerih, ampak ne moremo vedeti, ali bomo dobili dober rezultat, ali bi nam kak drugi pristop vrnil boljšega.

Alternativni pristop k reševanju odkrivanja skupnosti je, da se poslužimo mere za kvaliteto prereza, ki ne sloni na velikosti prereza in njenim variantam. Trdi se, da velikost prereza ni dobra mera, saj dobra delitev omrežja ne temelji le na majhnem številu povezav med skupnostmi. Trdi se, da je dobra delitev taka, kjer je manj povezav, kot se jih pričakuje. Če najdemo prerez, ki ima število povezav približno enako številu povezav v prerezu naključno generiranega grafa, potem lahko zagotovo rečemo, da nismo našli nobene posebne delitve. Ni važna minimalna velikost prereza, temveč koliko se velikost prereza razlikuje s pričakovano velikostjo prereza.

Konvencionalen razvoj te ideje obravnava število povezav v skupnostih in ne obratno. Oba pristopa sta ekvivalentna, saj vsaka povezava znotraj skupnosti ne leži med skupnostmi in zato lahko iz tega izračunamo, koliko je povezav med skupnostmi in obratno. Sledili bomo tej konvenciji v računih števila povezav znotraj skupnosti.

Naš cilj bo najti mero, ki nam pove, koliko povezav leži v skupinah glede na pričakovano število povezav v skupinah, ki temelji na verjetnosti. Modularnost pogostno definiramo kot število povezav znotraj skupin minus pričakovano število povezav znotraj skupin. Dobra delitev grafa v skupnosti bo tista, ki bo imela visoko vrednost te modularnosti.

To je tudi najbolj uporabljen pristop za reševanje odkrivanja skupnosti. Podobno

kot pri particiji grafa je maksimizacija modularnosti težek problem. Meni se, da edini algoritmi, ki točno rešijo problem z maksimizacijo modularnosti, potrebujejo eksponentno mnogo časa. Zato se tudi tukaj poslužimo uporabe hevristik.

4.3 Maksimalen prerez

Problem maksimalnega prereza spašuje po delitvi točk grafa $\Gamma = (V, E)$ na dve skupini S in S' ($V = S \cup S'$, $S \cap S' = \emptyset$), pri čem tokrat želimo imeti čim večje število povezav med skupinama.

- Par (S, S') imenujemo prerez.
- Število povezav z enim krajiščem v S in drugim v S' imenujemo *vrednost* (velikost) *prereza*.

Poleg teoretične pomembnosti ima problem maksimalnega prereza aplikacije v oblikovanju postavitve vezij in v statistični fiziki.

Problem maksimalnega prereza je rešljiv v polinomskem času za naslednje družine grafov:

- dvodelni grafi (grafi, katerih točke lahko razdelimo v dve skupini, kjer znotraj skupin ni povezav),
- ravninski grafi (grafi, ki jih lahko narišemo v ravnino, tako da se nobeni povezavi ne sekata).

Problem maksimalnega prereza je NP-težek problem za splošne grafe. Za njegovo rešitev se poslužimo ρ-aproksimacijskih algoritmov. To je eden izmed prvih problemov, za katerega je bila pokazana NP-težkost (leta 1972). Za uteženo različico problema je bila pokazana NP-težkost leta 1976. Ponovimo definicijo aproksimacijskega algoritma. Naj bo Π maksimizacijski problem in A algoritem polinomske časovne zahtevnosti za Π ter $0 < \rho \leq 1$. Pravimo, da je algoritem A ρ-aproksimacijski, če velja

$$ALG \geq OPT \cdot \rho\,,$$

kjer je OPT vrednost optimalne rešitve in ALG vrednost rešitve, ki jo vrne algoritem A.

Za problem maksimalnega prereza obstajata preprosta $\frac{1}{2}$-aproksimacijska algoritma. Prvi algoritem temelji na verjetnosti, drugi pa na lokalni optimizaciji.

Verjetnostni algoritem: Vozlišča grafa neodvisno in enakomerno naključno porazdelimo bodisi v S ali v S'.

Vsaka točka ima verjetnost $\frac{1}{2}$, da je dodeljena v S, in verjetnost $\frac{1}{2}$, da je dodeljena v S'. Verjetnost, da ima povezava (x, y) krajišči v različnih skupinah je enaka $\frac{1}{2}$. Sledi, da je matematično upanje enako $\mathbb{E}(ALG) = \sum_{(x,y)} \frac{1}{2} = \frac{|E|}{2}$. Matematično upanje je torej enako

$$\mathbb{E}(ALG) = \frac{|E|}{2} \geq \frac{1}{2} OPT.$$

Deterministični algoritem (lokalna optimizacija): Začni s poljubnim prerezom. Ponavljaj, dokler lahko: če obstaja vozlišče v, ki ima v drugi skupini več kot polovico svojih sosedov, ga premakni v drugo skupino. Tudi ta algoritem je $\frac{1}{2}$-aproksimacijski.

Naj $E(S)$ označuje povezave, ki imajo obe krajišči v množici S (podobno za S').

$$|E(S)| = \frac{1}{2}\sum_{v\in S} k_S(v) \leq \frac{1}{2}\sum_{v\in S} k_{S'}(v) = \frac{1}{2}ALG$$

$$|E(S')| \leq \frac{1}{2}ALG$$

$$OPT \leq |E| = ALG + |E(S)| + |E(S')| \leq 2ALG$$

$$ALG \geq \frac{1}{2}OPT$$

Število $k_S(v)$ predstavlja število sosedov točke v znotraj skupine S.

20 let ni bilo nobene izboljšave, dokler nista Goemans in Williamson leta 1994 objavila članek [6], v katerem sta predstavila 0.87856-aproksimacijski algoritem. Slednji bo obravnavan v poglavju 12.

4.4 Formalna opredelitev problemov, s katerimi se bomo soočili

4.4.1 Problem particije grafa

Dan je graf $\Gamma = (V, E)$ ter vrednosti $(s_1, s_2, ..., s_l)$, ki označujejo velikosti skupin v katere želimo graf razdeliti ($s_1 + s_2 + ... + s_l = |V|$). Razdeli množico vozlišč V na l delov (skupin) $V = c_1 + c_2 + \ldots + c_l$, tako da bo i-ta skupina velikosti s_i ($|c_i| = s_i$ $i \in \{1, 2, ..., l\}$) in da bo število povezav med skupinami minimalno.

Za ta problem ne bomo spoznali nobenega direktnega algoritma, spoznali pa bomo hevristične algoritme za bisekcijo grafa (ki je podproblem particije grafa). Z večkratno bisekcijo grafa pa lahko pridemo do približne rešitve za problem particije grafa.

4.4.2 Problem bisekcije grafa

Dan je graf $\Gamma = (V, E)$ ter vrednosti (s_1, s_2), ki označujejo velikosti dveh skupin v katere želimo graf razdeliti ($s_1 + s_2 = |V|$). Razdeli množico vozlišč V na dva dela (skupini) $V = c_1 + c_2$, tako da bo skupina c_1 velikosti s_1 ($|c_1| = s_1$), skupina c_2 velikosti s_2 ($|c_2| = s_2$) in da bo število povezav med obema skupinama minimalno.

Ta problem je enak zgornjemu za $l = 2$. V poglavjih 5 in 6 si bomo pogledali dva algoritma za problem bisekcije grafa: algoritem Kernighan-Lin in spektralno partcijo.

4.4.3 Problem odkrivanja skupnosti

Dan je graf $\Gamma = (V, E)$. Razdeli graf Γ na poljubno mnogo delov (skupin, skupnosti) poljubne velikosti glede na njegovo "naravno" delitev. Kaj privzamemo za naravno delitev,

je odvisno od algoritma do algoritma. Nekateri algoritmi odstranjujejo povezave, ki so pomembne (na primer ležijo na veliko najkrajših poteh ali pa ne vsebujejo kratih ciklov), drugi združujejo točke grafa v vse večje skupine glede na to, katere povezave so pomembne. Ti algoritmi se poslužujejo mere za "pomembnost" povezav in vrnejo kot rezultat hierarhično dekompozicijo grafa. Druga veja algoritmov se poslužuje modularnosti, ki je mera za dobro ali slabo delitev grafa. Preko maksimizacije modularnosti ti algoritmi pridejo do delitve za dan graf.

Spoznali bomo dva algoritma, ki slonita na modularnosti v poglavjih 7 in 8, ter dve ideji algoritma v poglavju 8. Ti algoritmi so omejeni na delitev grafa na dve skupnosti poljubne velikosti. Tudi tukaj lahko zaporedoma uporabimo algoritem za delitev grafa na več skupnosti. V poglavju 9 si bomo pogledali, kako iz teh algoritmov razdelimo graf na več skupnosti. Algoritmi, ki slonijo na merjenju "pomembnosti" povezav, se bodo pojavili v poglavjih 10 in 11.

4.4.4 Problem maksimalnega prereza

Dan je graf $\Gamma = (V, E)$. Razdeli množico vozlišč V na dva dela, $V = S \cup S'$, $S \cap S' = \emptyset$, tako da bo število povezav z enim krajiščem v S in drugim v S' maksimalno.

