UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO
ODDELEK ZA MATEMATIKO
Matematika - 3. stopnja

Selena Praprotnik

RAZVOJ SKUPIN V OMREŽJIH

Doktorska disertacija

Mentor: prof. dr. Vladimir Batagelj

Ljubljana, 2015

Podpisana Selena Praprotnik izjavljam:

- da sem doktorsko disertacijo z naslovom *Razvoj skupin v omrežjih* izdelala samostojno pod mentorstvom prof. dr. Vladimirja Batagelja in
- da Fakulteti za matematiko in fiziko Univerze v Ljubljani dovoljujem objavo elektronske oblike svojega dela na spletnih straneh.

Ljubljana, 25. 3. 2015 Podpis:

Zahvala

Bolj ko razmišljam o vseh ljudeh, ki so zaslužni, da je pričujoče besedilo prišlo v vaše roke, bolj se mi zdi, da ne obstaja dovolj besed, ki bi opisale hvaležnost, ki si jo zaslužijo. Vseeno se bom po najboljših močeh potrudila in poskusila opisati, kako zelo so (bili) pomembni v obdobju, ko je nastajal tale doktorat.

Zahvaljujem se mentorju, prof. dr. Vladimirju Batagelju, za pomoč pri pisanju disertacije in za resnično veliko razumevanje moje osebne situacije. Prepričana sem, da tudi vam ni bilo lahko. Zahvaljujem se vam iz vsega srca. Za članke, za izpite, za čas, ... za vse, kar ste naredili zame. Hvala vam!

Zahvaljujem se tudi svojim kolegom: Nejcu za spodbudo pri pisanju in Alenu za pozitivne misli in za to, da nikoli ni nehal verjeti, da mi bo uspelo. Predvsem pa se zahvaljujem Iztoku. Ne vem, kolikokrat si me zbudil v pisarni, ko sem zaspala ob računalniku, in me zvlekel na kavo. Hvala za vse klepete in za vso podporo. Za vse sobote ali nedelje, ko Izaka ni bilo doma. Zares sem ti hvaležna za vso pomoč.

Hvala tudi vsem tistim, ki ste si upali stopiti v moj osebni pekel zadnjih let in mi lajšali in olepševali življenje. Tistim, ki ste se potrudili, da bi nama bilo vsaj malo bolje. Hvala, ker ste mi pokazali resnico pregovora: "Čas zaceli, česar razum ne more."

Moji ljubi mami, ki je skrbela zame in za sina, ko nisem več zmogla bremena, ki mi je bilo naloženo; Matiji, ki me je pobral takrat, ko resnično nisem več verjela, da bo kdaj bolje; Katji, ki naju je sprejela, ko nisva imela iti kam drugam; Alešu, ki je ostal tudi takrat, ko so vsi odšli. Zahvaljujem se vam za prijateljstvo, ki ste mi ga izkazali, in za to, da ste bili tam (še posebej takrat), ko je bilo najhuje. Hvala, da ste mi kazali lepoto, ko je bil ves moj svet poln teme in bolečin. Zaradi vas sem ostala (kolikor toliko) prisebna in prepričana sem, da doktorata ne bi bilo brez vaše podpore v zasebnem življenju.

Zahvaljujem se Izaku, ki je najboljši sin, kar bi si jih lahko želela. Kljub mladosti si bil sposoben razumeti, da žal nimam časa, in me motiviral s svojim stalnim ponavljanjem stavka "Mami, piši! – Jama!!!" Neverjeten si in ponosna sem nate.

Za konec pa:

> *Life has a way of testing a person's will, either by having nothing happen at all or by having everything happen at once.* (Paulo Coelho)

. . . in za dežjem VEDNO posije sonce.

Povzetek

V disertaciji vpeljemo nov algebraičen pristop k analizi časovnih omrežij, ki temelji na časovnih količinah nad ustreznim polkolobarjem. Definiramo polkolobarje za analizo časovnih omrežij brez potovalnega in čakalnega časa in polkolobarje za analizo časovnih omrežij, ko potovalni in čakalni časi niso ničelni. Za omrežja brez potovalnih in čakalnih časov razvijemo algoritme za učinkovito računanje s časovnimi količinami in za izbrane mere pomembnosti, ki so posplošitve mer za analizo statičnih omrežij. Opišemo izračun stopnje vozlišč, nakopičenosti, dostopnosti in vmesnosti ter rekurzivnih mer pomembnosti. Razdelamo postopke za izračun mere pomembnosti glede na lastne vektorje sosednostne matrike omrežja, Katzove pomembnosti in Bonacichevih pomembnosti α in (α, β). Opišemo tudi postopka HITS (kazala in viri) in pageRank. Definiramo dejavnost in privlačnost vozlišč. Mere pomembnosti nam omogočajo identifikacijo skupine najpomembnejših vozlišč omrežja in z njihovim pregledom / primerjavo vpogled v spreminjanje njihove vloge skozi čas. Povemo, kako izračunamo ovojnico nad absorpcijskimi polkolobarji in z njeno uporabo časovno dosegljivost. S pomočjo dosegljivosti izračunamo časovno razbitje na šibke in krepke komponente. Opisani pristop lahko uporabimo za analizo skupin, ki so določene s poljubno drugo časovno ekvivalenčno relacijo na danem omrežju.

Opisani postopki so dostopni kot Pythonska knjižnica TQ. Algoritme preizkusimo na realnih omrežjih, Franzosijevem omrežju nasilja v Italiji in omrežju Reutersovih novic o terorističnem napadu 11. septembra.

Math. Subj. Class. (2010): 05C25, 05C50, 05C85, 68R10, 90B10, 90C35, 91D30, 16Y60, 93C55

Ključne besede: časovne količine, časovno omrežje, potovalni čas, polkolobar, mere pomembnosti, skupina, dosegljivost, povezanosti, algoritem, Pythonska knjižnica, nasilje.

Abstract

In the thesis we describe a new algebraic approach to the temporal network analysis based on the notion of temporal quantities. We define semirings for the analysis of temporal networks with zero latency and zero waiting time and semirings for the analysis of temporal networks where the latency is given. For temporal networks with zero latency and zero waiting time we present algorithms for the efficient operations with temporal quantities and for the computation of chosen centrality measures that are generalized cases of centrality measures for static networks. We describe the computation of degree, clustering coefficients, closeness, betweenness and recursive measures of centrality. We explain the eigenvector centrality, the Katz centrality measure, the Bonacich α and (α, β) centralities, the HITS (hubs and authorities) centrality, and the pageRank centrality. We define activity and attraction coefficients in temporal networks. Centrality measures allow us to identify groups of important vertices. With a review / comparison of the vertices from the groups we get an insight into their roles through time. We present the algorithm for computing the closure of a temporal network over an absorptive semiring and for computing the temporal reachability of nodes. We also describe the procedure for computing temporal weak and strong connectivity components using the appropriate closure. This approach can also be used for the analysis of groups that are determined by other equivalence relations on the given network. The described procedures are available as a Python library TQ. We tested the algorithms on real networks, the Franzosi's violence network and the Reuters terror news network.

Math. Subj. Class. (2010): 05C25, 05C50, 05C85, 68R10, 90B10, 90C35, 91D30, 16Y60, 93C55
Keywords: temporal quantity, temporal network, latency, semiring, centrality measure, group, reachability, connectivity, algorithm, Python library, violence.

Kazalo

Slike

Algoritmi

Poglavje 1

Uvod

Besedo omrežje pogosto uporabljamo za oznako relacijskih podatkov, ki se pojavljajo na praktično vseh področjih našega življenja. Zato termin analiza omrežij različnim ljudem pomeni različne stvari. V slovenščini imamo še dodatno težavo, saj angleško besedo *network* različne znanstvene discipline prevajajo različno. Biologi in zdravniki govorijo o nevronskih mrežah, sociologi o družabnih in družbenih omrežjih, informatiki o računalniških mrežah. Kemiki raziskujejo molekule, opazujemo tudi trgovske mreže itd. Matematiki smo se odločili za omrežje iz preprostega razloga – beseda mreža je že rezervirana za algebrsko strukturo.

Analiza omrežij se pri načrtovanju projektov, pri raziskovanju električnega ali vodovodnega omrežja in pri raziskavah družbenih in družabnih omrežij uporablja v različne namene. Uporabljamo jo v transportnih sistemih in v logistiki. Poznamo omrežje komunikacij, epidemiološka omrežja (raziskovanje širjenja okužb) in omrežja v bioinformatiki. Zelo veliko raziskav dela z internetnim omrežjem, z omrežji telefonskih klicev in z omrežji elektronske pošte. Analizo omrežij uporabljamo pri analizi besedil, v bibliometriki (omrežja citiranj, soavtorstev itd.) in v organizacijski teoriji. Uporabna je tudi pri genomskem raziskovanju, pri analizi dogodkov in za raziskovanje omrežja plenilcev in plena. Omrežja se pojavljajo pri raziskavah metabolizma in presnavljanja beljakovin ter pri simulacijah prometa. Imamo tudi primere terorističnih omrežij, transakcijska omrežja v bančništvu, omrežja trgovin, omrežja mednarodnih sodelovanj in omrežja trgovanja z vrednostnimi papirji. Z omrežji lahko opišemo razvoz paketov, širjenje govoric ali informacij in odnose v družbi ali organizaciji. Večino omrežij ustvari človeška dejavnost in imajo podobno zgradbo.

V resnici imamo vsi uporabniki analize omrežij opravka z relacijskimi podatki, pri katerih gre za akterje (nevrone, gene, atome, ljudi, skupine, organizacije, države, računalnike, uporabniška imena na Facebooku in drugje na internetu itd.) in za relacije med temi akterji. Nekatere relacije so očitne, fizične (bančne transakcije, prenos sredstev, prodaja, sodelovanje, preseljevanje, menjanje službe). Druge

niso tako jasno definirane in so odvisne od osebnih izbir (prijateljstvo, všečnost, spoštovanje). Relacije med akterji so tako različne kot znanstvene discipline, ki jih raziskujejo. Lahko so povezane s številom (dolžina ceste), lahko so različnih vrst (brat / sestra, formalne vloge), lahko je edina stvar, ki nas o relaciji zanima, ali relacija obstaja ali ne. Nekatere relacije so obojestranske (ceste), nekatere imajo očitno smer (bančne transakcije).

Na področje analize omrežij lahko gledamo kot na samostojno področje. V matematiki ima korenine v teoriji grafov in v diskretni matematiki. Še posebej dolgo zgodovino ima analiza omrežij v družbenih znanostih, zanimanje za analizo omrežij pa se je v zadnjem času znatno povečalo zaradi večje dostopnosti velike količine podatkov in globalnega interesa po analizi teh podatkov. Internet je bliskovito narasel na ogromen in skoraj neobvladljiv vir, zato so narasle tudi raziskovalne težnje po boljših in predvsem učinkovitejših metodah za analizo velikih redkih omrežij. Še posebej zanimiva postajajo omrežja, v katerih upoštevamo tudi čas dogajanja in opazujemo spremembe omrežja skozi čas.

Vsa omenjena področja so združena pod formalnim okriljem, ki je dokaj enotno. V resnici je neverjetno, da smo za opis vseh različnih primerov, v katerih so interesi in zahtevnost raziskovanega problema različni, uspeli najti matematičen zapis, v katerega lahko pospravimo vse naštete primere in mnoge druge. Naraven način za matematičen opis omrežja predstavlja diskretna struktura, ki jo imenujemo graf.

V doktorski disertaciji obravnavamo primer, ki je razširitev osnovnega modela opisa relacijskih pojavov; primer, v katerem se relacije in akterji lahko spreminjajo s časom. Definiramo matematičen opis, ki naj čim bolj univerzalno opiše časovne relacije in časovne spremembe v danem omrežju. Akterji lahko izginjajo ali se pojavljajo, lahko se spremeni vrednost, način ali vrsta relacije med njimi. Za poenostavljen opis razvijemo tudi algoritmično / računalniško podporo.

1.1 Definicije in oznake

Definicije so povzete po naslednjih virih [22, 38, 53].

Definicija 1.1. *Graf* $\mathcal{G}$ je urejen par $(\mathcal{V}, \mathcal{L})$ dveh množic, množica $\mathcal{V}$ je množica *vozlišč* in množica $\mathcal{L}$ je množica *povezav* med pari vozlišč. Povezave med vozliščema u in v so lahko *usmerjene* (u, v) ali *neusmerjene* $\{u, v\}$. Vozlišči u in v sta *krajišči* povezave in sta si *sosedni*. Z $\ell(u, v)$ zapišemo, da povezava ℓ vodi iz vozlišča u v vozlišče v. Če za usmerjeno povezavo ℓ velja $\ell(u, v)$, pravimo, da ima ℓ *začetek* u in *konec* v. Povezavo, ki povezuje vozlišče s samim sabo, imenujemo *zanka*.

Z n označujemo število vozlišč $|\mathcal{V}|$ v omrežju in z m število povezav $|\mathcal{L}|$. Privzamemo, da sta n in m končna.

Definicija 1.2. Povezavi sta *vzporedni*, če povezujeta isti krajišči v isti smeri. Graf brez vzporednih povezav imenujemo *enostavni graf*. Graf $\mathcal{G}' = (\mathcal{V}', \mathcal{L}')$ je *podgraf* grafa $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, če velja $\mathcal{V}' \subseteq \mathcal{V}$ in $\mathcal{L}' \subseteq \mathcal{L}$.

Definicija 1.3. *Usmerjen graf* je graf, ki ima samo usmerjene povezave – urejene pare vozlišč. Ločimo med dvema tipoma sosedov. *Vhodni sosedje* vozlišča u so vozlišča $v \in \mathcal{V}$, za katera obstaja povezava $\ell(v,u) \in \mathcal{L}$. *Izhodni sosedje* vozlišča u so vozlišča $v \in \mathcal{V}$, če obstaja povezava $\ell(u,v) \in \mathcal{L}$. Množico vseh sosedov vozlišča v označimo z $N(v)$, množico izhodnih sosedov z $N^+(v)$ in množico vhodnih sosedov z $N^-(v)$.

Pogosto je uporabno dodati vrednosti (uteži) na vozlišča ali na povezave grafa $\mathcal{G}$. Vrednosti uteži lahko predstavljajo kapacitete, dolžine, potovalni čas, ceno, podobnost itd. Uteži na povezavah definiramo kot funkcijo $w : \mathcal{L} \to A$, vrednosti v vozliščih pa kot funkcijo $c : \mathcal{V} \to A$, kjer je A ustrezna množica. Imena funkcij običajno izberemo tako, da nakazujejo, kaj smo želeli označiti z njihovimi vrednostmi.

Definicija 1.4. *Omrežje* $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ sestavljajo graf $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ z dodatnimi informacijami o utežeh $\mathcal{W}$ na povezavah in vrednostih (lastnostih) $\mathcal{P}$ v vozliščih.

Definicija 1.5. *Stopnjo* vozlišča v v grafu označimo z $\deg(v)$ in predstavlja število povezav, ki imajo v za krajišče. *Izhodna stopnja* $\mathrm{outdeg}(v)$ je enaka številu povezav, po katerih lahko zapustimo vozlišče v; *vhodna stopnja* $\mathrm{indeg}(v)$ pa je enaka številu povezav, po katerih lahko pridemo v vozlišče v.

Maksimalno in minimalno stopnjo vozlišča v grafu $\mathcal{G}$ označujemo z Δ in z δ.

Definicija 1.6. *Sprehod* v grafu $\mathcal{G}$ z *začetkom* v vozlišču v_0 in *koncem* v vozlišču v_p je končno alternirajoče zaporedje vozlišč in povezav

$$\pi = v_0 \ell_1 v_1 \ell_2 v_2 \ldots \ell_p v_p,$$

če velja $\ell_i(v_{i-1}, v_i),\ i = 1, 2, \ldots, p$. *Dolžina* sprehoda je število p povezav, ki jih sprehod vsebuje. Zaporedje π je *polsprehod* ali *veriga,* če ne upoštevamo usmerjenosti povezav, torej če velja $\ell_i(v_{i-1}, v_i)$ ali pa $\ell_i(v_i, v_{i-1})$ za vsak $i = 1, 2, \ldots, p$. Sprehod je *sklenjen,* če se začne in konča v isti točki, $v_0 = v_p$. Sprehod, v katerem se povezave ne ponavljajo, je *enostaven*. Sprehod je *osnoven*, če so vsa vozlišča, skozi katera gre, razen začetka in konca, med seboj različna. Tak sprehod imenujemo tudi *pot*. Vsak osnoven sprehod je tudi enostaven. Sklenjeno pot imenujemo *cikel*.

Definicija 1.7. Če obstaja sprehod od u do v, pravimo, da je vozlišče v *dosegljivo* iz vozlišča u. Če med vozliščema u in v obstaja polsprehod, sta vozlišči *šibko* povezani. Če je vozlišče u dosegljivo iz vozlišča v in obratno, sta vozlišči *krepko* povezani. Šibka in krepka povezanost določata ekvivalenčni relaciji na množici vozlišč grafa. Pripadajoči ekvivalenčni razredi določajo šibke in krepke *komponente* grafa. Šibke komponente so komponente grafa, v katerem ne upoštevamo usmerjenosti povezav. Krepke komponente vsebujejo le vozlišča, ki so vzajemno dosegljiva.

Definicija 1.8. Ena od možnih predstavitev grafa $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ je s *sosednostno matriko* $\mathbf{A}$ velikosti $n \times n$, ki jo definiramo kot

$$\mathbf{A} = [a_{uv}]_{u,v \in \mathcal{V}} = \begin{cases} 1, & (u,v) \in \mathcal{L}, \\ 0, & \text{sicer.} \end{cases}$$

Če je graf utežen po povezavah, namesto 1 na ustrezno mesto v sosednostni matriki postavimo vrednost uteži na povezavi, $w(u,v)$, in njeni elementi ustrezajo vrednostim na povezavah omrežja.

Definicija 1.9. *Vrednostna (prehodna) matrika* $\mathbf{A}$ omrežja $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ je definirana kot

$$\mathbf{A} = [a_{uv}]_{u,v \in \mathcal{V}} = \begin{cases} w(u,v), & (u,v) \in \mathcal{L}, \\ 0, & \text{sicer.} \end{cases}$$

Vrednostna matrika neusmerjenega omrežja je simetrična, kar v splošnem ne drži za usmerjena omrežja. Če omrežje nima zank, so diagonalni elementi vrednostne matrike enaki 0.

Definicija 1.10. Naj bo dana matrika $\mathbf{A} \in \mathbb{R}^{n \times n}$. *(Desni) lastni vektor* matrike $\mathbf{A}$ je neničeln vektor $x \in \mathbb{C}^n$, za katerega velja $\mathbf{A}x = \lambda x$ za neko kompleksno število $\lambda \in \mathbb{C}$. Številu λ pravimo *lastna vrednost* matrike $\mathbf{A}$, ki pripada lastnemu vektorju x. Podobno definiramo *levi lastni vektor* in *levo lastno vrednost*.