Obstaja tudi uteženа verzija problema, kjer ima vsaka povezava neko utež in pri kateri iščemo prerez največje skupne teže. Osredotočili se bomo le na neuteženo verzijo, rezultati pa veljajo tudi za uteženo verzijo.

V poglavju 12 si bomo pogledali ρ-aproksimacijski algoritem s faktorjem aproksimacije $\rho = 0,87856$.

Poglavje 5

ALGORITEM KERNIGHAN-LIN

Algoritem Kernighan-Lin je eden najpreprostejših in najbolj poznanih hevristik za problem bisekcije grafa. Ta algoritem sta predlagala Brian Kernighan in Shen Lin leta 1970 [16].

Algoritem začne tako, da razdeli graf na dve skupini zahtevane velikosti. To lahko naredimo naključno. Potem za vsak tak par (i, j) točk, da sta i in j vsak v svoji skupini, izračunamo, za koliko bi se velikost prereza razlikovala, če bi točki i in j zamenjali med seboj. Nato izmed vseh parov točk izberemo tak par, pri katerem se velikost prereza najbolj zmanjša oz. najmanj poveča, ter točki zamenjamo. Ta proces očitno ohranja velikost skupin.

Ta proces ponavljamo, vendar s pogojem, da lahko posamezno točko premaknemo samo enkrat. Ko ne moremo več nobenega para točk zamenjati, prenehamo. Nato pogledamo nazaj skozi vsako stanje algoritma in med njimi izberemo tistega, ki ima najmanjšo velikost prereza.

Celoten proces ponovimo z razliko v tem, da namesto naključne začetne particije grafa kot začetno postavitev, izberemo nazadnje dobljeno rešitev. Ponavljamo dokler ni moč najti nobene izboljšave.

Algoritem Kernighan-Lin lahko uporabljamo tudi za particijo grafa na tri ali več skupin. Najprej razdelimo graf na dve skupini in nato na eni izmed njiju ponovimo algoritem Kernighan-Lin in tako pridemo do treh podgrafov. Četudi bi se zgodilo, da bi pri obeh bisekcijah grafa dobili najbolj optimalno rešitev, ni zagotovila, da dobimo optimalno rešitev za delitev grafa na tri skupine.

Glede na začetno delitev grafa na dve skupini lahko dobimo različne končne delitve. Zato je včasih koristno večkrat pognati algoritem za različne začetne delitve grafa in med njimi izbrati najboljšo.

Slabost algoritma Kernighan-Lin je v tem, da je počasen. Število zamenjav točk med enim krogom algoritma je enako moči manjše izmed obeh skupin, kar je med 0 in $\frac{n}{2}$. Torej je v najslabšem primeru $O(n)$ zamenjav. Pred vsako zamenjavo moramo tudi preveriti, koliko se velikost prereza spremeni pri zamenjavi para točk iz različnih skupin. Število parov je v najslabšem primeru enako $\frac{n}{2} \cdot \frac{n}{2} = O(n^2)$. Poleg tega je potrebno za vsak par točk izračunati, za koliko se ob njuni zamenjavi velikost prereza spremeni.

Ko točko i premaknemo iz ene skupine v drugo, povezave, ki jo povezujejo s točkami v trenutni skupini, postanejo povezave med skupinama. Naj bo k_i^{ista} število teh povezav. Povezave, ki jih ima točka i z drugo skupino, katerih je k_i^{druga}, postanejo povezave znotraj

druge skupine, ko točko i premaknemo. Če točko i menjamo s točko j in obstaja povezava med i in j, potem ta povezava ostane povezava med skupinama. Sprememba velikosti prereza δ je enaka:

$$\delta = k_i^{druga} - k_i^{ista} + k_j^{druga} - k_j^{ista} - 2A_{i,j}.$$

Za graf prikazan v obliki seznama sosedov, je ocena za časovno zahtevnost tega izračuna enaka $2 \cdot$ povprečna valenca točk grafa, kar je enako $O(m/n)$.

Skupna časovna zahtevnost je enaka $O(n \cdot n^2 \cdot m/n) = O(mn^2)$, kar je enako $O(n^3)$ v redkih grafih in $O(n^4)$ v gostih grafih. To je že samo po sebi počasno, vendar še nismo končali. Potrebno je še pomnožiti s številom ponovitev algoritma, ki jih naredimo, ko uporabimo nov začetni približek za prvotno delitev grafa. Kolikokrat se algoritem ponovi, je zelo odvisno od grafa. Za grafe do nekaj tisoč točk se algoritem v povprečju ponovi od petkrat do desetkrat, za večje grafe pa se ne ve dobro, saj je algoritem prepočasen za analizo. Vseeno se zdi smiselno, da se na večjih grafih algoritem večkrat ponovi. Algoritem lahko malo pospešimo z uporabo nekaj trikov. Če začetno shranimo število sosedov k_i^{ista} in k_i^{druga}, ter jih posodobimo, vsakič ko premikamo točke, potem si prihranimo čas za preračunavanje teh vrednosti na vsakem koraku. Če imamo graf podan v obiki matrike sosednosti, potem lahko ugotovimo povezanost dveh točk v času $O(1)$. Vse skupaj nam dovoli izračunati δ v času $O(1)$ in izboljša celoten algoritem do $O(n^3)$. Na splošno je algoritem precej počasen. S časovno zahtevnostjo $O(n^3)$ je algoritem primeren za grafe z do nekaj tisoč točkami.

Poglavje 6

SPEKTRALNA PARTICIJA

Spektralna particija je široko uporabljena metoda, ki jo je razvil *Fiedler*. Algoritem temelji na lastnostih Laplaceove matrike L. Spektralna particija se uporablja za rešitev problema bisekcije grafa. Metodo lahko uporabimo tudi za delitev grafa na več kot dva dela preko večkratne bisekcije grafa.

Naj bo Γ graf na n točkah in m povezavah. Graf želimo razdeliti na dve skupini, ki jima bomo rekli prva skupina in druga skupina. Velikost prereza R med skupinama lahko opišemo kot

$$R = \frac{1}{2} \sum_{i,j \text{ v različnih skupinah}} A_{i,j}, \tag{6.1}$$

kjer dodamo faktor $\frac{1}{2}$ zato, da izravna dvojno štetje povezav.

Definirajmo množico količin s_i, za vsako točko i, ki ponazarja delitev grafa:

$$s_i = \begin{cases} 1, & \text{če je točka } i \text{ v prvi skupini, in} \\ -1, & \text{če je točka } i \text{ v drugi skupini.} \end{cases}$$

Potem je

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 0, & \text{če sta točki } i \text{ in } j \text{ v isti skupini, in} \\ 1, & \text{če sta točki } i \text{ in } j \text{ v različnih skupinah.} \end{cases}$$

Enačbo (6.1) lahko zato prepišemo kot

$$R = \frac{1}{4} \sum_{i,j} A_{i,j}(1 - s_i s_j), \tag{6.2}$$

kjer gresta tokrat spremenljivki i in j po vseh točkah grafa. Preko uporabe enakosti $\sum_j A_{i,j} = k_i$ in $s_i^2 = 1$ pridemo do sledečega rezultata:

$$\sum_{i,j} A_{i,j} = \sum_i k_i = \sum_i k_i s_i^2 = \sum_{i,j} k_i \delta_{i,j} s_i s_j.$$

Če to vstavimo v enačbo (6.2), ugotovimo:

$$R = \frac{1}{4} \sum_{i,j} (k_i \delta_{i,j} - A_{i,j}) s_i s_j = \frac{1}{4} \sum_{i,j} L_{i,j} s_i s_j. \tag{6.3}$$

To lahko zapišemo tudi v matrični obliki

$$R = \frac{1}{4} s^\top L s, \tag{6.4}$$

kjer je s vektor z elementi s_i. Izraz nam poda enačbo v matrični obliki za problem bisekcije grafa. Matrika L določa strukturo našega grafa, vektor s pa določa delitev grafa. Cilj je najti vektor s, ki minimizira R.

Minimizacija je na splošno težek problem (najverjetneje ni polinomski). Vrednosti s_i ne morejo zavzeti katerekoli vrednosti. Omejeni smo na ± 1. Če bi lahko zavzele katerekoli vrednosti, bi bila optimizacija lažja.

Zato se za ta problem poslužujemo hevristike. Recimo, da bi lahko vrednosti s_i zavzele katerekoli vrednosti (z nekaterimi omejitvami, ki jih bomo povedali kasneje) in nato našli vrednosti, ki minimizirajo R. Ta razmislek nas pripelje do t.i. *sprostitvene metode*.

Omejitve za vrednosti s_i sta v bistvu dve. Prva omejitev je ta, da vrednosti s_i lahko zavzamejo le vrednosti ± 1. Če pomislimo na s kot na vektor, potem s vedno kaže k ogliščem n-dimenzionalne kocke in ima vselej dolžino $\sqrt{n}$. Ta pogoj bomo sprostili tako, da mora vektor s samo imeti dolžino $\sqrt{n}$:

$$\sum_i s_i^2 = n. \tag{6.5}$$

Druga omejitev vektorja s_i je ta, da je število vrednosti vektorja enakih 1 enako željeni velikosti prve skupine in število vrednosti vektorja enakih -1 enako željeni velikosti druge skupine. Naj bosta n_1 in n_2 željeni velikosti skupin, na katere delimo graf.

Drugi pogoj lahko zapišemo kot

$$\sum_i s_i = n_1 - n_2, \tag{6.6}$$

oz. z uporabo vektorjev

$$1^\top s = n_1 - n_2, \tag{6.7}$$

kje je $1 \in \mathbb{R}^n$ vektor samih enic $(1, 1, \ldots, 1)$. Drugi pogoj pustimo nespremenjen. Problem bisekcije grafa v sproščeni obliki je problem minimizacije velikosti prereza (6.4) pri pogojih (6.5) in (6.6).

Ta problem se sedaj lahko reši s pomočjo vezanih ekstremov. Odvajali bomo po elementih s_i, z Langrangejevimi multiplikatorji, ki jih bomo označili z λ in 2μ:

$$\frac{\partial}{\partial s_i}\left[\sum_{j,k} L_{j,k} s_j s_k + \lambda(n - \sum_j s_j^2) + 2\mu((n_1 - n_2) - \sum_j s_j)\right] = 0.$$

Po odvajanju ugotovimo, da je

$$\sum_j L_{i,j} s_j = \lambda s_i + \mu,$$

oz. v matrični obliki

$$Ls = \lambda s + \mu 1. \tag{6.8}$$

Spomnimo se, da je vektor 1 lastni vektor Laplaceove matrike z lastno vrednostjo 0, torej $L \cdot 1 = 0$. Pomnžimo enačbo (6.8) z vektorjem $1^\top$ z leve in z uporabo enačbe (6.7) ugotovimo, da je $\lambda(n_1 - n_2) + \mu n = 0$, oz.

$$\mu = -\frac{n_1 - n_2}{n}\lambda.$$

Definirajmo nov vektor

$$x = s + \frac{\mu}{\lambda}1 = s - \frac{n_1 - n_2}{n}1, \tag{6.9}$$

potem nam enčba (6.8) pove

$$Lx = L(s + \frac{\mu}{\lambda}1) = Ls = \lambda s + \mu 1 = \lambda x, \tag{6.10}$$

kjer smo ponovno uporabili lastnost $L \cdot 1 = 0$.

Z drugimi besedami, x je lastni vektor Laplaceove matrike z lastno vrednostjo λ. Na tem koraku imamo izbiro, kateri lastni vektor vzamemo za x. Vsak lastni vektor bo zadoščal enačbi (6.10) in očitno moramo izbrati tistega, ki vrne najmanjšo velikost prereza R. Opazimo, da je vektor x pravokotnen na vektor 1:

$$1^\top x = 1^\top s - \frac{\mu}{\lambda}1^\top 1 = (n_1 - n_2) - \frac{n_1 - n_2}{n}n = 0.$$

To pomeni da, četudi x mora biti lastni vektor, ne more biti vektor 1. Potem katerega naj izberemo? Opazimo, da je

$$R = \frac{1}{4}s^\top Ls = \frac{1}{4}x^\top Lx = \frac{1}{4}\lambda x^\top x.$$

Če slednje uporabimo v enačbi (6.9), dobimo

$$\begin{aligned} x^\top x &= s^\top s + \frac{\mu}{\lambda}(s^\top 1 + 1^\top s) + \frac{\mu^2}{\lambda^2}1^\top 1 && (6.11)\\ &= n - 2\frac{n_1 - n_2}{n}(n_1 - n_2) + \frac{(n_1 - n_2)^2}{n^2}n && (6.12)\\ &= 4\frac{n_1 n_2}{n}, && (6.13) \end{aligned}$$

iz česar sledi enačba

$$R = \frac{n_1 n_2}{n}\lambda.$$

Ugotovili smo, da je velikost prereza sorazmerna z lastno vrednostjo λ. Ker je naš cilj minimizirati velikost prereza R, moramo izbrati vektor x, ki ima najmanjšo dovoljeno lastno vrednost Laplaceove matrike L. Laplaceova matrika L ima vse lastne vrednosti nenegativne. Vektor 1 je lastni vektor z lastno vrednostjo 0. Ker $x \neq 1$ in predpostavimo, da je graf povezan, sledi, da moramo izbrati drugo najmanjšo lastno vrednost za λ. Nato, ko izračunamo x, dobimo vrednosti s_i iz spodnje enačbe

$$s = x + \frac{n_1 - n_2}{n}1, \tag{6.14}$$

ki nam vrne optimalno sproščeno vrednost za s.

Kot smo že povedali, ima pravi vektor s dodatne pogoje ($s_i = \pm 1$, n_1 vrednosti vektorja s je enakih 1 in n_2 vrednosti je enakih -1). Običajno nam bodo ti pogoji preprečevali vzeti točno vrednost iz enačbe (6.14). Vseeno bomo poskušali izbrati s tako, da bo čim bližje relaksirani rešitvi, kar dosežemo tako, da produkt

$$s^\top (x + \frac{n_1 - n_2}{n} 1) = \sum_i s_i (x_i + \frac{n_1 - n_2}{n})$$

naredimo kar se le da velikega. Maksimum tega izraza dosežemo tako, da pripišemo $s_i = 1$ tistim točkam, ki imajo največje vrednosti $x_i + (n_1 - n_2)/n$ in $s_i = -1$ preostalim.

Opazimo, da najbolj pozitivne vrednosti izraza $x_i + (n_1 - n_2)/n$ so hkrati najbolj pozitivne vrednosti za x_i, ki so tudi najbolj pozitivni elementi lastnega vektorja v_2 (lastni vektor z drugo najmanjšo lastno vrednostjo), ki je sorazmeren z vektorjem x. Ko poračunamo v_2, damo prvih n_1 točk z največjimi vrednostmi v prvo skupino in ostale v drugo.

Vseeno je, katero od obeh skupin označimo za prvo ali drugo. Zato dobimo dve možnosti delitve grafa. Prva možnost je, da damo v prvo skupino točke, ki pripadajo n_1 najbolj pozitivnim vrednostim. Druga možnost je, da damo v prvo skupino točke, ki pripadajo n_2 najbolj pozitivnim vrednostim. Dve rešitvi dobimo samo v primeru, ko $n_1 \neq n_2$. Da dobimo pravo rešitev, samo poračunamo velikost prereza za obe možnosti ter izberemo tisto, ki ima manjšo velikost prereza.

Na kratko povzemimo celoten algoritem:

1. Izračunaj lastni vektor v_2 z drugo najmanjšo lastno vrednostjo λ_2 Laplaceove matrike L.

2. Razporedi elemente v vektorju v_2 od največjega proti najmanšemu.

3. Prvih n_1 točk, ki pripadajo največjim elementom v vektorju v_2, daj v prvo skupino, ostale v drugo skupino.

4. Prvih n_2 točk, ki pripadajo največjim elementom v vektorju v_2, daj v prvo skupino, ostale v drugo skupino.

5. Izmed obeh delitev grafa izberi tisto, ki ima manjšo velikost prereza.

Prednost spektralne particije je hitrost. Za izračun v_2 je potreben čas $O(mn)$ preko ortogonalizacijske metode ali Lanczosove metode. To je za cel faktor n boljše od algoritma Kernighan-Lin in ta algoritem je zato primernejši za večje grafe z do stotisočimi točkami.

Če je graf povezan, je druga najmanjša lastna vrednost Laplaceove matrike neničelna. Iz tega razloga ji tudi pravimo algebraična povezanost grafa. Kaj naredimo, če je druga najmanjša lastna vrednost enaka 0 (in pri tem graf ni povezan)? V tem primeru so obe najmanjši lastni vrednosti enaki 0 in pripadajoča lastna vektorja sta nedoločena (katerakoli linearna kombinacija lastnih vektorjev z enako lastno vrednostjo je tudi lastni vektor). To ni resen problem. Če graf ni povezan, potem nas ponavadi zanima ena od komponent grafa ali particija vsake od komponent grafa, katere jemljemo kot ločene povezane grafe.

Poglavje 7

ENOSTAVNA MAKSIMIZACIJA MODULARNOSTI

Enostaven algoritem za maksimizacijo modularnosti je analogen algoritmu Kernighan-Lin. Algoritem za maksimizacijo modularnosti razdeli graf na dve skupnosti glede na neko začetno delitev (npr. naključna delitev na dve skupnosti enake velikosti). Algoritem nato pogleda za vsako točko, za koliko bi se modularnost spremenila, če bi točko premestili v drugo skupnost. Nato točko, ki bi modularnost najbolj povečala oz. najmanj zmanjšala, premesti v drugo skupnost. Nato to ponavlja s pogojem, da premesti točko največ enkrat v tem krogu algoritma. Modularnost Q lako definiramo na več načinov. Najbolj pogosto se uporablja definicija (glej poglavje 8)

$$Q = \frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m}) \delta(c_i, c_j),$$

kjer je c_i skupnost, ki ji točka i pripada.