Lastni vrednosti $\lambda_{\max}$, ki ima največjo absolutno vrednost med vsemi lastnimi vrednostmi matrike, pravimo *dominantna lastna vrednost,* pripadajočemu lastnemu vektorju $x_{\max}$ pa *dominantni lastni vektor*. Paru $(\lambda_{\max}, x_{\max})$ pravimo *dominantni lastni par.*

Definicija 1.11. *Lastne vrednosti omrežja* $\mathcal{N}$ so definirane kot lastne vrednosti njegove vrednostne matrike $\mathbf{A}$. Množici vseh lastnih vrednosti omrežja pravimo *spekter* omrežja $\mathcal{N}$.

Spekter omrežja veliko pove o njegovih lastnostih. O lastnostih omrežja, ki sledijo iz lastnih vrednosti vrednostnih matrik, razen pri izpeljavi mer pomembnosti, ne bomo govorili. Več o tem lahko preberete v [24, 51, 65].

Če je $\mathbf{A}$ realna simetrična matrika, so njene lastne vrednosti realne. Za nenegativne matrike velja še več.

Izrek 1.1. *(Perron-Frobeniusov izrek) Naj bo* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *nenegativna matrika (vsi njeni elementi so nenegativni). Potem ima matrika* $\mathbf{A}$ *nenegativno realno dominantno lastno vrednost* λ*. Lastni vektor, ki pripada lastni vrednosti* λ*, je nenegativen realen vektor.*

Če velja tudi, da matrika $\mathbf{A}$ *nima bločno trikotne dekompozicije (to je, ne vsebuje* $k \times (n-k)$ *bloka samih ničel za neko permutacijo vrstic in stolpcev), potem je* λ *enostavna lastna vrednost in pripadajoči lastni vektor je pozitiven.*

Če je matrika $\mathbf{A}$ *pozitivna, ima* λ *strogo največjo absolutno vrednost med vsemi lastnimi vrednostmi.*

Posledica 1.1. *Krepko povezano omrežje* $\mathcal{N}$ *z nenegativnimi utežmi ima enostavno pozitivno lastno vrednost* $\lambda_{\max}$*. Lastni vektor, ki pripada* $\lambda_{\max}$*, je pozitiven. Če omrežje ni krepko povezano, enostavnost lastne vrednosti ni zagotovljena.*

Dokaz. Dokaz izreka in posledice v [10, 61]. □

Definicija 1.12. Za funkciji $f : \mathbb{N} \to \mathbb{N}$ in $g : \mathbb{N} \to \mathbb{N}$ rečemo, da je f *reda* $\mathcal{O}(g)$, če obstajata taki pozitivni konstanti $n_0 \in \mathbb{N}$ in $c \in \mathbb{R}^+$, da velja $f(n) \leq c \cdot g(n)$ za vse $n \geq n_0$.

Notacija $\mathcal{O}$ je uporabna za ocenjevanje asimptotske rasti funkcij in jo bomo uporabljali za opis časovne zahtevnosti algoritmov.

Velja $0 \in \mathbb{N}$. Uvedimo še oznake $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$, $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{\pm\infty\}$, $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ in $\overline{\mathbb{R}_0^+} = \mathbb{R}_0^+ \cup \{\infty\}$.

1.2 Zgradba doktorske disertacije

Disertacija je organizirana po poglavjih. V drugem poglavju definiramo polkolobar in opišemo uporabo polkolobarjev v analizi omrežij. Predstavimo tudi nekaj primerov. V tretjem poglavju definiramo in opišemo najpogosteje uporabljane mere pomembnosti v omrežjih. Numerične postopke za izračun približka za dominantni lastni par matrike in približka za rešitev sistema linearnih enačb na kratko povzamemo v četrtem poglavju. Nekaj obstoječih pristopov k analizi časovnih omrežij opišemo v petem poglavju.

V šestem poglavju najprej predstavimo osnovne definicije našega novega pristopa k analizi časovnih omrežij. V tem poglavju vpeljemo časovne količine in nekaj polkolobarjev časovnih količin. Polkolobarji za analizo časovnih omrežij so izvirni prispevek, ki smo ga opisali v [6, 73]. Opisani so novi časovni polkolobarji za analizo časovnih omrežij z ničelnimi potovanimi in čakalnimi časi in izvirna algebraična struktura (potovalni polkolobarji), ki poleg potovalnih časov upošteva tudi poljubne dodatne informacije na povezavah. V prihodnje nameravamo poiskati več primerov uporabe in razviti algoritme za predstavljene operacije.

V sedmem poglavju razvijemo postopke za analizo časovnih omrežij brez potovalnega in čakalnega časa, ki temeljijo na polkolobarjih iz šestega poglavja. Pristop predstavlja naravnejšo vzdolžno alternativo običajni prečni analizi, ki temelji na časovnih rezinah. Tudi rezultati analiz so izraženi kot časovne količine. Razvite postopke smo opisali v članku [6]. Razdelali smo časovna razbitja, ki opisujejo spreminjanje šibkih in krepkih komponent povezanosti omrežja. Podoben pristop lahko uporabimo tudi za analizo skupin, ki so določene s poljubno drugo časovno ekvivalenčno relacijo v danem omrežju. Predstavitev časovnih omrežij brez potovalnih in čakalnih časov je taka, da jo lahko uporabljamo v poljubnem kontekstu in ni ozko usmerjena v reševanje konkretnega primera, kar velja za večino obstoječih rešitev. Opisani postopki so dostopni kot Pythonska knjižnica TQ. Razvijamo tudi uporabniku prijaznejši vmesnik Ianus (po zgledu programa Pajek), za katerega upamo, da bo sčasoma postal široko uporabljan v akademskih krogih in tudi v praksi.

V osmem poglavju opišemo postopke za izračun mer pomembnosti časovnega omrežja brez potovalnih in čakalnih časov. Predstavili smo jih v člankih [6, 72, 73] in so posplošitev mer pomembnosti za statična omrežja. Mere pomembnosti nam omogočajo identifikacijo skupin najpomembnejših vozlišč / povezav omrežja in z njihovim pregledom / primerjavo vpogled v spreminjanje njihove vloge skozi čas. Za dano lastnost / utež bi lahko v prihodnosti določali časovne prereze (*cuts*) ali otoke (*islands*).

V devetem poglavju analiziramo izbrana časovna omrežja, zapisana v predstavljeni obliki.

Postopke za analizo omrežij brez potovalnih in čakalnih časov, knjižnico TQ in program Ianus smo predstavili na treh mednarodnih konferencah in na nekaj seminarjih v Sloveniji.

Disertacijo zaključimo s kratkim povzetkom in smernicami za prihodnje delo.

Poglavje 2

Polkolobarji

V analizi omrežij se uporablja precej različnih polkolobarjev ([3, 14, 27, 39, 52, 66, 84]). Na tem mestu bomo omenili tiste, ki jih v nadaljevanju uporabljamo tudi pri analizi časovnih omrežij, in opisali motivacijo za njihov nastanek.

Definicija 2.1. Naj bodo $a, b, c \in A$. Množica A z dvomestnima operacijama seštevanje $\oplus$ in množenje $\odot$, nevtralnim elementom za seštevanje 0 in enoto za množenje 1, označimo jo z $A(\oplus, \odot, 0, 1)$, je *polkolobar*, kadar velja

- množica A je za seštevanje $\oplus$ komutativen monoid z nevtralnim elementom 0 (seštevanje je komutativno, $a \oplus b = b \oplus a$, asociativno, $a \oplus (b \oplus c) = (a \oplus b) \oplus c$, in za vsak element $a \in A$ velja $a \oplus 0 = 0 \oplus a = a$);
- množica A je za množenje $\odot$ monoid z enoto 1 (množenje je asociativno, $a \odot (b \odot c) = (a \odot b) \odot c$, in za vsak element $a \in A$ velja $a \odot 1 = 1 \odot a = a$);
- veljata distributivnostna zakona

$$a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c),$$
$$(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c);$$

- element 0 je ničla (absorpcijski element, anihilator) za množenje

$$a \odot 0 = 0 \odot a = 0 \text{ za vsak } a \in A.$$

V vseh primerih predpostavimo, da ima operacija množenja prednost pred seštevanjem. Zadnjo točko definicije polkolobarja nekateri avtorji izpuščajo. Potrebujemo jo zato, da lahko nad danim polkolobarjem zgradimo polkolobar matrik. Če je dana množica A, za katero veljajo vse točke definicije razen zadnje, lahko množico A razširimo z elementom $\mathbb{H}$, za katerega po definiciji velja

$$a \oplus \mathbb{H} = \mathbb{H} \oplus a = a \quad \text{in} \quad a \odot \mathbb{H} = \mathbb{H} \odot a = \mathbb{H}$$

za vse $a \in A \cup \{\text{⌘}\}$. V razširjeni množici $A_{\text{⌘}} = A \cup \{\text{⌘}\}$ je element ⌘ ničla za množenje po definiciji in $(A_{\text{⌘}}, \oplus, \odot, \text{⌘}, 1)$ je polkolobar, v katerem veljajo vse točke definicije 2.1.

Glavna razlika med algebrskimi strukturami, ki smo jih vajeni, in polkolobarjem je v tem, da v polkolobarju obstoj inverznega elementa ni zagotovljen (v preprostem jeziku to pomeni, da v splošnem ne znamo odštevati ali deliti) in da množenje ni komutativno.

Definicija 2.2. Polkolobar je *poln*, če je seštevanje dobro definirano tudi za števne množice in tudi v tem primeru veljajo pravila za vsoto in distributivnostna zakona:

$$\bigoplus_{i\in\emptyset} a_i = 0, \quad \bigoplus_{i\in\{j\}} a_i = a_j, \quad \bigoplus_{i\in\{j,k\}} a_i = a_j \oplus a_k \text{ za } j \neq k,$$

$$\bigoplus_{j\in J}\left(\bigoplus_{i\in I_j} a_i\right) = \bigoplus_{i\in I} a_i, \text{ kjer je } \bigcup_{j\in J} I_j = I \text{ in } I_j \cap I_k = \emptyset, j \neq k,$$

in

$$\bigoplus_{i\in I}(a \odot a_i) = a \odot \left(\bigoplus_{i\in I} a_i\right), \quad \bigoplus_{i\in I}(a_i \odot a) = \left(\bigoplus_{i\in I} a_i\right) \odot a.$$

V tem primeru velja tudi

$$\left(\left(\bigoplus_i a_i\right) \odot \left(\bigoplus_j b_j\right)\right) = \bigoplus_{i,j} a_i \odot b_j = \bigoplus_i \left(\bigoplus_j (a_i \odot b_j)\right).$$

Definicija 2.3. Seštevanje je *idempotentno,* če za vsak $a \in A$ velja $a \oplus a = a$.

Definicija 2.4. Poln polkolobar $(A, \oplus, \odot, 0, 1)$ je *zaprt* natanko tedaj, ko je v njem definirana še dodatna unarna operacija *zaprtje* (ovojnica) $\star$, za katero velja

$$a^\star = 1 \oplus (a \odot a^\star) = 1 \oplus (a^\star \odot a) \text{ za vsak } a \in A.$$

V istem polkolobarju lahko obstajajo različna zaprtja. V zaprtem polkolobarju definiramo *strogo zaprtje* $\overline{a}$ kot

$$\overline{a} = a \odot a^\star.$$

Poln polkolobar je zaprt, če operacijo zaprtja definiramo kot

$$a^\star = \bigoplus_{k\geq 0} a^k. \tag{2.1}$$

V nadaljevanju bomo z izrazom zaprtje označevali operacijo iz enačbe (2.1), če ne bomo posebej omenili, za katero zaprtje gre.

V polkolobarjih, v katerih sta smiselni tudi operaciji odštevanja in deljenja, lahko definiramo $a^\star = (1-a)^{-1}$. Primer takega polkolobarja so realna števila za običajno seštevanje in množenje. V množici realnih števil vrednost $1^\star$ ni definirana, kar popravimo tako, da množici dodamo element ∞ in definiramo $1^\star = \infty$ [39].

Definicija 2.5. V polkolobarju velja *absorpcijski zakon,* če za vse $a, b, c \in A$ velja

$$(a \odot b) \oplus (a \odot c \odot b) = a \odot b.$$

Ker veljata distributivnostna zakona in obstaja enota za množenje, je za veljavnost absorpcijskega zakona dovolj preveriti $1 \oplus c = 1$ za vsak $c \in A$. V polkolobarjih, v katerih velja absorpcijski zakon, je $a^\star = 1$ za vse $a \in A$. Polkolobar, v katerem velja absorpcijski zakon, je idempotenten.

Definicija 2.6. Nad polkolobarjem $(A, \oplus, \odot, 0, 1)$ lahko zgradimo *polkolobar kvadratnih matrik* $A^{n\times n}$ reda n, katerih elementi so vrednosti iz polkolobarja A. Seštevanje in množenje v matričnem polkolobarju definiramo na običajen način:

$$\begin{aligned}(\mathbf{A} \oplus \mathbf{B})_{ij} &= a_{ij} \oplus b_{ij},\\ (\mathbf{A} \odot \mathbf{B})_{ij} &= \bigoplus_{k=1}^{n} a_{ik} \odot b_{kj}, \quad i, j = 1, 2, \ldots, n.\end{aligned}$$

Opozorimo, da znaki za operacije na levi in na desni strani enačb ne predstavljajo istih operacij. Na levi so operacije, ki delujejo v matričnem polkolobarju $A^{n\times n}$, na desni pa operacije, ki delujejo v polkolobarju A. Nevtralni element za seštevanje v polkolobarju $A^{n\times n}$ je matrika, katere elementi so enaki 0 (nevtralni element za $\oplus$ v polkolobarju A). Enota za množenje v $A^{n\times n}$ je matrika, ki ima na diagonali elemente enake 1 (enota polkolobarja A) in izven diagonale ničlo 0 (tukaj potrebujemo zadnjo točko definicije polkolobarja).

Trditev 2.1. *Množica $A^{n\times n}$ je polkolobar za operaciji seštevanja in množenja ter za nevtralni element in enoto iz definicije 2.6. Če je polkolobar A poln, je poln tudi matrični polkolobar $A^{n\times n}$ in je zato zaprt za operacijo*

$$\mathbf{A}^\star = \bigoplus_{k\geq 0} \mathbf{A}^k.$$

Dokaz. Dokaz v [52]. □

Za izračun zaprtja $\mathbf{A}^\star$ matrike $\mathbf{A}$ nad polnim polkolobarjem $(A, \oplus, \odot, 0, 1)$ lahko uporabimo Fletcherjev algoritem 1, opisan v [43].

Strogo zaprtje $\overline{\mathbf{A}}$ matrike $\mathbf{A}$ dobimo, če izbrišemo vrstico 7 v algoritmu 1. Če je seštevanje idempotentno, lahko matriko računamo na mestu (ne potrebujemo spodnjega indeksa v algoritmu).

Algoritem 1 Fletcherjev algoritem.

1: **function** *Fletcher*($\mathbf{A}$)
2: $\quad\mathbf{A}_0 \leftarrow \mathbf{A}$
3: $\quad$**for** $k = 1$ **to** n **do**
4: $\quad\quad$**for** $i = 1$ **to** n **do**
5: $\quad\quad\quad$**for** $j = 1$ **to** n **do**
6: $\quad\quad\quad\quad a_k[i,j] \leftarrow a_{k-1}[i,j] \oplus a_{k-1}[i,k] \odot (a_{k-1}[k,k])^\star \odot a_{k-1}[k,j]$
7: $\quad\quad a_k[k,k] \leftarrow 1 + a_k[k,k]$
$\quad$**return** $\mathbf{A}_n$ $\quad\triangleright$ matrika $\mathbf{A}_n$ je zaprtje matrike $\mathbf{A}$

V polkolobarjih, v katerih velja absorpcijski zakon, lahko Fletcherjev algoritem poenostavimo tako, da v vrstici 6 izračunamo vrednost $a_k[i,j] \leftarrow a_{k-1}[i,j] \oplus a_{k-1}[i,k] \odot a_{k-1}[k,j]$ (upoštevamo $(a_{k-1}[k,k])^\star = 1$).

V večini polkolobarjev, ki se uporabljajo v analizi omrežij (z izjemo kombinatoričnega polkolobarja), absorpcijski zakon velja.

Pomemben razred polkolobarjev predstavljajo polkolobarji, v katerih lahko definiramo relacijo urejenosti.

Definicija 2.7. Dvomestna relacija $\preceq$ na množici A je *delna urejenost*, če je refleksivna ($a \preceq a$), tranzitivna (če velja $a \preceq b$ in $b \preceq c$, potem velja $a \preceq c$) in antisimetrična (če velja $a \preceq b$ in $b \preceq a$, potem velja $a = b$). Delna urejenost je *popolna (linearna) urejenost*, če je vsak par elementov primerljiv (za vsak par $a, b \in A$ velja ali $a \preceq b$ ali $b \preceq a$).

Definicija 2.8. Polkolobar $(A, \oplus, \odot, 0, 1)$ je *urejen*, če v njem obstaja delna urejenost $\preceq$, ki je usklajena z operacijama:

$a \oplus c \preceq b \oplus c$ za vse $a, b, c \in A$ in $a \preceq b$,
$a \odot c \preceq b \odot c$ in $c \odot a \preceq c \odot b$ za vse $a, b, c \in A, 0 \preceq c, a \preceq b$.

Definicija 2.9. Če je relacija $\preceq$, definirana kot

$$a \preceq b \Leftrightarrow \exists c \in A : a \oplus c = b,$$

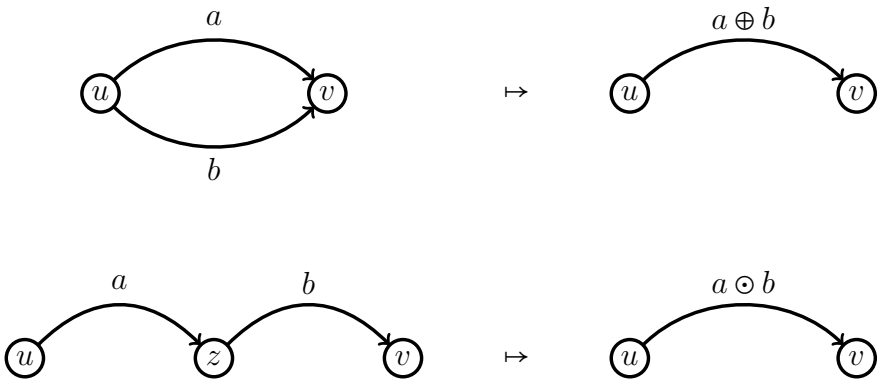
delna urejenost v A, potem tej relaciji pravimo *naravna ureditev* množice A.

Če je operacija seštevanje $\oplus$ idempotentna, je A naravno urejena.
Če je polkolobar A urejen, je urejen tudi polkolobar matrik nad A.

2.1 Uporaba polkolobarjev v analizi omrežij

Ker imamo v omrežju uteži na povezavah, ki opisujejo nek pojav, bi radi našli način, s katerim bi njihove vrednosti smiselno združevali. Sama vrednost na povezavi nam ne pove veliko o omrežju, če je ne znamo uporabiti. Zanima nas, kako ustrezno združujemo vrednosti na različnih povezavah.

Zgodita se lahko dva osnovna primera (slika 2.1) – želimo združiti vrednosti na vzporednih ali na zaporednih povezavah. Če si povezave predstavljamo kot potovanja med točkami, imamo intuitivno predstavo, kakšnim zahtevam morajo operacije zadoščati. Naj seštevanje ustreza združevanju vzporednih povezav in množenje združevanju zaporednih povezav.



Slika 2.1: Polkolobarsko seštevanje in množenje v omrežju.

Če združujemo povezave, nočemo, da bi bila vrednost združenih povezav odvisna od vrstnega reda združevanja, torej potrebujemo asociativnost. Želimo imeti možnost povedati, da povezava ne obstaja, zato potrebujemo ničlo 0. Želimo, da sta vzporedni povezavi enakovredni, saj nobena ni "prva." Od tod zahteva po komutativnosti za seštevanje. Pri množenju včasih želimo nekomutativnost, saj velikokrat imamo "prvo" povezavo in naslednje. Kadar imamo namesto povezav sprehode, želimo imeti pravilo, po katerem lahko združimo njihove vrednosti. Tukaj pride do izraza distributivnost.

Izkaže se, da tem zahtevam ustreza algebrska struktura polkolobar, če dodamo še tehnično zahtevo o obstoju enote za množenje.

Definicija 2.10. Dani so omrežje $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$, polkolobar $(A, \oplus, \odot, 0, 1)$ in uteži $w : \mathcal{L} \to A$. Uteži w razširimo na *vrednosti sprehodov* in množic sprehodov v omrežju $\mathcal{N}$:

- za ničelni sprehod ρ_v v vozlišču $v \in \mathcal{V}$ je vrednost sprehoda $w(\rho_v) = 1$;

- za neničelni sprehod $\pi = v_0\ell_1v_1\ell_2v_2\dots\ell_pv_p,\ p \geq 1$, v omrežju $\mathcal{N}$ je vrednost sprehoda

$$w(\pi) = w(\ell_1) \odot w(\ell_2) \odot \cdots \odot w(\ell_p) = \bigodot_{\ell\in\pi} w(\ell);$$

- prazna množica sprehodov $\emptyset$ ima vrednost $w(\emptyset) = 0$;

- če je A poln polkolobar, ima števna neprazna množica sprehodov $\mathcal{P} = \{\pi_1, \pi_2, \dots\}$ v omrežju $\mathcal{N}$ vrednost

$$w(\mathcal{P}) = w(\pi_1) \oplus w(\pi_2) \oplus \cdots = \bigoplus_{\pi\in\mathcal{P}} w(\pi).$$

  Zveza vselej velja za končne množice sprehodov. Če je polkolobar poln, jo lahko razširimo na števno neskončne;

- če sta $\pi_1 = v_0\ell_1v_1\ell_2v_2\dots\ell_pv_p$ in $\pi_2 = u_0l_1u_1l_2u_2\dots l_qu_q$ taka sprehoda, da je konec v_p sprehoda π_1 enak začetku u_0 sprehoda π_2, lahko sprehoda π_1 in π_2 staknemo v nov sprehod $\pi_1 \circ \pi_2 = v_0\ell_1v_1\dots\ell_pv_pl_1u_1l_2\dots l_qu_q$. Vrednost staknjenega sprehoda je enaka

$$w(\pi_1 \circ \pi_2) = w(\pi_1) \odot w(\pi_2).$$

Trditev 2.2. *Vrednosti sprehodov v grafu so tesno povezane z vrednostnimi matrikami: Naj bo $\mathbf{A} \in A^{n\times n}$ vrednostna matrika omrežja $\mathcal{N}$. Označimo s $\mathcal{P}_{uv}^p$ množico vseh sprehodov od vozlišča u do vozlišča v dolžine p, s $\mathcal{P}_{uv}^\star$ množico vseh sprehodov od vozlišča u do vozlišča v in s $\overline{\mathcal{P}}_{u,v}$ množico vseh neničelnih sprehodov od vozlišča u do vozlišča v. Potem je*

$$w(\mathcal{P}_{uv}^p) = (\mathbf{A}^p)_{uv}.$$

Če je osnovni polkolobar A poln in sta $\mathbf{A}^\star$ zaprtje matrike $\mathbf{A}$ in $\overline{\mathbf{A}}$ strogo zaprtje matrike $\mathbf{A}$, potem je

$$w(\mathcal{P}_{uv}^\star) = (\mathbf{A}^\star)_{uv} \quad \textit{in} \quad w(\overline{\mathcal{P}}_{uv}) = (\overline{\mathbf{A}})_{uv}.$$

Če je polkolobar absorpcijski, velja še $w(\mathcal{P}_{uv}^\star) = w(\mathcal{E}_{uv})$, kjer smo z $\mathcal{E}_{uv}$ označili množico vseh poti med vozliščema u in v.

Dokaz. Dokaz v [4]. □

Velik del analize omrežij lahko izvedemo s pomočjo zaprtja, ki ga uporabimo nad ustreznim polkolobarjem.

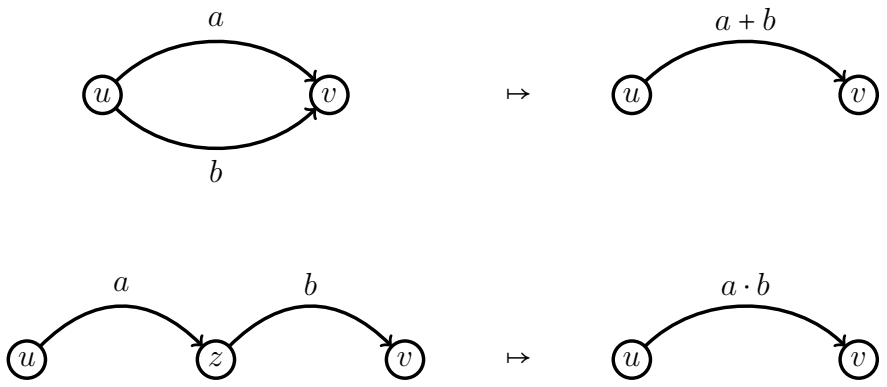
2.2 Primeri polkolobarjev

2.2.1 Kombinatorični polkolobar

Kombinatorični polkolobar imenujemo polkolobar naravnih števil za običajno seštevanje in množenje $(\overline{\mathbb{N}}, +, \cdot, 0, 1)$. Včasih namesto naravnih števil uporabimo druge številske množice, na primer $\overline{\mathbb{R}_0^+}$. Ta polkolobar je poln in zato tudi zaprt za $a^\star = \sum_{k\geq 0} a^k$. Seštevanje ni idempotentno in ne velja absorpcijski zakon.

V analizi omrežij kombinatorični polkolobar uporabljamo, kadar utež na povezavi omrežja vidimo kot število načinov za prečkanje povezave. Seštevanje in množenje v kombinatoričnem polkolobarju ustrezata pravilu vsote in pravilu produkta, ki smo ju vajeni iz kombinatoričnega preštevanja [74].

Dano je omrežje z vozliščema u in v ter vzporednima povezavama med njima. Vrednost uteži na eni povezavi je enaka a, vrednost uteži na drugi povezavi je enaka b. Od vozlišča u do vozlišča v lahko pridemo na $a + b$ načinov. Če imamo v omrežju tri vozlišča u, z in v ter zaporedni povezavi (u, z) z utežjo a in (z, v) z utežjo b, potem lahko od vozlišča u do vozlišča v pridemo na $a \cdot b$ načinov. Grafični prikaz povedanega je predstavljen na sliki 2.2.



Slika 2.2: V kombinatoričnem polkolobarju štejemo, na koliko načinov lahko pridemo iz enega v drugo vozlišče.

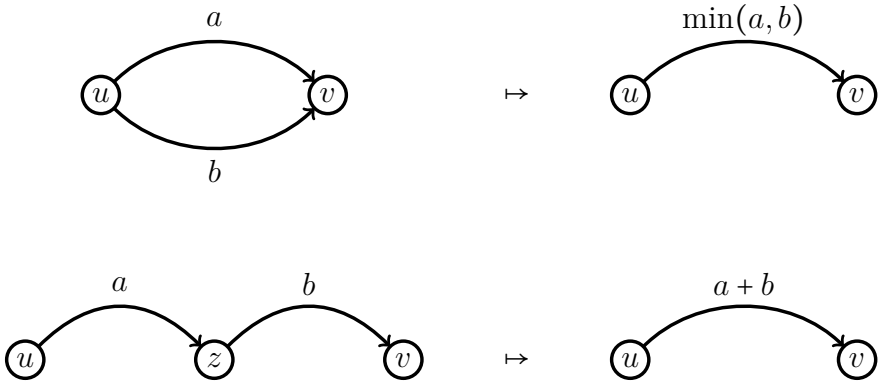
2.2.2 Polkolobar najkrajših poti

Običajno so dolžine poti nenegativne, zato je najpogostejša definicija polkolobarja najkrajših poti $(\overline{\mathbb{R}_0^+}, \min, +, \infty, 0)$. Polkolobar je poln in komutativen (tudi polkolobarsko množenje je komutativno). V njem velja absorpcijski zakon. Polkolobar je zaprt in $a^\star = \min\{0, a + a^\star\} = 0$ za vsak $a \in \overline{\mathbb{R}_0^+}$.

Včasih namesto množice $\overline{\mathbb{R}_0^+}$ uporabimo množico $\overline{\mathbb{N}}$. V tem primeru polkolobarju pravimo *tropski* polkolobar.

Motivacija za uporabo tega polkolobarja je klasični problem najkrajših poti: Dano je omrežje $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ z utežjo $w : \mathcal{L} \to \overline{\mathbb{R}_0^+}$ in vozlišče $s \in \mathcal{V}$ (izvor). Vrednost uteži $w(u, v)$ predstavlja dolžino povezave od vozlišča u do vozlišča v. Radi bi izračunali dolžine najkrajših poti od s do drugih vozlišč $v \in \mathcal{V}$.

Spet vzemimo omrežji kot pri kombinatoričnem polkolobarju. Ko imamo vzporedni povezavi med vozliščema u in v in je vrednost na eni povezavi enaka a, vrednost na drugi povezavi pa b, je najkrajša pot od vozlišča u do vozlišča v dolžine $\min(a, b)$. Če imamo zaporedni povezavi (u, z) z utežjo a in (z, v) z utežjo b, je najkrajša pot od vozlišča u do vozlišča v dolžine $a + b$. Grafični prikaz povedanega je predstavljen na sliki 2.3.



Slika 2.3: V polkolobarju najkrajših poti vrednosti na povezavah predstavljajo dolžino poti.

Običajen postopek za izračun dolžin d najkrajših poti od izvornega vozlišča s do drugih vozlišč $v \in \mathcal{V} \setminus \{s\}$ v omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ je, da definiramo $d(s) = 0$ in izračunamo razdaljo do vsakega vozlišča $v \in \mathcal{V} \setminus \{s\}$ s pomočjo Bellmanove enačbe

$$d(v) = \min_{u \in \mathcal{V}} \{d(u) + w(u, v)\}. \tag{2.2}$$

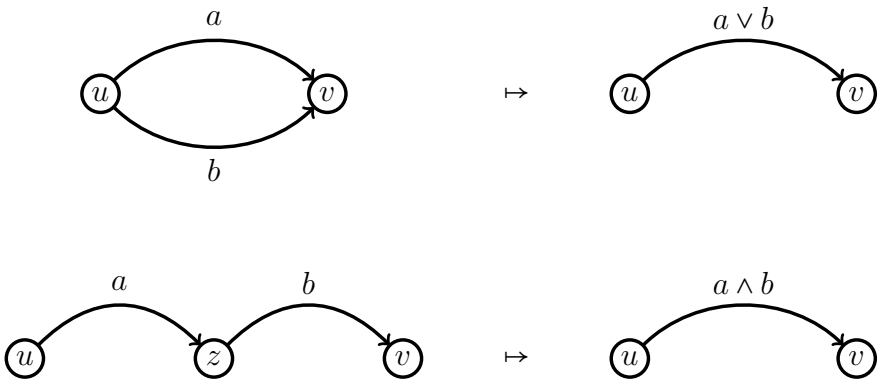
2.2.3 Logični ali povezanostni polkolobar

Logični ali povezanostni polkolobar je polkolobar $(\{0, 1\}, \vee, \wedge, 0, 1)$. V nekaterih znanostih ga (iz očitnega razloga) imenujejo logični ali Boolov polkolobar. V analizi omrežij mu rečemo povezanostni polkolobar, saj s pomočjo tega polkolobarja določamo dosegljivosti vozlišč. Povezanostni polkolobar je poln, zaprt in

$a^\star = 1 \vee (a \wedge a^\star) = 1$ za vsak $a \in \{0, 1\}$. V njem velja absorpcijski zakon.

Dosegljivost parov vozlišč v omrežju določamo s pomočjo zaprtja v polkolobarju matrik nad povezanostnim polkolobarjem.

Tudi tokrat najprej poglejmo primer vzporednih povezav med vozliščema u in v. Na eni povezavi je vrednost a, na drugi pa vrednost b. Vrednost 1 pomeni, da je povezava prehodna; vrednost 0 pa, da je prekinjena. Vozlišče v je dosegljivo iz vozlišča u natanko tedaj, ko je prehodna vsaj ena od povezav med njima, torej ko je vsaj ena od vrednosti a ali b enaka 1. Natanko v tem primeru ima izraz $a \vee b$ vrednost 1. V primeru zaporednih povezav (u, z) z utežjo a in (z, v) z utežjo b morata biti za dosegljivost vozlišča v iz vozlišča u prehodni obe povezavi, kar je res natanko tedaj, ko velja $a \wedge b = 1$. Grafični prikaz povedanega je predstavljen na sliki 2.4.



Slika 2.4: V povezanostnem polkolobarju opazujemo obstoj poti.

Produkt dvojiških matrik $\mathbf{A}$ in $\mathbf{B}$ na mestu uv je različen od 0 natanko tedaj, ko obstaja indeks (vozlišče) t, za katerega sta a_{ut} in b_{tv} enaka 1. Kvadrat sosednostne matrike $\mathbf{A}$ omrežja $\mathcal{N}$, izračunan v povezanostnem polkolobarju, ima vrednost 1 na mestu uv natanko tedaj, ko v omrežju $\mathcal{N}$ obstaja sprehod od u do v dolžine 2.

V splošnem $\mathbf{A}^k$ predstavlja sprehode dolžine k v omrežju $\mathcal{N}$. Vozlišče v je torej dosegljivo iz vozlišča u, če je na mestu uv enica v kateri od potenc matrike $\mathbf{A}$. Relacijo dosegljivosti lahko opišemo z matriko

$$\mathbf{I} \oplus \mathbf{A} \oplus \mathbf{A}^2 \oplus \ldots,$$

ki je enaka zaprtju matrike $\mathbf{A}$, če to zaprtje obstaja. Ker je povezanostni polkolobar zaprt, lahko s pomočjo operacije zaprtja $\mathbf{A}^\star$ izračunamo dosegljivost parov vozlišč v omrežju.

2.2.4 Geodezični polkolobar

V množici $A = \overline{\mathbb{R}_0^+} \times \overline{\mathbb{N}}$ definiramo seštevanje

$$(a,i) \oplus (b,j) = \left(\min(a,b), \begin{cases} i, & a < b \\ i+j, & a = b \\ j, & a > b \end{cases} \right)$$

in množenje

$$(a,i) \odot (b,j) = (a+b, i \cdot j).$$

Za ti operaciji je $\big(A, \oplus, \odot, (\infty, 0), (0,1)\big)$ poln zaprt polkolobar [4] za zaprtje

$$(a,i)^\star = \begin{cases} (0,\infty), & a = 0, i \neq 0, \\ (0,1), & \text{sicer.} \end{cases}$$

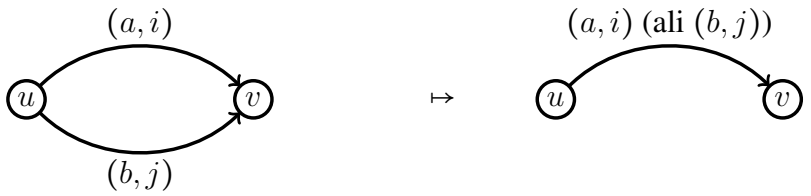
Imenujemo ga geodezični polkolobar. Polkolobar ni idempotenten.

Geodezični polkolobar je kombinacija polkolobarja najkrajših poti (ta je prva množica v kartezičnem produktu) in kombinatoričnega polkolobarja (ki je druga množica v kartezičnem produktu). Z njim hkrati izračunamo dolžino in število najkrajših poti med pari točk.

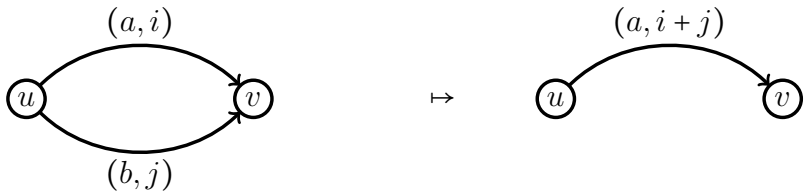
Poglejmo, kako seštevanje in množenje tolmačimo na osnovnih dveh primerih. V omrežju imamo vzporedni povezavi med vozliščema u in v, na prvi povezavi je vrednost (a,i), na drugi pa vrednost (b,j). Najkrajša pot je dolžine $\min(a,b)$, enako kot pri polkolobarju najkrajših poti, število najkrajših poti pa je odvisno od tega, kje dosežemo minimum. Če je najkrajša pot enolična, na primer tista, ki poteka po povezavi z utežjo (a,i), se število najkrajših poti ne spremeni in ostane enako i. Podobno velja, če enolična najkrajša pot poteka po povezavi z utežjo (b,j). Če sta dolžini poti po obeh povezavah enaki, se število najkrajših poti spremeni. Zdaj je enako vsoti $i+j$ števila najkrajših poti prek obeh povezav.