Podobno, vendar drugače kot pri algoritmu Kernighan-Lin, ta algoritem zamenja le eno točko naenkrat. Pri algoritmu Kernighan-Lin je morala velikost skupnosti ostati konstantna, pri enostavni maksimizaciji modularnosti pa ne. Zato premikamo le po eno točko naenkrat, saj nas ne veže pogoj konstantnih velikosti skupnosti.

Ko se vse točke enkrat premeščene, pogledamo, v katerem krogu je bila modularnost največja in delitev v tem krogu uporabimo kot naslednjo začetno delitev. Ta del algoritma se ponavlja, dokler se modularnosti začetne in najboljše delitve na trenutni ponovitvi algoritma ne razlikujeta. Nato nam algoritem kot končno rešitev vrne to delitev (recimo začetno delitev v zadnji ponovitvi algoritma).

Algoritem je kar učinkovit. V vsakem krogu algoritem izračuna modularnost, če bi točka zamenjala skupnost. Za vsako od n točk za izračun modularnosti potrebuje v povprečju $O(m/n)$ časa. Ker je n krogov, ima vsaka ponovitev algoritma časovno zahtevnost $O(mn)$, kar je precej boljše od algoritma Kernighan-Lin s časovno zahtevnostjo $O(mn^2)$. Ker smo potrebovali le pogledati modularnost za vsako točko grafa (in ne vsak par točk v grafu), je zato celoten algoritem za faktor n hitrejši od algoritma Kernighan-Lin.

Poglavje 8

SPEKTRALNA MAKSIMIZACIJA MODULARNOSTI

8.1 Spektralna maksimizacija modularnosti

Podobno kot obstaja analogen algoritem algoritmu Kernighan-Lin za odkrivanje skupnosti, obstaja analogen algoritem spektralni particiji za problem odkrivanja skupnosti [15]. Tokrat delimo točke grafa na dve skupnosti poljubnih velikosti.

Najbolj pogosto modularnost definiramo kot razliko med povezavami znotraj skupnosti in pričakovanim številom povezav znotraj skupnosti za naključno generiran graf z enakimi stopnjami točk.

Poglejmo si, kolikšno bi bilo število povezav znotraj skupin za tak naključno generiran graf. Imamo graf z n točkami stopenj k_i in m povezavami. Vsako povezavo razrežemo na dve polovici in nato vsako polpovezavo naključno povežemo z drugimi polpovezavami (dovoljujemo večkratne povezave in zanke: povezave točke s samo seboj). Na ta način ohranimo stopnje točk in dobimo naključen graf.

Skupno število polpovezav l_n je enako

$$l_n = \sum_{i=1}^{n} k_i = 2m.$$

Če sedaj naključno izberemo dve točki i in j s stopnjami k_i in k_j ter ponovno povežemo polpovezavi v povezavo za ti dve točki, potem je pričakovano število povezav med i in j enako $(k_i k_j)/l_n = k_i k_j/2m$. Verjetnost, da se polpovezava točke i pri naključnem vezanju poveže s polpovezavo točke j je enaka $k_j/(2m-1)$. Za velike grafe je vrednost $2m-1$ približno enaka $2m$ in zato prejšnjo verjetnost zaokrožimo na $k_j/2m$. Ker sedaj gledamo vsako polpovezavo točke i, je matematično upanje ponovno povezanih povezav med i in j približno $k_i k_j/2m$. To sedaj naredimo za vsak par točk i, j v isti skupnosti. Modularnost bo potem dana kot vsota $A_{i,j} - \frac{k_i k_j}{2m}$ po vseh točkah i, j, ki so v isti skupnosti.

Modularnost Q bomo zapisali kot (faktor $\frac{1}{2m}$ je zgolj konvencionalen)

$$Q = \frac{1}{2m} \sum_{i,j} (A_{i,j} - \frac{k_i k_j}{2m}) \delta(c_i, c_j) = \frac{1}{2m} \sum_{i,j} F_{i,j} \delta(c_i, c_j), \tag{8.1}$$

kjer je c_i skupnost, ki ji točka i pripada, $\delta(m,n)$ je Krockerjev delta in

$$F_{i,j} = A_{i,j} - \frac{k_i k_j}{2m}.$$

Opazimo, da imajo vrednosti $F_{i,j}$ lastnost

$$\sum_j F_{i,j} = \sum_j A_{i,j} - \frac{k_i}{2m}\sum_j k_j = k_i - \frac{k_i}{2m}2m = 0, \tag{8.2}$$

in podobno za vsoto po i. Ponovno si poglejmo delitev na dve skupnosti ter predstavimo delitev z vrednostmi

$$s_i = \begin{cases} 1, & \text{če je točka } i \text{ v prvi skupnosti, in} \\ -1, & \text{če je točka } i \text{ v drugi skupnosti.} \end{cases}$$

Spomnimo se, da je $\frac{1}{2}(s_i s_j + 1)$ enako 1, če sta i in j v isti skupnosti, in 0 sicer, zato je

$$\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1).$$

Če zgornje vstavimo v enačbo (8.1), ugotovimo

$$Q = \frac{1}{4m}\sum_{i,j} F_{i,j}(s_i s_j + 1) = \frac{1}{4m}\sum_{i,j} F_{i,j} s_i s_j, \tag{8.3}$$

kjer smo uporabili enačbo (8.2). V matrični obliki lahko to zapišemo kot

$$Q = \frac{1}{4m} s^\top F s, \tag{8.4}$$

kjer je s vektor z elementi s_i in F $n \times n$ matrika z elementi $F_{i,j}$, ki ji rečemo matrika modularnosti. Enačba (8.4) je podobna enačbi za velikost prereza (6.4) s pomočjo Laplaceove matrike. S pomočjo te podobnosti izpeljemo analogno metodo spektralni particiji.

Želimo poiskati delitev grafa, ki maksimizira modularnost Q, torej želimo poiskati vektor s, ki maksimizira enačbo (8.4) za dano matriko modularnosti F. Elementi v vektorju s so omejeni na vrednosti ± 1, tako da vektor vedno kaže v eno od oglišč n-dimenzionalne kocke, drugih omejitev pa ni. Število elementov vektorja s enakim $+1$ in -1 ni omejeno kot v analogni spektralni particiji.

Podobno kot pri spektralni particiji je ta optimizacijski problem težek, zato tudi tukaj posežemo po sprostitveni metodi. Sprostimo lahko pogoj, da mora vektor s kazati v oglišče hiperkocke, tako da lahko kaže v katerokoli smer, s tem da ohranja dolžino $\sqrt{n}$, oziroma v matrični obliki

$$s^\top s = \sum_i s_i^2 = n. \tag{8.5}$$

Maksimizacija je sedaj precej enostavno rešljiva. Maksimiziramo enačbo (8.3) z uporabo vezanih ekstremov, kjer tokrat uporabimo le en Lagrangejev multiplikator β:

$$\frac{\partial}{\partial s_i}\left[\sum_{j,k} F_{j,k} s_j s_k + \beta(n - \sum_j s_j^2)\right] = 0.$$

Ko odvajamo, dobimo

$$\sum_j F_{i,j} s_j = \beta s_i,$$

oziroma v matrični obliki

$$Fs = \beta s. \tag{8.6}$$

Z drugimi besedami, vektor s je lastni vektor matrike modularnosti F. Če enačbo (8.6) vstavimo v enačbo (8.4), ugotovimo, da je modularnost podana kot

$$Q = \frac{1}{4m}\beta s^\top s = \frac{n}{4m}\beta,$$

kjer uporabimo tudi enačbo (8.5). Za doseg maksimuma modularnosti, moramo izbrati s kot lastni vektor u_1, ki pripada največji lastni vrednosti matrike modularnosti F.

Podobno kot pri spektralni particiji, ne moremo izbrati $s = u_1$, saj imajo elementi vektorja s pogoj $s_i = \pm 1$. Ampak še vedno lahko izberemo s tako, da bo čim bližje u_1, kar pomeni maksimizirati produkt

$$s^\top u_1 = \sum_i s_i [u_1]_i,$$

kjer $[u_1]_i$ je i-ti element vektorja u_1. Maksimum je dosežen, ko je vsak sumand nenegativen, tj. ko

$$s_i = \begin{cases} +1, & \text{če je } [u_1]_i > 0, \\ -1, & \text{če je } [u_1]_i < 0. \end{cases}$$

V primeru, ko je $[u_1]_i = 0$, sta obe vrednosti za s_i enako dobri in izberemo tisto, ki želimo.

To nas privede do naslednjega algoritma. Izračunamo lastni vektor matrike modularnosti, ki pripada največji lastni vrednosti. Nato dodamo točke v skupnost glede na to, ali je na pripadajočem mestu lastnega vektorja pozitivno ali negativno število. Točke s pozitivno vrednostjo damo v prvo skupino in točke z negativno vrednostjo damo v drugo skupino.