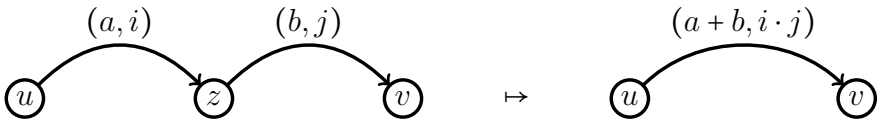
Če imamo v omrežju zaporedni povezavi (u,z) z utežjo (a,i) in (z,v) z utežjo (b,j), je dolžina najkrajše poti od u do v enaka $a+b$. Število teh poti je enako $i \cdot j$. Grafični prikaz je na sliki 2.5.

če je $a < b$ (ali $b < a$)



če je $a = b$





Slika 2.5: V geodezičnem polkolobarju iščemo dolžino najkrajše poti (prva komponenta) in štejemo število poti z najmanjšo dolžino (druga komponenta).

Poglavje 3

Mere pomembnosti

Določanje pomembnosti vozlišč omrežja je eden temeljnih pristopov v analizi omrežij. Začetnik iskanja bolj ali manj osrednjih vozlišč v omrežju je Alex Bavelas, ki je konec 40-ih let preteklega stoletja vodil skupino Group Networks Laboratory na univerzi MIT ([7, 8, 63]). Bavelas je bil med prvimi, ki so skušali odnose med ljudmi preslikati v omrežje. Njegovo zamisel o osrednji osebi v omrežju ponazorimo s primerom skupine žensk, ki so delale v veliki tovarni oblačil. Vse so govorile italijansko, le ena je znala tudi angleščino. Bavelas pravi: "It is difficult to imagine that the English speaking member would be other than central with respect to communication which had of necessity to pass through her ..." (Težko si je predstavljati, da ženska, ki govori angleško, ne bi bila osrednja za komunikacijo, ki nujno poteka prek nje.)

Mere pomembnosti uvedemo zato, da bi izmerili tisto, kar nam pravi občutek – da so nekatera vozlišča in / ali povezave v omrežju pomembnejše od drugih. Meram pomembnosti na neusmerjenih omrežjih pravimo mere središčnosti, na usmerjenih pa mere veljave. Omejili se bomo na pomembnosti vozlišč. Pomembnost vozlišča je skrita v zgradbi omrežja, ki mu vozlišče pripada. Vsako vozlišče v omrežju ima določen vpliv na vozlišča iz soseščine, ima nek ugled med svojimi sosedi, lahko si mislimo, da je bolj ali manj pomembno. Ni soglasja o tem, kaj natanko pomembnost je in kakšen je najboljši način za merjenje pomembnosti vozlišč [15]. Naloga tistih, ki analizirajo omrežje, je, da poiščejo in izberejo najprimernejšo metodo za dano omrežje in kontekst. Odločiti se morajo, kakšen je namen raziskave. Običajna vprašanja, ki si jih zastavljamo in ki jih rešujemo s pomočjo mer pomembnosti, so: Katere so vplivne osebe v tem omrežju? Katere ceste najpogosteje uporabljajo uporabniki cestnega omrežja? Katere spletne strani so pomembne?

Mere pomembnosti uporabljamo v politiki, pri transportu, pri organizaciji družbenih skupin in v komunikacijskih omrežjih. Odgovori na vprašanja o pomembnih vozliščih so uporabni v zdravstvu (koga in koliko ljudi je treba cepiti,

kdo in kje bo razdeljeval gradivo z informacijami), v kriminalistiki (koga je treba diskreditirati ali aretirati, da bo kriminalno omrežje propadlo, komu je treba povedati napačne ali zavajajoče informacije), v upravi (v katerih oddelkih je organizacija najbolj ranljiva, katerim zaposlenim je dobro povedati za načrtovane spremembe v podjetju) in drugod.

Raziskave o tem, kdo so ključni igralci v danem omrežju, delimo tudi glede na to, ali nas zanima posameznik (eno vozlišče) ali skupina (več vozlišč). Tudi to je odvisno od namena. Prvi pristop pri iskanju najpomembnejše skupine je, da izračunamo pomembnost za vsa vozlišča in izberemo k vozlišč, ki imajo največje vrednosti. Vendar ni nujno res, da je optimalna množica vozlišč tista, ki je sestavljena iz vozlišč, ki so samostojno optimalna. Obstajajo posebne mere pomembnosti za skupine. Več o teh merah v [22].

Vse mere pomembnosti vozlišč so funkcije, ki vozliščem omrežja pripišejo realno vrednost, $c : \mathcal{V} \to \mathbb{R}$. Razvili so jih tako, da vozlišča razvrstijo po pomembnosti glede na njihov položaj v omrežju. Kvantitativno merijo pomembnost, ki jo razumemo na različne načine glede na kontekst, ki nas zanima. Rečemo, da je vozlišče v manj pomembno od vozlišča u, če velja $c(u) > c(v)$. Za večino definicij pomembnosti je mogoče pokazati, da je vozlišče z največjo možno vrednostjo osrednje vozlišče (usmerjene) zvezde.

Ker ni vsaka mera pomembnosti primerna za vsako omrežje, so sčasoma uvedli še druge, posplošene mere, ki izhajajo iz narave problema, ki ga obstoječe mere niso opisale dovolj dobro ali dovolj učinkovito. Marsikatero mero pomembnosti so razvili le za določen problem in zato obstaja precej literature, ki obravnava vsaj eno mero pomembnosti ([19, 22, 46, 47, 80]). Opisali jih bomo le nekaj.

Stopnja vozlišča je verjetno najstarejša mera za pomembnost vozlišča v omrežju. Nekatere mere temeljijo na soseščini vozlišča, druge temeljijo na najkrajših poteh med vozlišči ali na naključnih procesih v omrežju. Spektralne mere pomembnosti temeljijo na (levem) dominantnem lastnem vektorju vrednostne matrike omrežja ali druge matrike, ki jo dobimo iz nje. Obstoj in enoličnost spektralnih mer sledita iz teorije nenegativnih matrik (izrek 1.1, stran 5). Utemeljitev uporabe lastnega vektorja kot mere pomembnosti je, da ni dovolj prešteti število sosedov vozlišča, upoštevati je treba tudi njihovo pomembnost. Iz istega razloga so razvili tudi druge rekurzivne mere pomembnosti, ki v računu upoštevajo pomembnost več vozlišč v omrežju.

Za večino mer pomembnosti nepovezanost (le šibka povezanost) omrežja predstavlja težavo – pri najkrajših poteh se zgodi, da nekatera vozlišča niso dosegljiva iz drugih (tistih, ki so v drugi komponenti), kar vodi do neskončnih razdalj med vozlišči in do težav pri meri dostopnosti ali do ničelnega števila najkrajših poti pri vmesnosti.

Za predstavo o tem, kaj merijo mere pomembnosti, poglejmo primer iz [22]. Namen primera je pokazati, da ni enega pravega načina, na katerega izračunamo

pomembnost, in da isto situacijo lahko smiselno analiziramo na različne načine glede na vprašanje, ki si ga zastavimo. Vsaka mera ima svojo intuitivno podlago, ki jo lahko razumemo kot prestiž, vpliv, nadzor, priljubljenost, pogostost, veljavo, ugled itd. Vozlišče je lahko potrebno za transport ali za prenos informacij po omrežju. Lahko je pomembno, ker je povezano s pomembnimi vozlišči. Pomembnost je torej močno povezana s kontekstom [20].

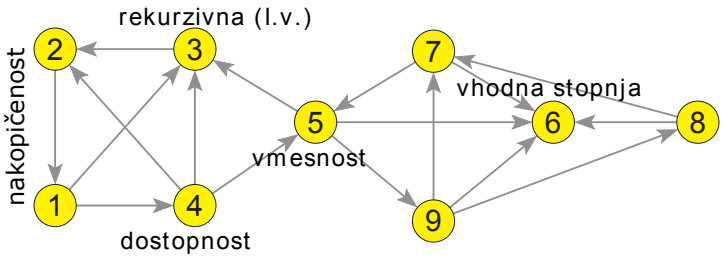
V razredu je 30 učencev, ki volijo razrednega predstavnika. Vsak učenec lahko voli enega od ostalih. Prvo vprašanje je, kdo zmaga na volitvah. Ustvarimo volitveno omrežje – vozlišča predstavljajo učence in povezava od učenca A do učenca B obstaja, če je učenec A volil učenca B. V tej situaciji lahko rečemo, da je učenec pomemben, če je zanj glasovalo veliko drugih učencev. Število glasov predstavlja vhodna stopnja vozlišča. Utemeljili smo uporabo prve mere pomembnosti, vhodne stopnje.

Drugačen pogled na isto situacijo da drugačno omrežje: Povezava med A in B predstavlja, da je učenec A prepričal učenca B, naj glasuje za njenega / njegovega najljubšega kandidata. Takemu omrežju pravimo omrežje vpliva. Predpostavimo, da je razred v grobem razdeljen v dve veliki skupini X in Y in da ima neka oseba odnose s člani obeh skupin. Če ima ta oseba najljubšega kandidata iz skupine X in prepriča precejšen del skupine Y, da glasuje za tega kandidata, je ta oseba pomembna, saj je bila posrednik večine informacij med obema skupinama. Na osnovi tega argumenta lahko rečemo, da je vozlišče pomembno, če je potrebno pri prenosu informacij ali mnenj med ostalimi udeleženci v omrežju. Glavni predstavnik mer pomembnosti, ki sledijo temu vidiku, je vmesnost.

Poglejmo še tretjo različico iste situacije. Zgradimo omrežje učencev v razredu, v katerem povezave nakazujejo, kdo je prijatelj s kom. Nekdo, ki je prijatelj pomembne osebe, je lahko vplivnejši od nekoga, ki ima prijatelje z nizkim socialnim statusom. Pomembnost te vrste je povezana s pomembnostjo sosednih vozlišč in ji pravimo rekurzivna mera pomembnosti.

Že v tem preprostem primeru vidimo, da nobena od metod ni boljša ali slabša od druge. Vsaka je dobra za iskanje odgovorov na nekatera vprašanja in je manj primerna za druga.

V nadaljevanju opišemo pet vrst pomembnosti: stopnjo, nakopičenost, dostopnost, vmesnost in rekurzivne mere. Na sliki 3.1 je poleg vozlišč omrežja napisano, za katero mero pomembnosti je vozlišče izbrano kot najpomembnejše. Vozlišče 6 je najpomembnejše glede na vhodno stopnjo, vozlišče 5 je najpomembnejše glede na vmesnost, vozlišče 4 glede na dostopnost in vozlišče 3 glede na rekurzivno mero pomembnosti na osnovi lastnih vektorjev. Največjo nakopičenost imata vozlišči 1 in 2.



Slika 3.1: Omrežje in najpomembnejša vozlišča glede na različne mere pomembnosti.

3.1 Stopnja vozlišča

Volitve so pogost dogodek v družbenih skupinah in vemo, katere osebe so pri tem dogodku pomembnejše od drugih. Na volitvah zmaga tisti, ki dobi največ glasov. Število glasov ustreza stopnji vozlišča v omrežju, v katerem obstoj povezave (u, v) pomeni, da je oseba, ki jo predstavlja vozlišče u, volila osebo v.

Stopnja $\deg(v)$ je verjetno prva, najbolj enostavna in najbolj očitna izbira za merjenje pomembnosti. V usmerjenih omrežjih definiramo dve meri veljave glede na stopnjo – izhodno $\mathrm{outdeg}(v)$ in vhodno $\mathrm{indeg}(v)$.

Mera pomembnosti glede na stopnjo je lokalna, saj je pomembnost vozlišča odvisna le od njegovih sosedov. Meri vidnost ali potencial za komunikacijo z ostalimi vozlišči. Razlagamo si jo kot število priložnosti vozlišča, da vpliva na druge ali da nanj vplivajo drugi. Uporabljamo jo, če nas zanimajo rezultati volitev, širjenje informacij, govoric, virusov, odpornosti, moč in voditeljske sposobnosti posameznika, zadovoljstvo, dostopnost znanja, izpostavljenost itd.

Stopnje vozlišč lahko izračunamo s pomočjo sosednostne matrike omrežja

$$\mathrm{outdeg}(v) = (\mathbf{Ae})_v, \quad \mathrm{indeg}(v) = (\mathbf{e}^T\mathbf{A})_v,$$

kjer je $\mathbf{e}$ vektor samih enic.

Pomembnosti glede na stopnjo vozlišča pogosto normaliziramo tako, da rezultat delimo z maksimalno možno stopnjo v omrežju, $n - 1$. Normalizirana pomembnost $\frac{\deg(v)}{n-1}$ pomeni delež možnih povezav, ki jih je vozlišče realiziralo. Normalizacijo uporabljamo, da laže primerjamo podatke v različnih ali ne (krepko) povezanih omrežjih.

3.2 Nakopičenost

Tudi nakopičenost je lokalna mera pomembnosti, ki je odvisna le od sosedov vozlišča.

Definicija 3.1. Naj bo $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ enostavno omrežje brez zank. V omrežju $\mathcal{N}$ vse morebitne neusmerjene povezave zamenjamo s parom nasprotno usmerjenih povezav. *Nakopičenost* $C(v)$ vozlišča v v omrežju $\mathcal{N}$ definiramo kot delež obstoječih povezav med sosedi vozlišča v v številu vseh možnih povezav med sosedi vozlišča v. Torej

$$C(v) = \frac{|\mathcal{L}(N(v))|}{\deg(v)(\deg(v)-1)},$$

kjer je $\mathcal{L}(N(v))$ množica povezav med sosedi vozlišča v. Za vozlišče v, ki nima sosedov ali ima samo enega soseda, definiramo $C(v) = 0$.

Nakopičenost meri lokalno gostoto soseščine vozlišča. Težave pri uporabi nakopičenosti kot mere pomembnosti so v tem, da običajno kot najgostejše vrne soseščine vozlišč z zelo majhno stopnjo. Zato ta mera ni uporabna v analizi podatkov. V programu Pajek uporabljamo *popravljeno nakopičenost* $C'(v)$, ki je definirana s

$$C'(v) = \frac{|\mathcal{L}(N(v))|}{\Delta(\deg(v)-1)}.$$

Popravljena nakopičenost $C'(v)$ doseže maksimalno vrednost 1 le na izoliranih klikah stopnje Δ.

Pri štetju uspelih povezav med sosedi vozlišča uporabimo dejstvo, da vsaka povezava iz $\mathcal{L}(N(v))$ določa trikotnik z oglišči v krajiščih povezave in v vozlišču v. Število trikotnikov v enostavnem neusmerjenem grafu je enako diagonalnim vrednostim v tretji potenci sosednostne matrike omrežja (nad kombinatoričnim polkolobarjem).

Za enostavne usmerjene grafe je štetje trikotnikov malo težje. Označimo $\mathbf{T} = \mathbf{A}^T$ in $\mathbf{S} = \mathbf{A}+\mathbf{T}$. S slike 3.2 vidimo, da se vsak trikotnik (ki je določen s povezavo, ki leži nasproti obarvanega vozlišča) pojavi natanko enkrat v izrazu

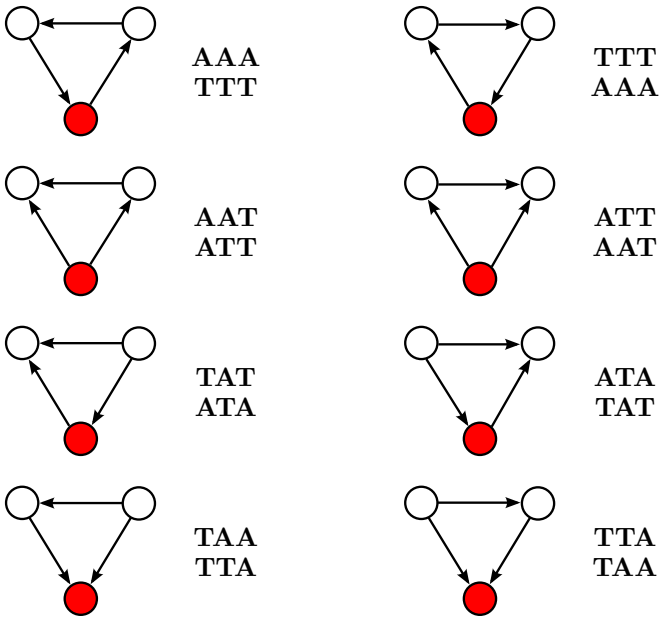
$$\mathbf{AAA} + \mathbf{AAT} + \mathbf{TAT} + \mathbf{TAA} = \mathbf{AAS} + \mathbf{TAS} = \mathbf{SAS}.$$

Diagonalni elementi matrike $\mathbf{SAS}$ nam povejo število uspelih povezav med sosedi ustreznega vozlišča.

3.3 Dostopnost

Dostopnost je eden od tradicionalnih indeksov, ki se uporabljajo v analizi družbenih omrežij [46]. Če omrežje ni krepko povezano, lahko pri izračunu pride do težav, ki smo jih omenili v uvodu tega poglavja.

Definicija 3.2. Naj bo d_{uv} dolžina (vrednost) najkrajše poti od vozlišča u do vozlišča v v omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$. *Izhodna dostopnost* vozlišča v je definirana



Slika 3.2: Štetje trikotnikov.

kot

$$ocl(v) = \frac{n-1}{\sum_{u \in \mathcal{V} \setminus \{v\}} d_{vu}},$$

vhodna dostopnost vozlišča v pa podobno

$$icl(v) = \frac{n-1}{\sum_{u \in \mathcal{V} \setminus \{v\}} d_{uv}}.$$

Dostopnost meri inverz vsote najkrajših razdalj do vseh ostalih vozlišč. Ker želimo upoštevati tudi velikost omrežja, izraz standardiziramo tako, da ga množimo z $n-1$. Če so uteži različne od 1, izraz množimo še z dolžino najkrajše poti v omrežju. Vsota v imenovalcu je ustrezna vrstična / stolpična vsota v matriki geodezičnih razdalj $\mathbf{D} = [d_{uv}]_{u,v \in \mathcal{V}}$. Matrika $\mathbf{D}$ je zaprtje vrednostne matrike $\mathbf{A}$ omrežja $\mathcal{N}$ nad polkolobarjem najkrajših poti $(\overline{\mathbb{R}_0^+}, \min, +, \infty, 0)$. Opozorimo, da morajo biti vrednosti v vrednostni matriki nenegativne. Vrednost standardizirane dostopnosti je med 0 in 1.

Mera pomembnosti glede na dostopnost izhaja iz problema nadzora omrežja. Kako priti do dobrih informacij o tem, kaj se dogaja v omrežju, tako da nadzorujemo le ključne akterje? Približen odgovor na to vprašanje nam da vhodna dostopnost.

Še drugačen primer uporabe: Želimo postaviti trgovino na tako mesto, da bo skupna razdalja do kupcev v regiji najmanjša. Tako bo pot do trgovine čim bolj udobna za vse potencialne stranke. Iskanje primerne lokacije lahko rešimo tako, da izračunamo dostopnost v cestnem omrežju območja.