Potencialen problem je ta, da matrika F, v nasprotju z Laplaceovo matriko, ni redka. Večinoma so vsi njeni elementi neničelni. Na prvi pogled mislimo, da bo kompleksnost algoritma večja kot pri normalni spektralni particiji. Iskanje vodilnega lastnega vektorja matrike traja $O(mn)$ časa, kar je $O(n^3)$ v gostih matrikah. Z uporabo določenih lastnosti matrike modularnosti je možno vodilni lastni vektor najti v času $O(n^2)$ v redkih grafih.

8.2 Prirejena metoda za komplement grafa

Če ima graf z dvemi skupnostmi približno enake velikosti

- malo povezav med skupnostmi in
- veliko povezav znotraj skupnosti,

potem ima njegov komplement

- veliko povezav med skupnostmi in
- malo povezav znotraj skupnosti.

Z drugimi besedami, komplement je skoraj dvodelen in lahko iskanje skupnosti prevedemo na iskanje večjih dvodelnih podgrafov v komplementu [18]. Dvodelnost grafa se lahko prepozna na dva načina:

- Graf je dvodelen, če sta najmanjša in največja lastna vrednost matrike sosednosti enaki v absolutni vrednosti. Izračunamo lastni vektor matrike sosednosti z najmanjšo lastno vrednostjo. Točko pripišemo v prvo skupino, če je pripadajoči element lastnega vektorja pozitiven, in v drugo skupino, če je negativen.
- Graf je dvodelen, če je največja lastna vrednost normalizirane Laplaceove matrike enaka 2. Izračunamo lastni vektor normalizirane Laplaceove matrike z največjo lastno vrednostjo. Točko pripišemo v prvo skupino, če je pripadajoči element lastnega vektorja pozitiven, in v drugo skupino, če je negativen.

Obe metodi uporabimo za delitev grafa na dve skupnosti. Metodo lahko zaporedemo uporabimo za delitev grafa na več komponent, vendar ni nobenega pogoja kdaj naj z delitvijo prenehamo.

Naj bo $A \in \mathbb{R}^{n\times n}$ matrika sosednosti grafa Γ. Potem je matrika sosednosti komplementa grafa Γ enaka $J_n - I_n - A$. Torej enotski lastni vektor x minimizira $x^\top(J_n - I - A)x$, oz. z drugimi besedami, maksimizira $x^\top(A-J)x = \sum_{i,j} A_{i,j}x_ix_j - \left(\sum_i x_i\right)^2$, kjer je x_i i-ti element vektorja x. Če vektor x z elementi ± 1 opisuje delitev na skupnosti c_1 in c_2, potem zgornja enačba postane oblike

$$2|\{(u,v)\in E : u,v\in c_1\}|+2|\{(u,v)\in E : u,v\in c_2\}|-2|\{(u,v)\in E : u\in c_1, v\in c_2\}|-(|c_1|-|c_2|)^2.$$

Torej bo težnja po maksimizaciji te enačbe težila k temu, da je čim več povezav v skupnostih, čim manj povezav med skupnostmi in čim manj razlike v velikosti skupnosti.

Podobno, enotski lastni vektor pripadajoč največji lastni vrednosti normalizirane Laplaceove matrike komplementa maksimizira

$$2\sum_{(u,v)\notin E}\frac{x_u}{\sqrt{n-1-k_u}}\frac{x_v}{\sqrt{n-1-k_v}}.$$

Če vektor x z elementi ± 1 opisuje delitev na skupnosti c_1 in c_2, potem, ko utežimo vsako točko z njeno "pomembnostjo", $w_u = \frac{1}{\sqrt{n-1-k_u}}$ zgornja enačba postane

$$2\sum_{(u,v)\in E,\, u,v\in c_1} w_uw_v + 2\sum_{(u,v)\in E,\, u,v\in c_2} w_uw_v - 2\sum_{(u,v)\in E,\, u\in c_1,\, v\in c_2} w_uw_v - \left(\sum_{u\in c_1} w_u - \sum_{v\in c_2} w_v\right)^2.$$

Poglavje 9

DELITEV NA VEČ KOT DVE SKUPINI

Algoritma, ki smo ju spoznali do sedaj za odkrivanje skupnosti, delujeta samo za delitev grafa na natanko dve skupnosti nedoločene velikosti. Skupnosti so definirane kot naravne skupine točk v grafu in ni nobenega razloga, zakaj bi obstajale samo dve. V splošnem nočemo določiti na koliko skupnosti želimo graf razdeliti, njihovo število bi moralo biti dano glede na graf in ne s strani opazovalca.

Metoda maksimiziranja modularnosti lahko reši ta problem zelo dobro. Toda namesto maksimiziranja modularnosti na dve skupnosti, maksimiziramo glede na delitev na poljubno mnogo skupnosti. Modularnost bo največja v najboljši delitvi ne glede na to, na koliko skupnosti je razdeljen graf.

Poglejmo si enostavno metodo, ki temeji na večkratni bisekciji grafa (bisekcija v okviru odkrivanja skupnosti) z uporabo prejšnih algoritmov. Pozorni moramo biti, kako bisekcijo naredimo. Ne moremo ravnati s skupnostmi po prvi bisekciji kot z manjšimi podgrafi in nato bisekcijo uporabiti na njih ločeno. Modularnost celotnega grafa se ne razcepi (kot velikost prereza) na neodvisne prispevke vsake skupnosti. Maksimizacija modularnosti za skupnost kot ločene grafe v splošnem ne bo pripeljala do maksimalne modularnosti za celoten graf.

Zato moramo obravnavati spremembo ΔQ v modularnosti celotnega grafa, ko ponovno naredimo bisekcijo skupnosti c velikosti n_c. To spremembo lahko zapišemo kot

$$\begin{aligned}
\Delta Q &= \frac{1}{2m}\left[\frac{1}{2}\sum_{i,j\in c} B_{i,j}(s_i s_j + 1) - \sum_{i,j\in c} B_{i,j}\right] && (9.1)\\
&= \frac{1}{4m}\left[\sum_{i,j\in c} B_{i,j} s_i s_j - \sum_{i,j\in c} B_{i,j}\right] && (9.2)\\
&= \frac{1}{4m}\sum_{i,j\in c}\left[B_{i,j} - \delta(i,j)\sum_{k\in c} B_{i,k}\right] s_i s_j && (9.3)\\
&= \frac{1}{4m} s^\top B^{(c)} s, && (9.4)
\end{aligned}$$

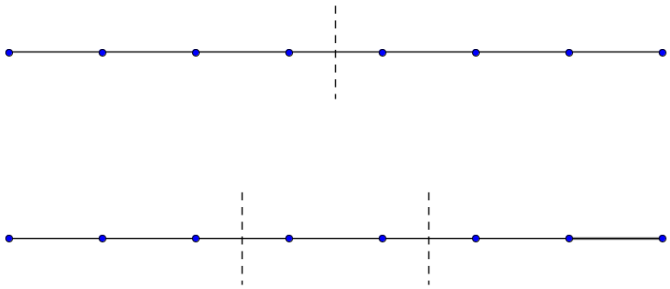
kjer smo uporabili lastnost $s_i^2 = 1$ in $B^{(c)}$ je $n_c \times n_c$ matrika z elementi

$$B_{i,j}^{(c)} = B_{i,j} - \delta(i,j) \sum_{k \in c} B_{i,k}.$$

Na enačbi (9.1) lahko sedaj uporabimo spektralni pristop. Podobno kot prej, maksimiziramo ΔQ, torej iščemo vodilni lastni vektor in razdelimo graf glede na predznak njegovih elementov.

Ko večkrat razdelimo graf na ta način, se vprašamo, kdaj moramo prenehati s subdivizijo. Odgovor na to je preprost. Glede na to, da želimo maksimizirati modularnost, subdivizijo ponavljamo, dokler se modularnost veča. Če ne moremo več najti delitve, ki bi povečala ΔQ, potem skupnosti ne razdelimo. Bisekcija grafa bi v tem primeru dala v prvo skupino vse točke skupnosti in v drugo nobene. Ko nobene skupnosti ne moremo več razdeliti, končamo z algoritmom.

Večkratna bisekcija deluje v praksi precej dobro v mnogih primerih, ampak ni popolna. Nobene garancije ni, da se najboljša delitev iz treh delov lahko najde preko prvotne delitve na dva dela. Poglejmo si to na naslednjem primeru:



Slika 9.1: Delitev grafa.

Na zgornjem delu slike je bisekcija grafa P_8. Na spodnjem delu je optimalna delitev istega grafa na tri dele. Vendar iz zgornje delitve po nadaljni delitvi ne bi nikoli mogli priti do spodnje delitve.

Alternativna metoda bi bila, da najdemo neposredno delitev omrežja namesto ponavljanja bisekcije. Take metode bi morale najti boljšo delitev, ampak so bolj kompleksne in imajo večjo časovno zahtevnost.

Poglavje 10

ALGORITMI, KI ODSTRANJUJEJO POVEZAVE

Eden od pogledov na iskanje skupnosti v grafu je, da pogledamo povezave med skupnostmi. Če lahko najdemo in odstranimo te povezave, nam bodo ostale samo ločene skupnosti.

Obstaja več možnosti, kako definiramo, katere povezave so med skupnostmi. To lahko definiramo preko *povezavne medtočkovne mere* (angl. edge betweenness centrality). Medtočkovna mera povezave v grafu je število najkrajših poti, ki gredo skozi to povezavo. Za povezave med skupnostmi se pričakuje visoko povezavno medtočkovno mero.

Računanje povezavne medtočkovne mere je precej enostavno. Za vsak par (povezanih) točk izačunamo najkrajšo pot med njima (npr. iskanje v širino) in nato vsaki povezavi na tej poti povečamo mero za 1. V kolikor je j najkrajših poti in se povezava na teh j poteh pojavi i-krat, mero povečamo za $\frac{i}{j}$. Mero vseh povezav se lahko izračuna v času $O(n(m+n))$.

Algoritem za odkrivanje skupnosti je sledeč: izračunamo mero vseh povezav in nato odstranimo tisto z najvišjo mero. Ko odstranimo povezavo, se mera spremeni na preostalem grafu in zato ponovno izračunamo mero vsake povezave. Nato ponovno odstranimo povezavo z najvišjo mero in ponavljamo. Graf bo čez čas razpadel na več komponent.

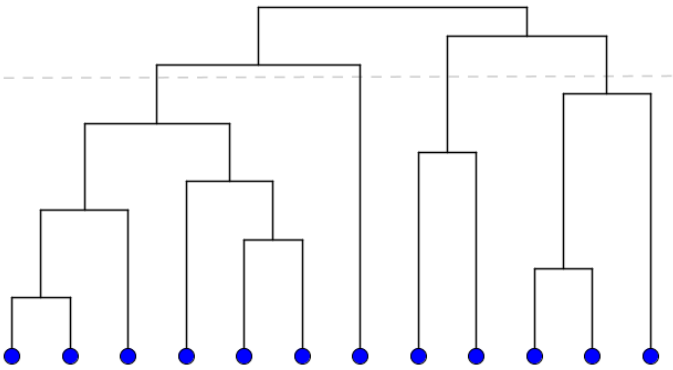
Potek algoritma lahko beležimo v dendrogramu.

Algoritem je drugačen od prejšnjih, saj ne dobimo ene rešitve, temveč izbor več možnih delitev. Uporabnik se odloči, katere delitve so uporabne glede na njegov namen in katere niso. Lahko bi uporabili modularnost, da bi izmerili, katera delitev je najbolj uporabna. To bi zgrešilo namen algoritma. Če bi želeli maksimalno modularnost, potem bi uporabili algoritem maksimiziranja modularnosti.

Na žalost je algoritem razmeroma počasen. Kot smo že povedali, izračun povezavne točkovne mere vsake povezave traja $O(n(m+n))$ časa. Mero vsake povezave moramo izračunati vsakič, ko odstranimo eno izmed m povezav. Skupna časovna zahtevnost je zato enaka $O(mn(m+n))$. Algoritem vrača precej dobre rezultate, vendar je manj v uporabi kot drugi hitrejši algoritmi.

Dendrogram je lahko zelo uporabna rešitev v primerjavi z eno samo delitvijo grafa. Delitve predstavljene v dendrogramu tvorijo *hierarhično dekompozicijo*, v kateri so skupnosti na nekem nivoji vsebovane v skupnostih na višjih nivojih.

Variacijo tega algoritma je razvil Radicchi. Skupna točka obeh algoritmov je v tem,

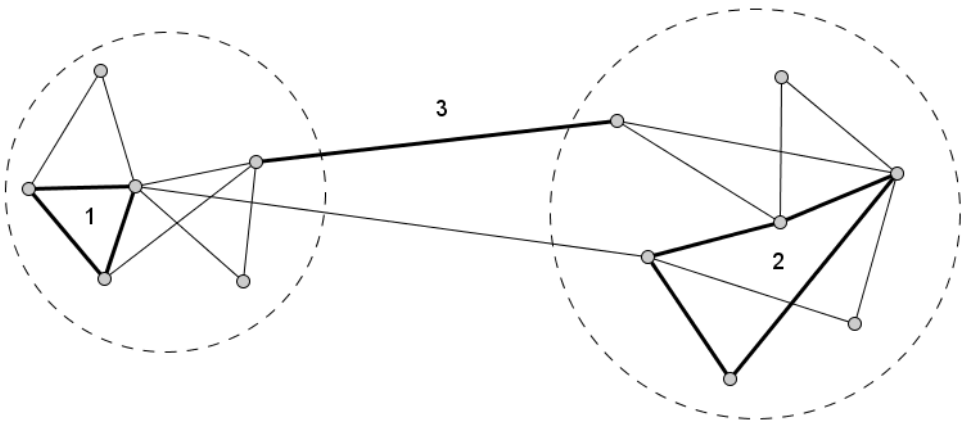


Slika 10.1: Dendrogram.

Rezultat algortima povezavne medtočkovne mere lahko prikažemo kot drevo oz. dendrogram, v katerem so točke postavljene na dno drevesa. Algoritem je najprej izločil povezave na vrhu in po vrsti navzdol nadaljeval z izločanjem. Vsak vodoraven prerez nam vrne delitev grafa.

da odstranjujemo povezave med skupnostmi, vendar je tokrat mera drugačna. Povezave med skupnostmi ponavadi ne ležijo na kratkih ciklih. Če se omejimo na cikle dolžine 3 ali 4, se algoritem po navadi najbolj obnese. Z odstranitvijo povezav, ki ležijo na majhnem številu takih ciklov, odkrijemo skupnosti v grafu.

Dobra lastnost te spremenjene metode je hitrost. Algoritem ima časovno zahtevnost enako $O(n^2)$ v redkih grafih. Slaba lastnost algoritma pa je, da deluje le na grafih, ki imajo dovolj veliko kratkih ciklov. Ta metoda deluje zelo dobro na socialnih omrežjih, ki imajo zelo veliko število kratkih ciklov. Za druga omrežja, kot so tehnološka in biološka, ne deluje dobro, saj imajo ta omrežja ponavadi majhno število kratkih ciklov in algoritem zato težko razloči med povezavami med skupnostmi in v skupnostih.



Slika 10.2: Radicchijev algoritem.

Algoritem išče povezave, ki pripadajo najmanj kratim ciklom. V mnogih grafih povezave v skupnostih tvorijo kratke cikle, kot so cikli označeni z 1 in 2. Povezave med skupnostmi, kot je povezava označena s 3, pa po navadi ne pripadajo takim ciklom.

Poglavje 11

HIERARHIČNO GROZDENJE

Algoritma v prejšnjem poglavju sta drugačna od ostalih, saj nam vrneta hierarhično dekompozijo grafa v obliki dendrograma. Tudi sledeči algoritem je tak in je eden najstarejših algoritmov odkrivanja skupnosti, algoritem hierarhičnega grozdenja.

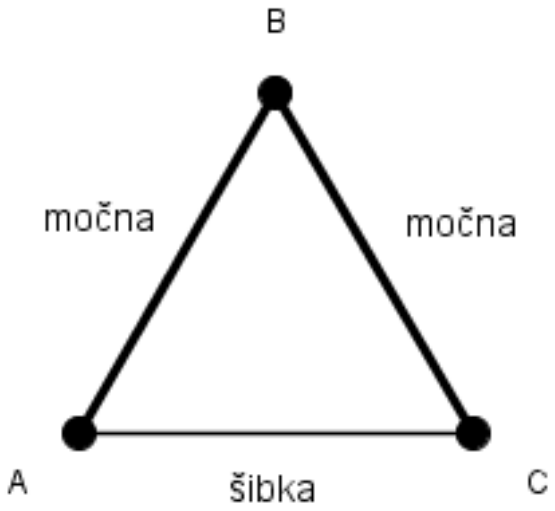
Hierarhično grozdenje ni samo en algoritem, temveč cela družina algoritmov z veliko variacijami in alternativami. Algoritmi slonijo na tem, da začnejo s točkami in jih združujejo v skupnosti. To je v nasprotju z mnogimi prejšnjimi metodami, ki so začele z celotnim greafom ter ga začeli deliti.

Osnovna ideja hierarhičnega grozdenja je v definiranju mere, ki meri moč povezanosti (podobnosti) med točkami, glede na strukturo grafa in povezovanju pomembnih točk med seboj. Obstaja več takih mer, ki merijo moč povzanosti med točkami. Velika izbira za mero ima svoje dobre in slabe lastnosti. Imamo večjo fleksibilnost in lahko izbiramo mero za določen graf. Več mer pomeni več različnih rezultatov in velikokrat se ne ve, katera mera je boljša za določen graf oz. katera bo dala boljši rezultat. Po navadi je izbira mere odvisna od izkušenj ali eksperimenta.

Ko se odločimo za mero podobnosti, jo izračunamo za vsak par točk v grafu. Želimo povezati točke z visoko mero podobnosti skupaj v skupnosti. To pripelje do naslednjega problema. Podobnost lahko vrne nasprotujoče si podatke o tem, katere točke bi združili. Recimo, da imata točki A in B visoko podobnost, ter prav tako točki B in C. Lahko bi rekli, da bi morale biti točke A, B, C vse v isti skupnosti. Kaj pa naredimo v primeru, ko imata točki A in C majhno podobnost? To nas privede do dileme, ali naj bosta točki A in C v isti skupnosti ali ne.