Dostopnost si lahko razlagamo tudi kot pričakovani čas, ob katerem bo informacija pripotovala do izbranega vozlišča (v primeru širjenja govoric bo najpomembnejše vozlišče glede na dostopnost prvo slišalo govorico).

3.4 Vmesnost

Tudi vmesnost je eden od klasičnih indeksov za analizo družbenih omrežij ([45, 46]).

Definicija 3.3. *Vmesnost vozlišča* v v omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ je definirana kot

$$b(v) = \frac{1}{(n-1)(n-2)} \sum_{\substack{u,w \in \mathcal{V} \\ |\{v,u,w\}|=3}} \frac{n_{uw}(v)}{n_{uw}},$$

kjer je n_{uw} število najkrajših poti od u do w in $n_{uw}(v)$ število najkrajših poti od u do w, ki potekajo skozi vozlišče v. Če je $n_{uw} = 0$, je tudi $n_{uw}(v)/n_{uw} = 0$.

Vmesnost je najpogosteje uporabljana mera pomembnosti, ki temelji na najkrajših poteh v omrežju. Razmerje $\frac{n_{uw}(v)}{n_{uw}}$ lahko razumemo kot verjetnost, da komunikacija med u in w poteka skozi vozlišče v. To pomeni, da izračunana pomembnost implicitno predpostavlja, da vsa komunikacija med vozliščema poteka po najkrajši poti. Če ni tako, mnogokrat dobimo napačne rezultate. Nekaj alternativnih možnosti so pomembnosti, ki jih definira Borgatti v [19].

Če poznamo matriko

$$\mathbf{C} = [(d_{uv}, n_{uv})]_{u,v \in \mathcal{V}},$$

kjer je d_{uv} dolžina najkrajše poti od u do v in n_{uv} število najkrajših poti od u do v, lahko enostavno določimo vrednost $n_{uw}(v)$:

$$n_{uw}(v) = \begin{cases} n_{uv} \cdot n_{vw}, & d_{uv} + d_{vw} = d_{uw}, \\ 0, & \text{sicer.} \end{cases}$$

Matriko $\mathbf{C}$ izračunamo kot zaprtje prirejene vrednostne matrike $\mathbf{G}$ omrežja nad geodezičnim polkolobarjem $\left(\overline{\mathbb{R}_0^+} \times \overline{\mathbb{N}}, \oplus, \odot, (\infty, 0), (0, 1)\right)$, ki smo ga opisali v razdelku 2.2.4 na strani 16. Vrednostno matriko omrežja priredimo v matriko $\mathbf{G}$, katere elementi so pari [4]

$$g_{uv} = \begin{cases} (1, 1), & (u, v) \in \mathcal{L}, \\ (\infty, 0), & \text{sicer.} \end{cases}$$

Mero pomembnosti glede na vmesnost lahko izračunamo po postopku, ki je zapisan v algoritmu 2. Trenutno najučinkovitejši postopek za izračun vmesnosti je predlagal Brandes v [21].

Algoritem 2 Računanje vmesnosti vozlišč.

1: sestavi matriko $\mathbf{G}$
2: izračunaj matriko $\mathbf{C}$ kot zaprtje matrike $\mathbf{G}$ nad geodezičnim polkolobarjem
3: **for** $v \in \mathcal{V}$ **do**
4: $\quad r \leftarrow 0$
5: $\quad$ **for** $u \in \mathcal{V}, w \in \mathcal{V}$ **do**
6: $\quad\quad$ **if** $n[u,w] \neq 0 \wedge |\{v,u,w\}| = 3 \wedge d[u,w] = d[u,v] + d[v,w]$ **then**
7: $\quad\quad\quad r \leftarrow r + n[u,v] \cdot n[v,w]/n[u,w]$
8: $\quad b[v] \leftarrow r/((n-1)\cdot(n-2))$

Vmesnost so uvedli v [45] kot odgovor na težave mere pomembnosti glede na dostopnost v omrežjih, ki niso (krepko) povezana. V tem primeru je razdalja med dvema vozliščema, ki ne ležita v isti (krepko) povezani komponenti, neskončna. Dostopnost nam torej ne da nobene informacije o omrežju, saj so vse vrednosti enake $\frac{n-1}{\infty}$.

Zamisel, na kateri temelji mera pomembnosti glede na vmesnost, je problem motenja delovanja omrežja. Kje moramo ukrepati, da bomo povzročili največ motenj? Izračun vmesnosti za omrežje kriminalne dejavnosti bi povedal, katere akterje je treba poslati v zapor, da bo omrežje, ki ostane po odstranitvi, čim bolj ohromljeno. Lahko si predstavljamo tudi, da vmesnost meri, koliko nadzora ima določeno vozlišče nad prenosom informacij, ki so dostopne v omrežju. Predstavlja indeks potenciala za varovanje, združevanje, pogajanje. Pove, kakšen nadzor ima vozlišče nad tokom v omrežju in kako pomembno je za povezanost ostalih delov omrežja. Meri strateški položaj vozlišča.

3.5 Rekurzivne mere pomembnosti

Mere tega razdelka pri izračunu pomembnosti vozlišča uporabijo tudi pomembnosti okoliških vozlišč. Nekatere izmed njih so ene najstarejših mer pomembnosti (Katz), druge izhajajo iz analize družbenih omrežij (Bonacich). Zadnja skupina pripada velikemu razredu metod za analizo omrežja spletnih strani in hiperpovezav (HITS, pageRank).

Verjetno najbolj jasna oblika pomembnosti ostaja mera glede na stopnjo vozlišča. Ker marsikdaj število povezav ne odraža dejanskega vpliva vozlišča v omrežju, Wasserman in Faust v [80] govorita o merah pomembnosti, pri katerih na pomembnost vozlišča rekurzivno vplivajo tudi pomembnosti sosednih vozlišč.

Zamisel, ki stoji za tem, je: "Bolje je poznati nekaj vplivnih ljudi, kot veliko ljudi brez vpliva." Vse rekurzivne mere imajo podobno obliko.

Definicija 3.4. Naj bo $\mathbf{A}$ vrednostna matrika omrežja $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ in naj bo x vektor vhodnih mer pomembnosti. *Vhodna rekurzivna mera pomembnosti* vozlišča v ima obliko

$$x_v = a_{1v}x_1 + a_{2v}x_2 + \cdots + a_{nv}x_n = \sum_{u:(u,v)\in\mathcal{L}} a_{uv}x_u.$$

Podobno definiramo *izhodno rekurzivno mero pomembnosti*.

Vhodna pomembnost vozlišča v je linearna kombinacija vhodnih pomembnosti vhodnih sosedov vozlišča v. V matrični obliki lahko množico zgornjih enačb zapišemo kot

$$\mathbf{A}^T x = x. \tag{3.1}$$

3.5.1 Pomembnost na osnovi lastnih vektorjev

V enačbi (3.1) je x lastni vektor matrike $\mathbf{A}^T$, ki pripada lastni vrednosti 1. Ta nima neničelnih rešitev, če 1 ni lastna vrednost matrike $\mathbf{A}^T$. Ena rešitev je normalizacija vrstic, tako da je vsota elementov v vsaki vrstici enaka 1. Potem ima normalizirana matrika lastno vrednost 1 in obstaja rešitev enačbe (3.1).

Vhodna lastna pomembnost, ki jo je predstavil Bonacich v [16], posploši enačbo (3.1) na običajno enačbo za lastne vrednosti matrike $\mathbf{A} \in \mathbb{R}^{n\times n}$. Pri tem predpostavi, da je vhodna pomembnost vozlišča sorazmerna z uteženo vsoto vhodnih pomembnosti sosedov.

Definicija 3.5. *Vhodna lastna pomembnost* vozlišča v v omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ je definirana kot

$$\lambda x_v = a_{1v}x_1 + a_{2v}x_2 + \cdots + a_{nv}x_n = \sum_{u:(u,v)\in\mathcal{L}} a_{uv}x_u.$$

Strnjeno jo zapišemo v matrični obliki

$$\mathbf{A}^T x = \lambda x. \tag{3.2}$$

Ta enačba ima vedno neničelno rešitev x.

V nekaterih primerih je bolje definirati izhodno pomembnost vozlišča v kot kombinacijo izhodnih pomembnosti izhodnih sosedov. Po enakem premisleku definiramo izhodno lastno pomembnost.

Definicija 3.6. *Izhodne lastne pomembnosti* so komponente vektorja x, ki je rešitev enačbe

$$\mathbf{A}x = \lambda x. \tag{3.3}$$

V obeh primerih iščemo dominanten lastni par (transponirane) vrednostne matrike omrežja $\mathbf{A}$.

Najenostavnejša metoda za izračun dominantnega lastnega para je potenčna metoda (poglavje 4, stran 35). Spomnimo se, da po Perron-Frobeniusovem izreku (izrek 1.1, stran 5) za krepko povezana omrežja obstaja natanko ena dominantna lastna vrednost vrednostne matrike omrežja.

Če omrežje ni krepko povezano, ima vrednostna matrika (z morebitnim preoštevilčenjem vozlišč) bločno obliko

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}.$$

V tem primeru dominantni lastni vektor ni nujno enoličen in so mnenja o tem, kako razlagamo rezultate, deljena. V večini primerov ima desni lastni vektor obliko $[\tilde{x}, 0]^T$, kar pomeni, da nimamo informacije o pomembnosti za precejšen del omrežja.

Kadar omrežje ni povezano, ima (morda preoštevilčena) vrednostna matrika bločno diagonalno obliko. Naj bo

$$\mathbf{A} = \begin{bmatrix} 0.8000 & 0.7500 & 0 & 0 \\ 0.2000 & 0.2500 & 0 & 0 \\ 0 & 0 & 0.4000 & 0.5455 \\ 0 & 0 & 0.6000 & 0.4545 \end{bmatrix}$$

vrednostna matrika nepovezanega omrežja. Njene lastne vrednosti so $\lambda_{1,2} = 1, \lambda_3 = -0.1455$ in $\lambda_4 = 0.0500$. Dominantna lastna vektorja, ki pripadata lastnima vrednostima 1, sta vektorja $v_1 = [0.9662, 0.2577, 0, 0]$ in $v_2 = [0, 0, -0.6727, -0.7399]$. Ker pripadata isti lastni vrednosti, je tudi njuna vsota $v_1 + v_2 = [0.9662, 0.2577, -0.6727, -0.7399]$ ali katera koli linearna kombinacija vektorjev v_1 in v_2 tudi lastni vektor. Kako izberemo pravega? Odgovora na to vprašanje ni. Lastna vektorja v_1 in v_2 ustrezata pomembnostim vozlišč v vsaki od komponent, kar je smiselno. Dobrega načina za primerjavo vrednosti pa nimamo.

V primeru nepovezanih omrežij se pojavijo še dodatne težave. Vrednosti, ki jih dobijo vozlišča največje komponente, niso nujno neničelne. Pogosto so najvišje ovrednotena vozlišča, ki pripadajo majhnim krepko povezanim komponentam, velikokrat komponentam, ki vsebujejo le dve vozlišči. Običajno nas zanima največja krepko povezana komponenta omrežja in ne majhne komponente, vendar vozlišča v največji komponenti najverjetneje ne dobijo višjih vrednosti od vozlišč v majhnih komponentah.

Če omrežje ni krepko povezano, se lahko uporabnik odloči, ali bo omrežje razbil na krepko povezane komponente in izračunal lastno pomembnost posebej za vsako komponento. Druga možnost je, da uporabi katero od drugih rekurzivnih pomembnosti, o katerih govorimo v nadaljevanju, ki nimajo težav z nepovezanostjo.

3.5.2 Katzova pomembnost

Motivacija za uporabo Katzove mere pomembnosti je preiskovanje vpliva na omrežje: kako čim bolj učinkovito širiti zamisli, (napačne) informacije, material, reklame, bolezni itd.

V svojem članku [58] Katz opiše mero pomembnosti, pri kateri na pomembnost vozlišča v vplivajo vsa vozlišča, iz katerih je vozlišče v dosegljivo. V pristopu uporabi utež α, s katero omili vpliv bolj oddaljenih vozlišč. Utež α je lahko odvisna od vsebine in akterjev omrežja. Konstanto α vidimo kot verjetnost uspeha povezave: vrednost $\alpha = 0$ pomeni, da tudi sosedi nimajo nobenega vpliva na vozlišče, vrednost $\alpha = 1$ pa, da oddaljena vozlišča enako močno vplivajo na vozlišče kot njegovi sosedi.

Matematičen opis zamisli dobimo s potencami sosednostne matrike $\mathbf{A}$ omrežja. Stolpične vsote matrike $\mathbf{A}^p$ predstavljajo število vseh sprehodov dolžine p iz drugih vozlišč. Iščemo stolpične vsote matrike

$$\mathbf{T} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \cdots + \alpha^k\mathbf{A}^k + \cdots = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I},$$

kar je enakovredno [58] reševanju sistema linearnih enačb

$$\left(\frac{1}{\alpha}\mathbf{I} - \mathbf{A}^T\right) t = d, \tag{3.4}$$

kjer je d vektor vhodnih stopenj. To velja, če je $\|\alpha\mathbf{A}\| < 1$. Vektor t ima elemente t_v, ki so enaki vsoti ustreznega stolpca matrike $\mathbf{T}$.

Dobre izbire za $1/\alpha$ so vrednosti med dominantno lastno vrednostjo matrike $\mathbf{A}$ in podvojeno dominantno lastno vrednostjo. Za manjše vrednosti $1/\alpha$ je učinek oddaljenih vozlišč večji.

Običajno so mere pomembnosti normalizirane glede na število vseh možnih izidov. Uporabimo isto zamisel in delitelj elementa t_v je Katz definiral z

$$m = \alpha(n-1) + \alpha^2(n-1)^{(2)} + \alpha^3(n-1)^{(3)} + \ldots,$$

kjer smo označili $(n-1)^{(k)} = (n-1)(n-2)\cdots(n-k)$. Dobra aproksimacija za m je

$$m \doteq (n-1)!\alpha^{n-1}e^{1/\alpha}, \tag{3.5}$$

ki se izboljšuje z $n \to \infty$.

Definicija 3.7. Normaliziran *Katzov vektor pomembnosti* je $\frac{1}{m}t$, kjer je t rešitev sistema (3.4) in m kot v enačbi (3.5).

Za reševanje linearnih sistemov obstaja precej metod, nekaj jih opišemo v nadaljevanju (poglavje 4, stran 35).

Postopek deluje tudi za vrednostne matrike omrežja. V tem primeru potence sosednostne matrike predstavljajo vrednosti poti in razlaga ni tako jasna.

3.5.3 Bonacichevi pomembnosti α in (α, β)

Dominantni lastni vektor iz poglavja 3.5.1 je ena od standardnih mer pomembnosti v omrežjih. Kot smo že omenili, pa pogosto pride do težav. Ena izmed težav je tudi ta, da imajo vozlišča z ničelno izhodno stopnjo tudi ničelno pomembnost. Zato imajo tudi vozlišča, ki kažejo na vozlišča z ničelno pomembnostjo, ničelno pomembnost. Pojav se širi po omrežju in lahko se zgodi, da lastna pomembnost ne da informacije o velikem številu vozlišč v omrežju. Nekaj rešitev tega problema je opisanih v [17, 18, 71].

Vsakemu vozlišču v lahko pripišemo status s_v, ki ni odvisen od povezav v omrežju. Vektor s lahko odraža učinek zunanjega statusa, velikokrat pa ga definiramo kot vektor samih enic.

Definicija 3.8. Dana sta vektor s in parameter α. Naj bo $\mathbf{A}$ vrednostna matrika omrežja $\mathcal{N}$. *Bonacicheve pomembnosti* α so komponente vektorja x, ki je rešitev sistema

$$x = \alpha(\mathbf{A}^T x) + s. \tag{3.6}$$

Parameter α določa relativno pomembnost omrežnih virov glede na zunanje vire, ki jih opisuje vektor s.

V matrični obliki lahko rešitev sistema (3.6) zapišemo kot

$$x = (\mathbf{I} - \alpha\mathbf{A}^T)^{-1}s,$$

od koder hitro vidimo, da je Bonacicheva pomembnost α skoraj popolnoma enaka Katzovi meri pomembnosti iz razdelka 3.5.2.

Še eno rešitev je Bonacich predlagal v [17] in je prav tako zelo podobna Katzovi.

Definicija 3.9. *Bonacichevo pomembnost* (α, β) za vozlišče v v omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, w)$ definiramo z

$$c_v(\alpha, \beta) = \sum_{u \in \mathcal{V}} (\alpha + \beta c_u) a_{uv},$$

kar zapišemo v matrični obliki

$$c(\alpha, \beta) = \alpha(\mathbf{I} - \beta\mathbf{A})^{-1}\mathbf{A}\mathbf{e}, \tag{3.7}$$

kjer smo z $\mathbf{e}$ označili stolpec samih enic.

Parameter β pove, koliko pomembnosti vozlišča je prišlo iz njegove soseščine. Če je β pozitivna, se status vozlišča povečuje s številom povezav. To je res v primerih, ko opazujemo npr. komunikacijska omrežja, v katerih se količina posamezniku dostopnih informacij povečuje s številom informacij, ki so dostopne njegovim kontaktom. Kadar želimo, da se status, moč ali vpliv vozlišča povečajo s povezavami z drugimi vplivnimi vozlišči, izberemo pozitivno β.

V nekaterih situacijah je koristno poznati ljudi, ki nimajo veliko drugih možnosti (npr. pri kupčijah in barantanju). V tem primeru imamo prednost, če smo povezani z vozlišči, ki imajo majhno moč. Pomembnost vozlišča se zmanjšuje s povezavami z močnimi vozlišči in izberemo negativno β. Največja razlika med Bonacichevo (α, β) in Katzovo pomembnostjo je ta, da dovoljujemo negativno β. Pomembnost (α, β) je prva mera, ki opisuje negativno odvisnost od števila povezav, in ji pogosto rečemo kar mera pomembnosti za barantanje.

Velikost absolutne vrednosti parametra β pove vpliv oddaljenih vozlišč. Ko je $\beta = 0$, je pomembnost za barantanje sorazmerna stopnji. Ko povečujemo $|\beta|$, (dosegljiva) oddaljena vozlišča vedno bolj vplivajo na pomembnost vozlišča.

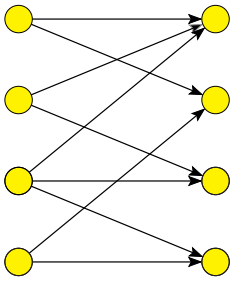
Iz enačbe (3.7) vidimo, da α vpliva samo na dolžino rešitve. Če α ne podamo, rezultat normaliziramo tako, da velja $\|c(\alpha, \beta)\|_2^2 = n$. Za normalizirane pomembnosti potem $c_v(\alpha, \beta) = 1$ pomeni, da vozlišče v nima posebnega položaja v omrežju.