Osnovna strategija hierarhičnega grozdenja je, da najprej združimo pare točk z največjo podobnostjo. Tako naredimo skupnost ali skupnosti velikosti 2. Nato združimo najbolj podobne skupnosti v večje skupnosti itd. Za podobnost med točkami imamo mero, vendar potrebujemo tudi mero za podobnost skupnosti. To naredimo tako, da združimo podobnosti posameznih parov točk v podobnost skupnosti.

Obstajajo tri načini združevanja podobnosti točk. Pravimo jim eno-, celotno-, in povprečno-povezavno grozdenje. Naj imata skupnost 1 in skupnost 2 n_1 in n_2 točk. Potem obstaja $n_1 n_2$ parov točk tako, da je ena točka v skupnosti 1 in druga v skupnosti 2. V eno-povezavnem grozdenju je podobnost med dvemi skupnostmi enaka podobnosti najbolj podobnemu paru med temi $n_1 n_2$ pari točk. V drug ekstrem gre celotno-povezavno grozdenje, ki definira podobnost med skupnostmi kot podobnost para izmed $n_1 n_2$ parov



Slika 11.1: Dilema.
Če sta povezavi (A, B) in (B, C) močni in povezava (A, C) šibka, ali naj bosta A in C v isti skupnosti ali ne?

točk z najmanjšo podobnostjo. Med obema ekstremoma se nahaja povprečno-povezavno grozdenje. Podobnost dveh skupnosti je tokrat enaka aritmetični sredini podobnosti vseh $n_1 n_2$ parov točk. Slednja definicija je verjetno najboljša, vendar je iz neznanih razlogov redko uporabljena.

Celoten algoritem hierarhičnega grozdenja zgleda tako:

1. Izberi mero podobnosti in jo izračunaj za vsak par točk.
2. Določi skupnosti tako, da je vsaka točka v svoji skupnosti. Začetne podobnosti skupnosti so enake podobnostim med točkami.
3. Najdi pare skupnosti, ki imajo največjo podobnost in jih združi v eno skupnost.
4. Izračunaj podobnost med novo nastalimi skupnostmi in preostalimi z uporabo enega izmed treh tipov povezavnega grozdenja.
5. Ponavljaj od 3. koraka dalje, dokler se vse točke ne združijo v eno skupnost.

Izračun novih podobnosti je precej enostaven. Poglejmo si izračun vsakega od treh tipov grozdenja posebej. V primeru eno-povezavnega grozdenja je podobnost dveh skupnosti enaka podobnosti najbolj podobnemu paru točk. Če združimo skupnosti 1 in 2 je podobnost nove skupnosti z skupnostjo 3 enaka večji od podobnosti med 1 z 3 in 2 z 3. Za celotno-povezavno grozdenje se gleda manjšo izmed podobnosti med 1 z 3 in 2 z 3 kot podobnost med skupnostma 1 združena z 2 in 3.

Povprečno-povezavno grozdenje je malo bolj zahtevno. Predpostavimo, da združujemo skupnosti 1 in 2 z n_1 in n_2 točkami. Naj bo $\sigma_{1,3}$ podobnost med skupnostma 1 in 3 ter

$\sigma_{2,3}$ podobnost med skupnostma 2 in 3. Potem je podobnost $\sigma_{12,3}$ med novo združeno skupnostjo in skupnostjo 3 enaka

$$\sigma_{12,3} = \frac{n_1\sigma_{1,3} + n_2\sigma_{2,3}}{n_1 + n_2}.$$

Podobno kot prej se da izračun novih podobnosti izračunati v $O(1)$ časa. Na vsakem koraku algoritma moramo na novo izračunati podobnost združene skupnosti z ostalimi skupnostmi, katerih je $O(n)$. Torej bo računanje novih podobnosti na vsakem koraku vzelo $O(n)$ časa. Naivno iskanje para z največjo podobnostjo traja $O(n^2)$. To lahko pospešimo tako, da shranjujemo podobnosti v *dvojiško kopico*, ki nam dovoli dodajati in odstraniti vnose v času $O(\log_2 n)$ in najde največji vnos v času $O(1)$. To upočasni ponoven izračun podobnosti na $O(n \log_2 n)$ in hkrati pospeši iskanje največjega vnosa na $O(1)$.

Vsak proces združevanja skupnosti moramo ponoviti $n - 1$ krat, dokler niso vse točke združene v eno samo skupnost. Torej je skupna časovna zahtevnost algoritma enaka $O(n^3)$ v naivni implementaciji in $O(n^2 \log_2 n)$, če uporabimo dvojiško kopico. Algoritem ne deluje zmeraj dobro. Dober je za izbiro jeder v grafu, kjer so si vse točke med seboj zelo podobne. Je slabši v pripisovanju točk skupnostim, ki so nekje vmes med jedri. Take točke niso zelo podobne ostalim in jih algoritem ponavadi ne upošteva do zadnjih nekaj korakov algoritma. Pogost rezultat hierarhičnega grozdenja je množica zelo prepletenih jeder z nekaj posameznimi točkami in majhnimi skupnostmi. Tak rezultat lahko vseeno vsebuje veliko koristnih informacij o grafu.

Poglavje 12

ALGORITEM GOEMANSA IN WILLIAMSONA ZA PROBLEM MAKSIMALNEGA PREREZA

Algoritem, ki ga bom predstavil za problem maksimalnega prereza, temelji na semidefinitnem programiranju. Najprej ponovimo osnove semidefinitnega programiranja.

12.1 Semidefinitno programiranje

Definicija 12.1. *Kvadratna simetrična matrika $A \in \mathbb{R}^{n\times n}$ je pozitivno semidefinitna, če za vsak $x \in \mathbb{R}^n$ velja $x^\top Ax \geq 0$.*

Za dano simetrično matriko A so naslednje trditve ekvivalentne:

- A je pozitivno semidefinitna.
- Vse lastne vrednosti matrike A so nenegativne.
- Obstaja taka realna kvadratna matrika W, da velja $A = W^\top W$.

Definicija 12.2. *Semidefinitni program je optimizacijski problem, pri katerem iščemo optimum linearne funkcije elementov simetrične matrike pri linearnih pogojih in dodatni zahtevi, da je matrika pozitivno semidefinitna.*

Standardna oblika semidefinitnega programa:

$$\min\ C \bullet X$$

$$\text{pri pogojih } A_i \bullet X = b_i \text{ za vse } i = 1, ..., m$$

$$X \succ 0,$$

kjer so $C, A_1, ..., A_m, X$ $n \times n$ matrike in dodatno je X simetrična matrika. Operacija $\bullet$ je skalarni produkt matrik:

$$A \bullet B = \sum_{i,j} A_{i,j}B_{i,j} = \mathrm{tr}(A^\top B).$$

$A \succ B$ natanko tedaj, ko je matrika $A - B$ pozitivno semidefinitna.

Semidefinitno programiranje (SDP) je razred vseh semidefinitnih programov. Osnovna dejstva o semidefinitnem programiranju:

- Semidefinitno programiranje je posplošitev linearnega programiranja. Če semidefinitnem programu v standardni obliki dodamo pogoj, da je X diagonalna matrika, dobimo linearni program v standardni obliki.
- Semidefinitno programiranje je poseben primer t.i. konveksnega in tudi koničnega programiranja. Množica vseh pozitivno semidefinitnih matrik tvori konveksni stožec.
- Obstaja teorija dualnosti za semidefinitno programiranje (Wolkowicz 1981 [20], Alizadeh 1995 [1]).
- Simpleksno metodo je moč posplošiti na semidefinitno programiranje (Pataki 1995 [17]).
- Za poljuben $\varepsilon > 0$ je moč probleme semidefinitnega programiranja rešiti znotraj aditivne napake ε v polinomskem času tj., v času

$$p(\text{velikost vhodnih matrik in vektorjev}, \log_2 \frac{1}{\varepsilon}),$$

kjer je p polinom in ε del vhodnih podatkov [9].

12.2 Algoritem Goemansa in Williamsona

Naj bo $\Gamma = (V, E)$ graf. Iščemo tak prerez (S, S'), da bo vrednost $|\{(u, v) \in E : u \in S, v \in S'\}|$ maksimalna.

Problem najprej zapišemo s kvadratnim celoštevilskim programom. Spremenljivkam $(x_v : v \in V)$ priredimo vrednosti

$$x_v = \begin{cases} +1, & \text{če je } v \in S, \\ -1, & \text{sicer.} \end{cases}$$

Maksimizirati moramo vsoto

$$\max \sum_{(u,v)\in E} \frac{1 - x_u x_v}{2},$$

pri pogojih

$$x_v \in \{+1, -1\}, \quad \forall v \in V.$$

Naj bo $x \in \{+1, -1\}^V$ dopustna rešitev. Prerez (S, S') dobimo tako, da točke, ki imajo pripadajoči element v vektorju x enak 1, damo v množico S ($S = \{v \in V : x_v = 1\}$).

Podobno kot v prejšnjih poglavjih moramo pogoj relaksirati zato, da bo rešljiv v polinomskem času. Relaksacijo zgornjega problema do t.i. *vektorskega programa* dobimo tako, da predpostavimo, da je x_v enotski vektor v Evklidskem prostoru dimenzije $n = |V|$

(namesto v eni dimenziji). Zgornji problem maksimizacije se nam prevede na sledečo maksimizacijo:

$$\max \sum_{(u,v)\in E} \frac{1 - x_u \cdot x_v}{2},$$

pri pogojih

$$||x_v||_2 = 1,\ x_v \in \mathbb{R}^n \forall v \in V.$$

Pri tem je $a \cdot b$ skalarni produkt vektorjev a in b: $a \cdot b = \sum_{i=1}^{n} a_i b_i = \langle a, b \rangle$.

Če postavimo $y_{(u,v)} = x_u \cdot x_v$, lahko zgornji program zapišemo kot semidefinitni program:

$$\max \sum_{(u,v)\in E} \frac{1 - y_{(u,v)}}{2},$$

pri pogojih

$$y_{(v,v)} = 1, \forall v \in V.$$

$$Y = (y_{(u,v)}) \text{ je simetrična in pozitivno semidefinitna matrika.}$$

Naj bo *SDP* vrednost zgornjega semidefinitnega programa. Očitno velja:

$$SDP \geq OPT\,.$$

Vsako dopustno rešitev kvadratnega programa lahko namreč dopolnimo do dopustne rešitve vektorskega programa, tako da $n-1$ koordinat postavimo na 0.

Zgled: Če je $\Gamma = C_3$, potem je $OPT = 2$, po drugi strani pa je $SDP = 9/4$: trije vektorji v ravnini, ki so paroma narazen za 120 stopinj.

Za dani algoritem torej zadošča pokazati, da velja $ALG \geq \rho \cdot SDP$. Posledično bo potem tudi $ALG \geq \rho \cdot OPT$.

Algoritem: Naj bo $\{x_v\}$ optimalna rešitev zgornjega SDP v $\mathbb{R}^n$. Naključno izberimo enotski vektor r iz enotske sfere

$$S^{n-1} = \{x \in \mathbb{R}^n : ||x||_2 = 1\}$$

in postavimo

$$S = \{v \in V : r \cdot x_v \geq 0\}.$$

Geometrijski pomen: Naključno izberemo hiperravnino skozi izhodišče, katere normala je vektor r. Ta hiperravnina razdeli množico vektorjev (točk) na dva dela, ki tvorita prerez množice V.

Zgornji postopek imenujemo verjetnostno zaokroževanje (angl. randomized rounding).

- Zaokroževanje je invariantno za rotacije (prav tako kot vektorski program).
- Kako izberemo naključen vektor r v S^{n-1}? Naj bodo $X_1, ..., X_n$ neodvisne standardno normalno porazdeljene slučajne spremenljivke, $X_i \sim N(0,1)$ za vse i. Za r izberemo enotski vektor v smeri $(X_1, ..., X_n)$.

Izrek 12.3. *Matematično upanje vrednosti prereza, ki ga pridela zgornji algoritem, je vsaj* $0.87856 \cdot SDP$.

Lema 12.4. *Za vse povezave* $(u, v) \in E$ *velja*

$$P[\text{S vsebuje natanko eno krajišče povezave } (u,v)] = \frac{\alpha_{u,v}}{\pi},$$

kjer je $\alpha_{u,v} \in [0, \pi]$ *kot med* x_u *in* x_v.

Dokaz. Zaradi rotacijske invariantnosti vektorja r in rešitve SDP lahko brez škode za splošnost privzamemo, da sta x_u in x_v vsebovana v (dvodimenzionalni) ravnini.

Privzamemo lahko torej $X_3 = \ldots = X_n = 0$. Vektor r je torej naključno izbran vektor v enotski krožnici v ravnini. Če si pomagamo z dvo-dimenzionalno sliko, se hitro prepričamo, da je verjetnost, da hiperravnina z normalo r loči x_u od x_v, enaka $\frac{\alpha_{u,v}}{\pi}$. □

Dokaz izreka (12.3):

Iz trditve sledi, da za vsako povezavo $(u, v) \in E$ velja

$$P[\text{S vsebuje natanko eno krajišče povezave } (u,v)] = \frac{\alpha_{u,v}}{\pi}.$$

Z nekaj analize lahko pokažemo, da velja

$$\min_{0 \le \theta \le \pi} \frac{2\theta}{\pi(1 - \cos\theta)} > 0,87856$$

in posledično

$$\frac{\alpha_{u,v}}{\pi} \ge 0,87856 \cdot \left(\frac{1 - \cos\alpha_{u,v}}{2}\right) = 0,87856 \cdot SDP.$$

□

Če postopek ponovimo velikokrat (n^2-krat), lahko z veliko verjetnostjo pričakujemo, da bo najboljši izmed dobljenih prerezov vseboval vsaj $87,8\%$ povezav optimalnega prereza. V praksi je algoritem še precej boljši, kot to nakazuje faktor aproksimacije $0,87856$.

Poglavje 13

ZAKLJUČEK

V magistrskem delu smo spoznali probleme particije grafa, odkrivanja skupnosti in maksimalnega prereza. Za nobenega od naštetih se ne pozna natančnega algoritma, ki bi rešil problem v polinomskem času. Zato smo si pogledali različne hevristike in aproksimacijske algoritme za te probleme z mnogimi različnimi prijemi: od predstavljivega premeščanja točk do nepredstavljivega algoritma, ki temelji na lastnih vrednostih matrik. Seveda to še zdaleč niso vsi algoritmi za navedene probleme. Drži pa, da je veliko algoritmov zelo podobnih algoritmom, predstavljenih v tem magistrskem delu. Nekateri se začnejo enako in se razlikujejo le v nekaj malenkostih na koncu.

Problemi niso le teoretičnega pomena, temveč imajo uporabne aplikacije v resničnem svetu. Algoritmi za particijo grafa se uporabljajo pri delitvi naloge na več procesorjev ali na več vzporednih računalnikov. Algoritmi za odkrivanje skupnosti se uporabljajo za odkrivanje strukture raznih omrežij (internetno, prijateljsko, socialno, biološko itd.). Algoritmi za maksimalni prerez imajo aplikacije v statistični fiziki in vezjih.

Literatura

[1] F. Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization, *SIAM Journal on Optimization 5* (1995), 13-51.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, 2. izd., MIT Press, McGraw-Hill, 2001.

[3] M. R. Garey, D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, W.H. Freeman and Co., San Francisco, 1979.

[4] GeoGebra, http://www.geogebra.org/, dostopno na spletu 1.9.2012.

[5] GNU Octave, http://www.gnu.org/software/octave/, dostopno na spletu 1.9.2012.

[6] M. Goemans, D. Williamson, .878-approximation algorithms for MAX CUT and MAX 2SAT, 26th STOC (1994), 422-431.

[7] M. C. Golumbic, Algorithmic graph theory and perfect graphs, 2. izd., Elsevier, North Holland, 2004.

[8] G. Grimmett, D. Welsh, Probability: an introduction, Oxford Science Publications, Oxford, 1986.

[9] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1 (1981), 169-197.

[10] M. Kolar, B. Zgrablić, Več kot nobena, a manj kot tisoč in ena rešena naloga iz linearne algebre, Pedagoška fakulteta, Ljubljana, 1996.

[11] B. Korte, J. Vygen, Combinatorial optimization: theory and algorithms, 2. izd., Springer-Verlag, 2002.

[12] J. Kozak, Podatkovne strukture in algoritmi, Društvo matematikov, fizikov in astronomov Slovenije, Ljubljana, 1997.

[13] M. Milanič, Izbrana poglavja iz diskretna matematike 1, UP FAMNIT, zapiski predavanj, 2010-2011.

[14] M. Milanič, Verjetnost, UP FAMNIT, zapiski predavanj, 2008-2009.

[15] M. E. J. Newman, Modularity and community structure in networks, *Proc. Natl. Acad. Sci. USA* 103 (2006), 8577-8582.

[16] M. E. J. Newman, Networks: an introduction, Oxford University Press, Oxford, 2009.

[17] G. Pataki, On cone-LP's and semi-definite programs: Facial structure, basic solutions, and the simplex method, GSIA Working paper WP 1995-03, Carnegie-Mellon Univ., Pittsburgh, Pa., 1995.

[18] D. Stevanović, Spectral approaches to community detection, CRM Conference on Applications of Graph Spectra in Computer Science, Barcelona, Spain, julij 16-20, 2012.

[19] V. V. Vazirani. Approximation Algorithms, Springer-Verlag, 2001.

[20] H. Wolkowicz, Some applications of optimization in matrix theory. *Linear Algebra and its Applications* 40 (1981), 101-118.

[21] J. Žerovnik, Osnove teorije grafov in diskretne optimizacije, Fakulteta za strojništvo, Maribor, 2005.