3.5.4 Kazala in viri (HITS)

Meri pomembnosti za kazala in vire so uvedli za pomoč pri iskanju po spletu, vendar nista omejeni na uporabo v omrežjih, ki izhajajo iz tekstovnega iskanja na internetu. Posebej uporabni sta, kadar lahko na akterje omrežja gledamo na dva načina, na primer nasilneži ali žrtve, kupci ali prodajalci, ponudniki ali povpraševalci itd.

V času, ko so algoritem prvič predstavili, so se spletni iskalniki zanašali na indeksiranje spletnih strani in so ustvarjali strukturirano zbirko indeksiranih spletnih strani. Vendar tako iskanje ne razlikuje med stranmi, ki ustrezajo izbranemu vprašanju (ključni besedi, ki jo vnesemo v iskalnik), in stranmi, ki so splošno priljubljene. Iskali so način, kako zadetke ovrednotiti glede na položaj strani v spletnem omrežju. Ustvarjalec strani v je pripisal pomembnost strani u, ko je vključil povezavo do strani u na svojo stran. Kleinberg je v [59] definiral dve vlogi spletnih strani – kazala (*hubs*) in vire (*authorities*). Stran, na katero se sklicuje veliko

strani, je priljubljena. Takim stranem pravimo viri. Poleg takih strani obstajajo še strani, ki združujejo sezname pomembnih strani. Pravimo jim kazala. Dobro kazalo kaže na dobre vire in na dober vir kažejo dobra kazala (slika 3.3). Stran dobi vrednost vira iz vrednosti kazal strani, ki kažejo nanjo. Vrednost kazala dobi iz vrednosti virov strani, na katere kaže. Kleinberg je definiral rekurzivni pravili za posodabljanje vrednosti kazal in vrednosti virov, ki ju uporabimo iterativno. Za pregled glejte tudi [71].



Slika 3.3: Kazala (leva vozlišča) in viri (desna vozlišča) v omrežju.

Algoritem deluje na izbranem podomrežju interneta, ki ga dobimo iz tekstovnega iskalnika. S konstrukcijo podomrežja se ne bomo ukvarjali in bomo predpostavili, da je primerno podomrežje dano.

Definicija 3.10. Naj bo $\mathbf{A}$ sosednostna matrika izbranega podomrežja. Vsakemu vozlišču v, ki predstavlja spletno stran, pripišemo dve vrednosti: *vrednost kazala* x_v in *vrednost vira* y_v. Vrednosti povezujejo rekurzivna pravila

$$\lambda y_v = \sum_{u:(u,v)\in\mathcal{L}} x_u = \sum_{u\in\mathcal{V}} a_{uv} x_u = (\mathbf{A}^T x)_v,$$
$$\mu x_v = \sum_{u:(v,u)\in\mathcal{L}} y_u = \sum_{u\in\mathcal{V}} a_{vu} y_u = (\mathbf{A} y)_v,$$

kar zapišemo z matrikama

$$\lambda\mu x = \mathbf{A}\mathbf{A}^T x \quad \text{in} \quad \lambda\mu y = \mathbf{A}^T\mathbf{A} y.$$

To pomeni, da so vrednosti kazal x in vrednosti virov y elementi dominantnih lastnih vektorjev matrik $\mathbf{A}\mathbf{A}^T$ in $\mathbf{A}^T\mathbf{A}$. Če poznamo uporabnika, lahko iskanje prilagodimo in izračun posplošimo z vrednostno matriko.

3.5.5 PageRank

PageRank je mera pomembnosti, ki jo Google uporablja za razvrstitev spletnih strani. Zaradi Googleovega uspeha je precej literature, ki obravnava razne različice algoritma, na primer [12, 23, 25, 54, 62, 71]. Brin in Page sta prvič opisala računanje pageRanka v njunem izvirnem članku [23].

Na postopek pageRank lahko gledamo na dva bistveno različna načina – lahko ga opišemo kot oceno, pridobljeno z naključnim sprehodom v omrežju, ali kot lastni vektor matrike, ki jo dobimo iz sosednostne matrike omrežja. Na kratko bomo opisali oba načina. Zaradi velikosti interneta je v praksi uporabljana različica naključnega sprehoda, o kateri kasneje ne bomo več govorili.

Naključni sprehod je stacionaren proces na vsakem neusmerjenem grafu. Pomembnost vozlišča, ki izhaja iz naključnega sprehoda, definiramo kot število obiskov vozlišča, ki jih sprehajalec opravi v naključnem sprehodu. V usmerjenih grafih naključni sprehod ni nujno stacionaren proces, saj so vozlišča z izhodno stopnjo enako 0 (ponori) privlačujoča stanja. Ko enkrat pridemo v vozlišče z ničelno izhodno stopnjo, ne moremo oditi iz njega. Da bo proces stacionaren, sprehajalcu omogočimo, da odide iz ponora.

Naključni sprehajalec pageRank simulira obnašanje povprečnega uporabnika interneta. Večino časa uporabnik klika povezave na straneh (brska), včasih pa vpiše naslov spletne strani (skoči). Osnovnemu modelu naključnega sprehoda dodamo skoke, ki se zgodijo z verjetnostjo q in odpeljejo simuliranega uporabnika na naključno stran. Ta proces lahko enostavno opišemo s sistemom enačb

$$p_v = \frac{q}{n} + (1-q) \sum_{u:(u,v)\in\mathcal{L}} \frac{p_u}{\mathrm{outdeg}(u)}, \quad v = 1, 2, \ldots, n, \tag{3.8}$$

kjer je p_v vrednost pageRank vozlišča v. Vsota teče po vseh vhodnih sosedih vozlišča v.

Tipično izberemo verjetnost skoka $q = 0.1$. Majhne vrednosti q bolje ohranijo informacije o povezavah omrežja. Če je $q = 0$, proces ni nujno stacionaren in pageRank ni dobro definiran. Ko je $q = 1$, skoki prevladajo in vsa vozlišča imajo enako vrednost pageRank, tj. $\frac{1}{n}$.

Za zapis enakovredne matrične različice izračuna pageRank vrednosti vozlišč označimo sosednostno matriko omrežja z $\mathbf{A}$ in diagonalno matriko izhodnih stopenj z $\mathbf{D}$. Potem ima normalizirana matrika $\mathbf{S} = \mathbf{D}^{-1}\mathbf{A}$ vrstične vsote enake 1. Kadar stran v nima izhodnih povezav, je vrstična vsota, ki ustreza v v matriki $\mathbf{A}$, enaka 0 in ne moremo izračunati ustrezne vrstice v matriki $\mathbf{S}$. V tem primeru definiramo $\mathbf{S}_{vu} = \frac{1}{n}$ za vse $u \in \mathcal{V}$. Zdaj izračunamo matriko $\mathbf{M}$ kot

$$\mathbf{M} = \frac{q}{n}\,\mathbf{1} + (1-q)\,\mathbf{S}, \tag{3.9}$$

kjer je $\mathbf{1}$ matrika samih enic. Matrika $\mathbf{M}$ je pozitivna in ima enoličen normiran pozitiven levi lastni vektor x, tako da je $x^T\mathbf{M} = x^T$. Vrednost pageRank vozlišča v je vrednost x_v.

3.6 Kaj če omrežje ni (krepko) povezano?

Večina predstavljenih mer pomembnosti predvideva, da je omrežje krepko povezano, sicer lahko pride do težav. Za lokalne indekse, kakršna sta stopnja in nakopičenost, predpostavka o povezanosti nima posledic. V splošnem pa ni tako.

Kot smo že omenili, imajo mere pomembnosti, ki temeljijo na najkrajših poteh, težave, ker v ne (krepko) povezanih omrežjih obstajajo pari vozlišč, za katere je dolžina najkrajše poti enaka ∞.

Zelo naiven pristop je ta, da omejimo izračun pomembnosti na podomrežja, v katerih je mera dobro definirana. Npr. izračunamo pomembnost vsakega vozlišča glede na krepko povezano komponento, ki ji pripada. V večini primerov uporabe ta pristop ni najbolj smiseln.

V omrežju iz dveh krepkih komponent, od katerih je ena poln graf na dveh vozliščih, druga poln graf na $n-2$ vozliščih in n velik, bo tak pristop vrnil vrednost dostopnosti enako 1 za vsa vozlišča, čeprav se zdi očitno, da so vozlišča iz velike krepke komponente precej pomembnejša od ostalih dveh vozlišč.

Običajno to težavo odpravimo tako, da izračunane vrednosti mere pomembnosti pomnožimo z velikostjo komponente, kar ustreza intuiciji, da so vozlišča velike komponente bolj pomembna. To se zdi smiselno, a tudi ni popolnoma zadovoljivo, če se mera pomembnosti ne obnaša sorazmerno z velikostjo omrežja. In v primeru dostopnosti se ne.

Dodatna dva načina, na katera lahko popravimo pomembnosti, uporabljata inverzne dolžine poti in dodatne fiksne vrednosti za razdalje med nepovezanimi vozlišči. Na koncu dobimo mero, ki nič kaj ne spominja na izvirno in velikokrat nima veliko skupnega z njo.

Izkaže se, da je še vedno najboljša izbira, da za resnično težavne primere izberemo pomembnost, ki s povezanostjo nima težav.

Poglavje 4

Numerični postopki

V tem poglavju bomo iz [36] povzeli najenostavnejše numerične metode za računanje dominantnega lastnega para matrike in za iterativno reševanje sistemov linearnih enačb. V našem programu za analizo časovnih omrežij zaenkrat uporabljamo matrično predstavitev omrežja (poglavje 7, stran 59), zato bomo opisali le metode, ki so jih razvili za polne matrike. Vse opisane metode delujejo bolj učinkovito, če izberemo redko predstavitev matrike.

Ker se računanje lastnih parov in reševanje sistemov linearnih enačb zelo pogosto uporabljata, ko je matrika zelo velika in redka, so bili razviti precej učinkovitejši iterativni algoritmi za take primere. Konvergenca teh postopkov je odvisna tudi od elementov matrike in od tega, ali ima matrika posebno obliko. V veliki večini iterativne metode za redke matrike temeljijo na podprostorih Krilova (Arnoldijev in Lanczosev algoritem za generiranje baze podprostora Krilova, v katerem iščemo približek za rešitev), obstaja pa tudi posplošitev Jacobi-Davidsona, ki išče rešitve na drugačen način.

Na tem mestu bomo bralca napotili drugam, če ga zanima iterativno računanje lastnih parov [2] ali reševanje linearnih sistemov [75] na velikih redkih matrikah. V [40, 75] so zapisani tudi modeli redke predstavitve matrik in implementacije množenja z njimi. V primeru redkih matrik je večinoma potrebno tudi predpogojevanje [41, 75], to je dodaten korak v iterativnem algoritmu, v katerem spremenimo matriko tako, da je konvergenca hitrejša. Za nadaljnje delo s časovnimi omrežji so zanimivi tudi lastni problemi parametriziranih matrik $\mathbf{A}(p)$, ki imajo za elemente funkcije parametra p. Numerične metode za iskanje lastnih vrednosti parametriziranih matrik so opisane v [26, 33, 37].

4.1 Potenčna metoda

Najenostavnejša metoda za izračun dominantnega lastnega para matrike je *potenčna metoda*. Postopek za izračun dominantnega lastnega para matrike je zapisan v algoritmu 3.

Algoritem 3 Potenčna metoda za izračun dominantnega lastnega para.

```
1: function power(A, x_0, k = 100)        ▷ x_0 je začetni približek za lastni vektor
2:     for i = 0 to k do
3:         y_{i+1} ← A x_i
4:         x_{i+1} ← y_{i+1} / ||y_{i+1}||_2          ▷ Približek za lastni vektor
5:         if dovolj natančno then
6:             λ ← x_i^T A x_i                        ▷ Približek za lastno vrednost
7:             break
```

Trditev 4.1. *Potenčna metoda konvergira k dominantnemu lastnemu paru, če je* $|\lambda_2/\lambda_{\max}| < 1$, *kjer je* λ_2 *po absolutni vrednosti druga največja lastna vrednost dane matrike. Red konvergence je odvisen od velikosti razmerja* $|\lambda_2/\lambda_{\max}|$ *. Ko se razmerje bliža 1, postaja konvergenca vedno počasnejša.*

Če sta začetni približek x_0 *in dominantni lastni vektor* $x_{\max}$ *ortogonalna, potenčna metoda ne konvergira.*

Dokaz. Dokaz v [36]. □

Precejšnja pomanjkljivost potenčne metode je v tem, da ni stabilna, ko naredimo veliko korakov, saj se razlika med vektorji (kót med njimi) manjša. Posebno težavo predstavlja tudi konvergenca, ki je zelo počasna, če sta po absolutni vrednosti največji lastni vrednosti skoraj enaki. Če imamo srečo in metoda hitro konvergira, s potenčno metodo zelo enostavno dobimo približek za dominanten lastni par.

4.2 Iterativne metode za reševanje sistemov linearnih enačb

Najenostavnejše metode za iterativno reševanje velikih sistemov linearnih enačb oblike $\mathbf{A}x = b$ so Jacobijeva, Gauss-Seidelova in SOR metoda.

Pri vseh treh metodah najprej razdelimo matriko $\mathbf{A}$ na tri dele

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U},$$

kjer so $\mathbf{L}$, $\mathbf{D}$ in $\mathbf{U}$ strogo spodnje-trikotna matrika, diagonalna matrika in strogo zgornje-trikotna matrika. Predpostavimo, da so vsi diagonalni elementi matrike $\mathbf{A}$ različni od 0.

Definicija 4.1. *Jacobijeva iteracija* izračuna približke za rešitev x linearnega sistema $Ax = b$ po formuli

$$x_{k+1}^{(J)} = \mathbf{D}^{-1}\big(b - (\mathbf{L} + \mathbf{U})x_k^{(J)}\big).$$

Podobno deluje tudi *Gauss-Seidelova iteracija*, ki za izračun naslednjega približka uporablja že izračunane vrednosti na trenutnem koraku iteracije

$$x_{k+1}^{(GS)} = (\mathbf{D} + \mathbf{L})^{-1}\big(b - \mathbf{U}x_k^{(GS)}\big).$$

Obe iteraciji lahko zapišemo v obliki

$$\mathbf{M}x_{k+1} = \mathbf{N}x_k + b = (\mathbf{M} - \mathbf{A})x_k + b,$$

za neko razbitje matrike $\mathbf{A} = \mathbf{M} - \mathbf{N}$. Za vsako tako razbitje lahko definiramo utstrezno iteracijo, če je matrika $\mathbf{M}$ nesingularna. Iteracijo dobimo iz $\mathbf{A}x = \mathbf{M}x - \mathbf{N}x = b$ in zato $x = \mathbf{M}^{-1}\mathbf{N}x + \mathbf{M}^{-1}b = \mathbf{R}x + c$. Za Jacobijevo iteracijo je $\mathbf{M} = \mathbf{D}$ in za Gauss-Seidelovo iteracijo je $\mathbf{M} = \mathbf{D} + \mathbf{L}$.

Metoda SOR je izboljšava Gauss-Seidelove iteracije. Uvedemo relaksacijski parameter ω, s katerim utežimo približke iz prejšnjega in trenutnega koraka. Parameter izberemo med 0 in 1. Če je $\omega = 1$, dobimo ravno Gauss-Seidelovo iteracijo. Pri metodi SOR matriko $\mathbf{A}$ razbijemo na

$$\omega\mathbf{A} = (\mathbf{D} + \omega\mathbf{L}) + (\omega\mathbf{U} - (1 - \omega)\mathbf{D}).$$

Definicija 4.2. Iteracija, ki jo izvajamo pri *metodi SOR* za dan parameter ω, je

$$(\mathbf{D} + \omega\mathbf{L})x_{k+1}^{(SOR)} = \big(-\omega\mathbf{U} + (1 - \omega)\mathbf{D}\big)x_k^{(SOR)} + \omega b.$$

Povejmo nekaj še o konvergenci iterativnih metod za reševanje linearnih sistemov.

Trditev 4.2. *Iteracija $x_{k+1} = \mathbf{R}x_k + c$ konvergira k rešitvi enačbe $\mathbf{A}x = b$ pri poljubnem začetnem približku x_0 in pri poljubni desni strani b natanko tedaj, ko je $|\lambda_{\max}(\mathbf{R})| < 1$.*

Dokaz. Dokaz v [36]. □

Definicija 4.3. Matrika $\mathbf{A}$ je *strogo diagonalno dominantna*, če velja

$$|a_{jj}| > \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}|, \quad j = 1, 2, \ldots, n.$$

Trditev 4.3. *Če je matrika $\mathbf{A}$ strogo diagonalno dominantna, potem Jacobijeva in Gauss-Seidelova iteracija konvergirata za vsak začetni približek x_0.*

Dokaz. Dokaz v [75]. □

Poglavje 5

(Kratek) pregled področja časovnih omrežij

V časovnem omrežju se prisotnost in dejavnost vozlišč in povezav lahko spreminjata skozi čas. Časovne podatke so dodali omrežjem na različnih znanstvenih področjih. Začetki segajo v analizo transportnih omrežij [9, 32], v načrtovanje projektov (CPM, Pert) v operacijskih raziskavah [64] in v omrežja omejitev v umetni inteligenci [35]. Obstajajo tudi kvalitativni pristopi k analizi časovnih omrežij [1, 79]. Za statistični pristop glejte [60, 78]. Naš pristop je kvantitativen.

Veliko v zadnjih letih opravljenih raziskav je bilo namenjenih dinamičnim sistemom, ki so na prvi pogled različni. Najbolj se razvijajo področja komunikacijskih omrežij (oddajanje in prenašanje sporočil, novic, telefonskih klicev, elektronskih sporočil), transportnih omrežij (razvoz in prometni tokovi) in bioloških omrežij (nevroznanost, genetika). Pregled najdete v [56, 57]. Vsaka znanstvena disciplina raziskuje določene lastnosti, ki so za to področje najbolj zanimive. Pogosto so znotraj področja določene lastnosti tudi formalno zapisali. V več primerih se je zgodilo, da so različno poimenovani pojavi v resnici pomenili isto, enako ali vsaj zelo podobno stvar. Na primer koncept časovne razdalje [82], čas dosegljivosti [55], zakasnitev informacije in še kak drug izraz poimenujejo isto stvar v različnih okoljih. Podoben primer je tudi pojem potovanja [82], ki ga drugi imenujejo časovna pot, pot glede na čas, pot z urnikom in podobno. Vsi izrazi označujejo podobne, če ne enake zamisli, vendar zadovoljivega formalnega opisa še ni.

Skupna točka vseh raziskav je, da se omrežje spreminja s časom in da sta količina in hitrost sprememb preveliki in preveč pomembni za klasično statično analizo. V teh omrežjih časovne spremembe predstavljajo ključno informacijo.

Kot je (statičen) graf naraven način za zapis statičnega omrežja, želimo, da bi našli dinamični (časovni) graf, s katerim bi lahko opisali zelo dinamična omrežja, ki se spreminjajo s časom. Vse definicije in postopki, ki so jih uvedli razisko-

valci z različnih področij, očitno želijo doseči isto. To pove že sam izraz časovno omrežje. Nekateri raziskovalci uporabljajo časovne oznake na povezavah, drugi pa govorijo o razvijajočem se grafu, ki ga definiramo kot zaporedje statičnih grafov. Precejšen del analize omrežij, v katerih so dostopni tudi časovni podatki, deluje tako, da časovno omrežje definiramo kot zaporedje statičnih grafov, ki predstavljajo statično sliko dinamičnega omrežja v določenem trenutku. Takemu statičnemu grafu rečemo *časovna rezina*. Razlike so v tem, kaj storiti med različnimi rezinami. Običajno vrednosti nekako združimo v eno samo statično vrednost.

Eden od možnih opisov dinamičnega omrežja je *time-aggregated graph* [48, 49, 50] oblike

$$taG = (\mathcal{V}, \mathcal{L}, TF, f_1, \ldots, f_n, g_1, \ldots, g_m, w_1, \ldots, w_m),$$

kjer je $\mathcal{V}$ množica vozlišč, $\mathcal{L}$ množica povezav in TF predstavlja *življenjsko dobo* omrežja. Funkcije $f_i : \mathcal{V} \to \mathbb{R}^{TF}$ in $g_i : \mathcal{L} \to \mathbb{R}^{TF}$ so preslikave vozlišč (povezav) v časovno vrsto, ki jo asociiramo z vozlišči (povezavami). Vrednosti w_i predstavljajo uteži na povezavah, ki so odvisne od časa (običajno potovalne čase). Vsako vozlišče in vsaka povezava ima pripisano časovno vrsto, ki predstavlja čase, ob katerih je vozlišče (povezava) prisotno v omrežju. Za to obliko zapisa je bil razvit postopek za iskanje najhitrejšega potovanja med vozliščema, ki se začne v določenem trenutku, in najboljši čas, ob katerem začeti potovanje. Izkaže se, da problema nista enostavna, saj ne velja, da je začetek najhitrejšega potovanja tudi sam najhitrejši. Zato požrešna metoda odpove.

Pristop, ki se najbolj približa enotni formulaciji, je pristop *time-varying graph*, TVG, opisan v [29].

5.1 Zapis časovnega omrežja v obliki TVG

V tem razdelku bomo opisali način zapisa časovnega omrežja, ki mu avtorji v [28, 29] pravijo TVG (*Time-Varying Graph*). Na tem modelu temelji naš opis časovnega omrežja.

TVG je opisan z množico akterjev $\mathcal{V}$ in množico relacij med akterji $\mathcal{L}$. Relacije so lahko različnih vrst, ki so opisane z abecedo Σ. Torej $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{V} \times \Sigma$. Pomen množice Σ je odvisen od problema. Lahko jo tudi izpustimo.

Ker TVG opisuje dinamično omrežje, predpostavimo, da relacije obstajajo znotraj časovnega intervala $\mathcal{T} \subseteq \overline{\mathbb{N}}$ ali $\mathcal{T} \subseteq \overline{\mathbb{R}_0^+}$, ki mu pravimo *življenjska doba* omrežja. Spremembe, ki se v opazovanem sistemu dogajajo skozi čas, opišemo s pomočjo grafa TVG $\mathcal{G} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \rho, \zeta)$, kjer sta

$$\rho : \mathcal{L} \times \mathcal{T} \to \{0, 1\}$$

funkcija *prisotnosti*, ki opisuje, ali je določena povezava prisotna v omrežju v danem trenutku, in

$$\zeta : \mathcal{L} \times \mathcal{T} \to \mathcal{T}$$

funkcija *potovalnega časa*, ki opisuje čas, ki ga potrebujemo, da prečkamo povezavo v določenem trenutku.

Model lahko naravno posplošimo s funkcijo *prisotnosti vozlišč* $\psi : \mathcal{V} \times \mathcal{T} \to \{0, 1\}$ in s funkcijo *čakalnega časa* $\varphi : \mathcal{V} \times \mathcal{T} \to \mathcal{T}$, ki opisuje čas čakanja v določenem vozlišču.

5.1.1 TVG in zaporedje časovnih rezin

Za dan TVG $\mathcal{G} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \rho, \zeta)$ definiramo *časovno rezino* grafa TVG med časovnima točkama t_1 in t_2 kot statični graf $\mathcal{G}^{[t_1,t_2)} = (\mathcal{V}, \mathcal{L}^{[t_1,t_2)})$, ki vsebuje povezave $\mathcal{L}^{[t_1,t_2)} = \{\ell \in \mathcal{L};\ \exists t \in [t_1, t_2), \rho(\ell, t) = 1\}$. Časovna rezina združi vse interakcije v določenem časovnem obdobju v statičen graf.

Če razbijemo življenjsko dobo $\mathcal{T}$ omrežja na zaporedje časovnih intervalov $\mathcal{T} = \big([t_i, t_{i+1})\big),\ i \in I \subseteq \mathbb{N}$, lahko definiramo in analiziramo pripadajoče zaporedje časovnih rezin $SF(t) = (\mathcal{G}^{[t_0,t_1)}, \mathcal{G}^{[t_1,t_2)}, \dots)$.

Izbira dolžine intervalov je odvisna od omrežja, ki ga raziskujemo. Enak pristop lahko uberemo tudi, če se intervali prekrivajo. Zanimiv je tudi primer, v katerem tudi vozlišča niso prisotna ves čas.

Ker je vsak graf v zaporedju časovnih rezin statičen, lahko na njem izračunamo katero od klasičnih mer za analizo omrežij. V tem primeru je zelo pomembno, kako smo izbrali dolžine časovnih intervalov. Če je čas diskreten, lahko izberemo dolžino intervala enako najmanjši časovni enoti. Pogosto interval izberemo tako, da časovne točke $t_0, t_1, t_2, \dots$ pomenijo čase, ob katerih se katera od lastnosti omrežja spremeni. V tem primeru je zaporedje enakovredno modelu razvijajočih se grafov iz [42]. Drug ekstrem je, da izberemo en sam interval $[t_0, t_1) = \mathcal{T}$. Potem zaporedje vsebuje le eno časovno rezino, ki združi vse interakcije znotraj življenjske dobe omrežja. Če raziščemo razvoj statičnih parametrov v zaporedju časovnih rezin, lahko začenjamo razumeti nekatere pojave v časovnem omrežju [76].

Vse tiste mere, ki so osnovane na poteh, so v primeru zaporedja časovnih rezin vprašljive. V časovnem omrežju poti postanejo potovanja, za katera nujno potrebujemo podatke o časih. Takih mer ne moremo posplošiti na zaporedje statičnih grafov, saj se lahko zgodi, da obstaja pot med vozliščema u in v v vseh časovnih rezinah, potovanje med njima pa ne obstaja. Analizo takih primerov moramo delati na dinamičnih omrežjih.

5.1.2 Potovanja in razdalje med vozlišči v zapisu TVG

Zaporedje parov $\sigma = \{(\ell_1, t_1), (\ell_2, t_2), \ldots, (\ell_k, t_k)\}$, za katere $(\ell_1, \ell_2, \ldots, \ell_k)$ predstavlja sprehod v statičnem grafu $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, je *potovanje*, če velja $\rho(\ell_i, t_i) = 1$ in $t_{i+1} \geq t_i + \zeta(\ell_i, t_i)$ za vse $i < k$. Včasih je dobro dodati še zahtevo, da mora povezava obstajati ves čas, ko potujemo prek nje, $\rho_{[t_i, t_i+\zeta(\ell_i,t_i)]}(\ell_i) = 1$.

Čas t_1 imenujemo *odhod,* čas $t_k + \zeta(\ell_k, t_k)$ pa *prihod* potovanja σ.

Potovanja imajo več dolžin – fizično in časovno. *Dolžina* potovanja $|\sigma| = k$ je enaka številu povezav, ki jih prehodimo, *trajanje* potovanja pa je enako času prihod – odhod.

Naj bo $\mathcal{J}^\star$ množica vseh potovanj v danem TVG in $\mathcal{J}^\star_{uv}$ množica vseh potovanj v danem TVG, ki se začnejo v vozlišču u in končajo v vozlišču v. Če tako potovanje obstaja, pravimo, da je vozlišče v *dosegljivo* iz vozlišča u. Očitno dosegljivost ni simetrična ne glede na to, ali so povezave usmerjene ali ne.

Razdalje med vozlišči lahko definiramo na dva različna načina:

Grafovska razdalja $d_{u,t}(v)$ od vozlišča u do vozlišča v ob času t je definirana kot

$$d_{u,t}(v) = \min\{|\sigma| : \sigma \in \mathcal{J}^\star_{uv},\ \text{odhod}(\sigma) \geq t\}.$$

Za dan čas t je potovanje z odhodom $t' \geq t$ in grafovsko razdaljo enako $d_{u,t}(v)$ *najkrajše* potovanje.

Časovna razdalja $\hat{d}_{u,t}(v)$ od vozlišča u do vozlišča v ob času t je definirana kot

$$\hat{d}_{u,t}(v) = \min\{\text{prihod}(\sigma) : \sigma \in \mathcal{J}^\star_{uv},\ \text{odhod}(\sigma) \geq t\} - t.$$

Za dan čas t je potovanje z odhodom $t' \geq t$ in prihodom $t + \hat{d}_{u,t}(v)$ *prvo* potovanje. Za dan čas t je potovanje z odhodom $t' \geq t$ in časovno razdaljo enako $\{\min \hat{d}_{u,t'}(v) : t' \geq t\}$ *najhitrejše* potovanje.

Algoritme za izračun najkrajšega, prvega in najhitrejšega potovanja so za poseben primer časovnih omrežij razvili v [82].

Računska zahtevnost standardnih problemov iz statičnih omrežij je v časovnih omrežjih lahko večja. Na primer, problem iskanja krepko povezanih komponent v časovnem omrežju je NP-poln [11, 67].

5.2 Primeri časovnih omrežij

5.2.1 Omrežja citatov

Omrežja citatov dobimo iz bibliografskih baz podatkov kot sta Web of Science (Knowledge) [81] in Scopus [77]. V omrežju citatov vozlišča predstavljajo izbrana dela (knjige, članki, poročila, patenti itd.). Med delom u in delom v obstaja usmerjena povezava (u, v), če delo u citira delo v. Življenjska doba omrežja je

interval let od prvega do zadnjega objavljenega dela. Vozlišče v je v omrežju prisotno od trenutka objave. Povezavo lahko opazujemo na dva načina – lahko rečemo, da je prisotna le tistega leta, ko je bilo delo objavljeno, lahko pa je prisotna od trenutka objave naprej. Omrežju lahko dodamo še druge podatke, npr. število strani ali jezik, v katerem je delo napisano. Povezave del z avtorji, ključnimi besedami in založniki (revijami) običajno predstavimo v obliki dvovrstnih omrežij.

5.2.2 Omrežja sodelovanj v projektih

Omrežja sodelovanj v projektih pridobimo iz baz podatkov, kakršna je Cordis [30]. Množico vozlišč sestavljajo vse institucije, ki so sodelovale pri katerem od projektov. Med vozliščema obstaja povezava, če sta instituciji sodelovali pri skupnem projektu. Življenjska doba omrežja je interval datumov (dni), za katere so bili podatki zbrani. Vozlišča so v omrežju prisotna ves čas, povezave pa le na časovnem intervalu, v katerem sta instituciji sodelovali pri skupnem projektu.

5.2.3 Omrežja KEDS/WEIS političnih dogodkov

To so omrežja, v katerih so zapisani politični dogodki v kritičnih področjih (Srednji Vzhod, Balkan, zahodna Afrika) na podlagi dnevnih poročil. Včasih je novice zbiral KEDS (Kansas Event Data System), trenutno pa jih zbirajo pri Parus Analytical Systems [70].

Množica vozlišč vsebuje vse vpletene akterje (države, politične skupine, mednarodne organizacije itd.), povezave pa so usmerjene in opisujejo dogodke:

$$(datum, akter_1, akter_2, dogodek)$$

Na datum $datum$ je $akter_1$ storil $dogodek$ proti $akter_2$. Različni dogodki določajo različne relacije – dobimo večrelacijsko omrežje z množico povezav, ki ima razbitje glede na relacije, ki jim povezave pripadajo.

Življenjsko dobo spet določa opazovano obdobje. Ker večina akterjev obstaja ves čas, so vsa vozlišča stalno prisotna v omrežju. Čas, ob katerem so pristone povezave v omrežju, je določen z dnevi, ob katerih sta se utrezna akterja soočala.

Omrežje bi lahko zapisali tudi kot enorelacijsko omrežje in informacije o vrsti dogodka shranili poleg vrednosti na povezavi.

5.2.4 Drugi primeri

S časovnimi omrežji lahko opisujemo tudi rodovnike, omrežja kontaktov, omrežja telefonskih klicev, vozne rede transportnih služb itd. Velik del časovnih omrežij, v katerih so potovalni in čakalni časi bistvenega pomena, predstavljajo omrežja

med sateliti, vozili in večina komunikacijskih omrežij. V teh primerih je jasno, da lahko komunikacija poteka nemoteno tudi, ko omrežje v določenem trenutku ni povezano. Lahko se celo zgodi, da nikoli ni povezano v smislu statičnega omrežja, komunikacija pa poteka nemoteno. Pomislite npr. na omrežje elektronske pošte. Več primerov je opisanih v [13].

Poglavje 6

Polkolobarji časovnih količin

Za opis časovnih omrežij predlagamo posplošeno različico pristopa, ki ga uporabljamo v programu Pajek [34]. Podoben je pristopu TVG. Članki, ki uporabljajo pristop TVG, večinoma raziskujejo potovalne čase. V našem pristopu obravnavamo tudi vrednosti na povezavah. V večini primerov merijo intenzivnost ali pogostost dejavnosti, lahko pa bi dodali vrednosti uteži, kapacitete, cene, dolžine povezav, potovalne čase itd.

V Pajku poznamo dva načina opisov časovnih omrežij – glede na prisotnost in glede na dogodke (Pajek 0.47, julij 1999). Naš pristop opisuje prisotnost vozlišč in povezav. V tem delu bomo razdelali le pristop, ki temelji na prisotnosti.

Definicija 6.1. *Časovno omrežje* $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \mathcal{P}, \mathcal{W})$ dobimo tako, da običajnemu (statičnemu) omrežju $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{P}, \mathcal{W})$ dodamo čas $\mathcal{T}$. Množica $\mathcal{T}$ je množica časovnih točk $t \in \mathcal{T}$, ki jo imenujemo *življenjska doba* omrežja. Z množico $\mathcal{P}$ označujemo množico časovnih lastnosti vozlišč in z $\mathcal{W}$ množico časovnih lastnosti povezav (uteži na povezavah) [5]. Življenjska doba $\mathcal{T}$ je običajno ali podmnožica celih števil $\mathcal{T} \subseteq \overline{\mathbb{Z}}$ ali podmnožica realnih števil $\mathcal{T} \subseteq \overline{\mathbb{R}}$. V Pajku uporabljamo $\mathcal{T} \subseteq \overline{\mathbb{N}}$.

Definicija 6.2. V časovnem omrežju vozlišča $v \in \mathcal{V}$ in povezave $\ell \in \mathcal{L}$ niso nujno ves čas prisotne ali dejavne. Naj bo $T(v),\ T \in \mathcal{P}$, množica časovnih točk, ob katerih je prisotno vozlišče v, in $T(\ell),\ T \in \mathcal{W}$, množica časovnih točk, ob katerih je prisotna povezava ℓ. V časovnem omrežju zahtevamo, da velja *pogoj usklajenosti*: Če je v trenutku t prisotna povezava $\ell(u, v)$, potem morata biti v trenutku t prisotni tudi njeni krajišči u in v. Formalno pogoj zapišemo kot

$$T(\ell(u, v)) \subseteq T(u) \cap T(v). \tag{6.1}$$

Definicija 6.3. Statično omrežje, ki vsebuje povezave in vozlišča, ki so prisotna v časovnem omrežju ob času $t \in \mathcal{T}$, označimo z $\mathcal{N}(t)$ in mu rečemo *časovna rezina*

omrežja ob času t. Naj bo $\mathcal{T}' \subset \mathcal{T}$ (na primer časovni interval). Zamisel časovne rezine posplošimo na množico $\mathcal{T}'$ kot

$$\mathcal{N}(\mathcal{T}') = \bigcup_{t \in \mathcal{T}'} \mathcal{N}(t).$$

Kadar nas zanimajo sprehodi v časovnih omrežjih imamo običajno na voljo dve dodatni informaciji na povezavah:

Definicija 6.4. *Potovalni čas* $\tau \in \mathcal{W},\ \tau : \mathcal{L} \times \mathcal{T} \to \overline{\mathbb{R}_0^+}$. Vrednost $\tau(\ell, t)$ predstavlja čas, ki ga potrebujemo za prečkanje povezave ℓ, če povezavo prečkamo ob času t. Če funkcija τ ni podana, lahko predpostavimo, da velja $\tau(\ell, t) = 0$ za vse povezave $\ell \in \mathcal{L}$ in vse čase $t \in \mathcal{T}$.

Definicija 6.5. *Utež* $w \in \mathcal{W},\ w : \mathcal{L} \times \mathcal{T} \to \overline{\mathbb{R}}$, katere vrednost $w(\ell, t)$ predstavlja dolžino, ceno, tok itd. na povezavi ℓ v trenutku t. Če funkcija w ni podana, predpostavimo, da velja $w(\ell, t) = 1$ za vse povezave $\ell \in \mathcal{L}$ in vse čase $t \in \mathcal{T}$. V določenih primerih so lahko uteži strukturirane.

Definicija 6.6. Sprehod v časovnem omrežju imenujemo *potovanje*. Potovanje $\sigma(v_0, v_k, t_0)$ od začetnega vozlišča v_0 do končnega vozlišča v_k z začetnim časom t_0 je končno zaporedje

$$(t_0, v_0, (t_1, \ell_1), v_1, (t_2, \ell_2), v_2, \ldots, v_{k-2}, (t_{k-1}, \ell_{k-1}), v_{k-1}, (t_k, \ell_k), v_k),$$

kjer so $v_i \in \mathcal{V}, i = 0, 1, \ldots, k$, in $\ell_i \in \mathcal{L}, t_i \in \mathcal{T}, i = 1, 2, \ldots, k$. Trojice $v_{i-1}, (t_i, \ell_i)$ povejo, da smo iz vozlišča v_{i-1} ob času t_i startali prek povezave ℓ_i. Označimo $t'_0 = t_0,\ t'_i = t_i + \tau(\ell_i),\ i = 1, 2, \ldots, k$. Za potovanje mora veljati $\ell_i(v_{i-1}, v_i),\ t'_{i-1} \leq t_i$ in povezava ℓ_i mora biti prisotna v časovnem intervalu $[t_i, t'_i]$ za vse $i = 1, 2, \ldots, k$. Zahtevamo še, da je ob času t_0 vozlišče v_0 prisotno (po pogoju usklajenosti velja tudi, da je vozlišče v_i prisotno ob času t_i in vozlišče v_{i+1} prisotno ob času t'_i).

Definicija 6.7. Pravimo, da je potovanje *regularno*, če je vozlišče v_i prisotno v omrežju ves časovni interval $[t'_{i-1}, t_i],\ i = 1, 2, \ldots, k-1$. Medtem ko čakamo v vozlišču na prehod prek naslednje povezave, mora biti vozlišče prisotno.

Definicija 6.8. Potovanje σ ima (grafovsko) *dolžino* enako številu k prepotovanih povezav, $|\sigma| = k$. *Trajanje* potovanja je enako $t(\sigma) = t'_k - t_0$ in *vrednost* potovanja je enaka

$$w(\sigma) = w(\ell_1, t_1) \odot w(\ell_2, t_2) \odot \cdots \odot w(\ell_k, t_k) = \bigodot_{(t,\ell) \in \sigma} w(\ell, t)$$

za množenje v ustreznem polkolobarju. Čas t_0 je *začetni čas* potovanja, t_1 je *odhod* in t'_k je *prihod* (končni čas potovanja). Količina $t'_k - t_1$ je *strogo trajanje* potovanja. Časom $t_i - t'_{i-1}$ pravimo *čakalni časi* potovanja.

Definicija 6.9. *Najhitrejše potovanje* je tisto, ki ima najmanjše strogo trajanje, *prvo potovanje* je tisto, ki ima najmanjši čas prihoda. *Najkrajše potovanje* je potovanje z najmanjšo vrednostjo.

Definicija 6.10. *Skok* je potovanje znotraj izbrane časovne rezine $\mathcal{N}(t)$. Skoki imajo ničelni potovalni in čakalni čas.

Definicija 6.11. Delu potovanja $\sigma(v_0, v_k, t_0)$ od vozlišča v_i do vozlišča v_j z začetnim časom t_i,

$$(t_i, v_i, (t_{i+1}, \ell_{i+1}), v_{i+1}, \ldots, v_{j-1}, (t_j, \ell_j), v_j),$$

pravimo *odsek potovanja*. *Potovanje prvih odsekov* je prvo potovanje, v katerem je vsak odsek prvo potovanje med vozliščema v_i in v_j z začetkom po času t_i.

6.1 Časovne količine

Poleg prisotnosti (odsotnosti) vozlišč in povezav se lahko časovno spreminjajo tudi njihove lastnosti. Naj bo $a(t)$ vrednost lastnosti a v trenutku t. Predpostavimo, da vrednosti funkcije $a(t)$ pripadajo polkolobarju $(A, \oplus, \odot, 0, 1)$. Ker vozlišče ali povezava, katere lastnost opisuje funkcija a, lahko ne obstaja v nekaterih trenutkih, funkcija a ni povsod definirana. Njeno definicijsko območje označimo s T_a. Za opis časovnih sprememb lastnosti za celoten čas $\mathcal{T}$ uvedemo časovne količine.

Definicija 6.12. Naj bo $(A, \oplus, \odot, 0, 1)$ polkolobar in naj funkcija $a : T_a \to A$ opisuje lastnost v omrežju, ki se spreminja s časom. *Časovna količina* $\hat{a} : \mathcal{T} \to A$ je razširitev funkcije a,

$$\hat{a}(t) = \begin{cases} a(t), & t \in T_a, \\ 0, & t \in \mathcal{T} \setminus T_a. \end{cases}$$

Opozorimo, da smo vrednost časovne količine, ko vozlišče ali povezava ne obstaja, določili kot nevtralni element za seštevanje ustreznega polkolobarja A, ki je tudi ničla za množenje. To pomeni, da bo vrednost potovanj (staknjenih zaporednih potovanj) enaka 0 (neobstoječe potovanje), če eno izmed zaporednih potovanj ne obstaja.

V nadaljevanju časovne količine označujemo z a namesto z $\hat{a}$.

6.2 Enostavni časovni polkolobarji

Najprej se omejimo na primer, ko so potovalni in čakalni časi v omrežju ves čas enaki nič.

Definicija 6.13. Naj bo $A_{\mathcal{T}}$ množica vseh časovnih količin nad polkolobarjem A za čas $\mathcal{T}$, $A_{\mathcal{T}} = \{a : \mathcal{T} \to A\}$. Oznaka A lahko označuje tudi razširjen polkolobar $A_{\text{⌘}}$, če je to potrebno. V množici časovnih količin $A_{\mathcal{T}}$ definiramo *vsoto* (spomnimo se, da vsota ustreza vzporednima povezavama)

$$(a \oplus b)(t) = a(t) \oplus b(t),$$

in *produkt* (ustreza zaporednima povezavama)

$$(a \odot b)(t) = a(t) \odot b(t).$$

Operaciji na levi strani delujeta v množici $A_{\mathcal{T}}$ časovnih količin nad polkolobarjem A za čas $\mathcal{T}$, operaciji na desni strani pa v polkolobarju A.

Trditev 6.1. *Za operaciji iz definicije 6.13 je $A_{\mathcal{T}}$ polkolobar z nevtralnim elementom $0(t) = 0,\ t \in \mathcal{T}$, in enoto za množenje $1(t) = 1,\ t \in \mathcal{T}$.*

Dokaz. Lastnosti polkolobarja $A_{\mathcal{T}}$ sledijo iz lastnosti polkolobarja A, saj sta operaciji definirani po točkah. □

Definicija 6.14. Naj bo A kombinatorični (povezanostni, geodezični itd.) polkolobar. Polkolobarju $A_{\mathcal{T}}$ pravimo *enostavni časovni kombinatorični (povezanostni, geodezični itd.)* polkolobar.

Nad enostavnimi časovnimi polkolobarji lahko definiramo polkolobar matrik z elementi, ki so časovne količine (trditev 2.1, stran 9).

Definicija 6.15. Matrikam in vektorjem, ki imajo za elemente časovne količine, pravimo *časovne matrike* in *časovni vektorji*.

Vrednostne matrike časovnega omrežja brez potovalnega in čakalnega časa vsebujejo elemente iz enostavnega časovnega polkolobarja $A_{\mathcal{T}}$. Ker vrednosti $a(t)$ in $b(t)$ v definicijah ustrezata isti časovni točki t, morata biti potovalni in čakalni čas v omrežju ničelna za vse čase iz $\mathcal{T}$. Uporaba tega polkolobarja v časovnih omrežjih je omejena na skoke in ne na splošna potovanja, da imata operaciji smisel.

6.3 Polkolobar naraščajočih funkcij

Definicija 6.16. Funkcija f je *naraščajoča*, če velja $f(x) \geq f(y)$ za vse x, y iz njenega definicijskega območja, za katere je $x \geq y$.

Pravimo, da je funkcija f *povečujoča*, če velja $f(x) \geq x$ za vse x iz njenega definicijskega območja.

Trditev 6.2. *Množica*

$$A = \left\{ \; f : \overline{\mathbb{N}} \to \overline{\mathbb{N}};\ \textit{funkcija } f \textit{ je naraščajoča in povečujoča} \; \right\}$$

je polkolobar za operaciji

$$f \oplus g = \min(f, g),$$
$$f \odot g = g \circ f.$$

Nevtralni element za seštevanje je funkcija $f \equiv \infty$ in enota za množenje je funkcija identitete $f = id$. Za domeno in kodomeno funkcij f bi lahko vzeli tudi množice $\overline{\mathbb{R}_0^+}, \overline{\mathbb{Z}}$ ali $\overline{\mathbb{R}}$.

Dokaz. Asociativnost in komutativnost seštevanja in nevtralni element so očitni. Prav tako sta očitni asociativnost za množenje in enota za množenje. Preverimo še oba distributivnostna zakona:

$$\begin{aligned}
\big(f \odot (g \oplus h)\big)(x) &= \big(f \odot \min(g,h)\big)(x) \\
&= \big(\min(g,h) \circ f\big)(x) \\
&= \min\big(g(f(x)), h(f(x))\big) \\
&= \big(f \odot g \oplus f \odot h\big)(x), \\
\big((g \oplus h) \odot f\big)(x) &= \big((\min(g,h)) \odot f\big)(x) \\
&= \big(f \circ \min(g,h)\big)(x) \\
&= \min\big(f(g(x)), f(h(x))\big), \text{ saj je } f \text{ naraščajoča} \\
&= \big(g \odot f \oplus h \odot f\big)(x).
\end{aligned}$$

Distributivnostna zakona veljata. Nevtralni element je tudi ničla, saj je f povečujoča funkcija:

$$(f \odot \infty)(x) = \infty(f(x)) = \infty \quad \text{in} \quad (\infty \odot f)(x) = f(\infty(x)) = f(\infty) = \infty.$$

Torej je množica A polkolobar. □

Definicija 6.17. Polkolobarju A iz trditve 6.2 pravimo *polkolobar naraščajočih funkcij.*

Polkolobar naraščajočih funkcij je poseben primer algebre endomorfizmov iz [52]. Je poln, idempotenten ($\min(f,f) = f$) in zaprt za $f^\star = 1 \oplus f \odot f^\star = \min(id, f^\star \circ f) = id$, saj sta $f^\star$ in f povečujoči in naraščajoči funkciji. Velja tudi absorpcijski zakon ($\min(id, f) = id$), ker je f povečujoča.

6.4 Polkolobar prvih potovanj

Spomnimo se Bellmanove enačbe (enačba (2.2) na strani 14) za klasičen problem najkrajših poti:

$$d(v) = \min_{u \in \mathcal{V}} \{d(u) + w(u, v)\},$$

kjer $d(v)$ pomeni dolžino najkrajše poti od vozlišča s do vozlišča v in velja $d(s) = 0$.

Poiščimo podobno enačbo za iskanje prvih potovanj v časovnem omrežju. Naj časovna količina a_{uv} opisuje potovalni čas prek povezave (u, v). Naj bo $T(u, v, t_0)$ prvi čas, ob katerem lahko pridemo v vozlišče v, če začnemo v vozlišču u ob času t_0. Potem velja

$$T(u, u, t_0) = t_0$$

in

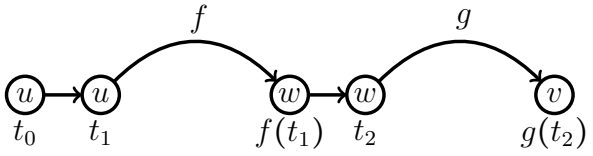
$$T(u, v, t_0) = \min_{w:(w,v)\in\mathcal{L}} \left(\min_{t \geq T(u,w,t_0)} (t + a_{wv}(t)) \right). \tag{6.2}$$

Če želimo izračunati trajanje potovanja, rezultatu odštejemo začetek potovanja t_0.

Podobno kot smo iz Bellmanove enačbe dobili polkolobar najkrajših poti, želimo tudi za ta primer definirati ustrezen polkolobar. To ni ravno očitno, saj imamo tri operacije, dva minimuma in seštevanje. Radi bi poiskali polkolobar, v katerem bosta operaciji $\oplus$ in $\odot$ dali ravno enačbo (6.2). Kot vedno lahko začnemo z dvema vzporednima in z dvema zaporednima povezavama.

Poglejmo, kako lahko enačbo (6.2) razumemo s potovanji v omrežjih. Vidimo, da je pametno definirati funkcijo (časovno količino), ki vrne prvi čas, ob katerem lahko pridemo v izbrano vozlišče, ki bo odvisen tudi od začetnega vozlišča in začetnega časa.

Recimo, da želimo opraviti potovanje po dveh zaporednih povezavah (u, w) in (w, v). Prvi čas, ko lahko pridemo v vozlišče w prek povezave (u, w), opisuje časovna količina f, prvi čas, ko lahko pridemo v vozlišče v prek povezave (w, v), pa opisuje časovna količina g. Potem to potovanje izgleda nekako takole (glej sliko 6.1):



Slika 6.1: Prikaz potovanja po zaporednih povezavah.

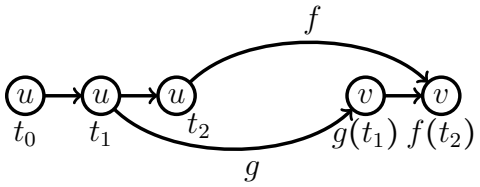
V vozlišču u od začetka potovanja t_0 počakamo na nek ugoden trenutek t_1, ko prečkamo povezavo (u, w). Ta del potovanja se konča ob času $f(t_1)$. Potem v vozlišču w čakamo na ugoden trenutek t_2, ko prečkamo povezavo (w, v). Potovanje se konča ob času $g(t_2)$. Zanima nas prvi čas, ob katerem lahko pridemo iz vozlišča u v vozlišče v prek vozlišča w, če začnemo ob času t_0. V jeziku polkolobarjev nas zanima $f \odot g$. To lahko zapišemo kot

$$(f \odot g)(t_0) = \min_{\substack{t_1 \geq t_0 \\ t_2 \geq f(t_1)}} g(t_2).$$

Če sta f in g naraščajoči funkciji, se enačba poenostavi v

$$(f \odot g)(t_0) = g(f(t_0)) = (g \circ f)(t_0).$$

Poglejmo še, kaj se zgodi pri potovanju po vzporednih povezavah. Eno od možnosti prikazuje slika 6.2. Potovanje začnemo ob času t_0 in čakamo do primernega časa t_1, ko se splača iti prek spodnje povezave, na kateri prvi čas prihoda opisuje funkcija g. To potovanje se konča ob času $g(t_1)$. Če želimo iti prek zgornje povezave, na kateri prvi čas prihoda opisuje funkcija f, čakamo na primeren čas t_2 in končamo potovanje v trenutku $f(t_2)$. Prvi čas, ob katerem lahko iz vozlišča u pridemo v vozlišče v, je manjši od časov $f(t_2)$ in $g(t_1)$.



Slika 6.2: Prikaz potovanja po vzporednih povezavah.

To zapišemo z

$$(f \oplus g)(t_0) = \min_{\substack{t_1 \geq t_0 \\ t_2 \geq t_0}} \big(f(t_2), g(t_1)\big) = \min_{t \geq t_0} \big(f(t), g(t)\big).$$

Če sta f in g naraščajoči, se enačba poenostavi v

$$(f \oplus g)(t_0) = \min(f(t_0), g(t_0)).$$

Primerno seštevanje je $f \oplus g = \min\{f, g\}$ in primerno množenje je $f \odot g = g \circ f$. To sta ravno operaciji v polkolobarju naraščajočih funkcij iz razdelka 6.3.

Naj vrednosti časovne količine a predstavljajo potovalni čas prek povezave ob času t. Opozorimo, da je ob časih $t \in \mathcal{T} \setminus T_a$ vrednost $a(t) = \infty$. Časovni količini a priredimo funkcijo f takole:

$$a \mapsto f: \ f(t) = \min_{\tau \geq t} \{\tau + a(\tau)\}. \tag{6.3}$$

Funkcija f je naraščajoča in povečujoča, če je $a \geq 0$, kar je logična predpostavka, saj potovalni časi niso negativni. Če je a časovna količina, ki opisuje potovalni čas prek povezave (u, v), tako prirejeno funkcijo f razumemo kot prvi čas, ob katerem lahko pridemo iz vozlišča u v vozlišče v, če potovanje začnemo ob času t.

Potovanje v časovnem omrežju s čakalnimi in potovalnimi časi lahko prevedemo na seštevanje in množenje funkcij, prirejenih časovni količini a, v polkolobarju monotonih funkcij.

Definicija 6.18. Naj bo $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, a)$ časovno omrežje in naj časovna količina $a : \mathcal{T} \to \mathcal{T}$ označuje potovalne čase. Časovni količini a priredimo funkcijo f kot v enačbi (6.3). Polkolobar

$$\mathbb{T} = \big(\{f : \mathcal{T} \to \mathcal{T}\}, \min, \circ, \infty, id\big)$$

imenujemo *polkolobar prvih potovanj.*

6.5 Posplošeni geodezični polkolobarji

Posplošene geodezične polkolobarje definiramo po zgledu geodezičnega polkolobarja (razdelek 2.2.4, stran 16).

Definicija 6.19. V množici $\mathcal{T} \times A$, kjer je množica $\mathcal{T}$ enaka $\mathcal{T} = (\overline{\mathbb{R}_0^+}, \min, +, \infty, 0)$ ali $\mathcal{T} = (\overline{\mathbb{N}}, \min, +, \infty, 0)$ in $(A, \oplus, \odot, \mathbf{0}, \mathbf{1})$ poljuben poln polkolobar (recimo kombinatorični polkolobar, povezanostni polkolobar, polkolobar najkrajših poti, geodezični polkolobar itd.), definiramo operaciji *seštevanje* $\boxplus$ in *množenje* $\boxtimes$ kot

$$(\tau, a) \boxplus (\sigma, b) = \left(\min(\tau, \sigma), \begin{cases} a, & \tau < \sigma, \\ a \oplus b, & \tau = \sigma, \\ b, & \tau > \sigma \end{cases} \right)$$

in

$$(\tau, a) \boxtimes (\sigma, b) = (\tau + \sigma, a \odot b).$$

Trditev 6.3. *Množica $\mathcal{T} \times A$ je polkolobar za seštevanje $\boxplus$ in množenje $\boxtimes$. Nevtralni element za seštevanje $\boxplus$ je $(\infty, \mathbf{0})$ in enota za množenje $\boxtimes$ je $(0, \mathbf{1})$.*

Dokaz. Konstrukcija je enaka tisti za geodezični polkolobar in lastnosti polkolobarja sledijo po enakem postopku kot v [4] iz lastnosti operacij v polkolobarjih $\mathcal{T}$ in A. □

Definicija 6.20. Polkolobarju $G_{\mathcal{T} \times A} = \big(\mathcal{T} \times A, \boxplus, \boxtimes, (\infty, \mathbf{0}), (0, \mathbf{1})\big)$ pravimo *posplošeni geodezični polkolobar.*

6.6 Potovalni polkolobarji

Naslednje vprašanje je, kaj storimo, če imamo na povezavah več vrednosti, na primer potovalni čas in število načinov, kako pridemo prek povezave, ali potovalni čas in dolžino poti.

Naj časovna količina $a : \mathcal{T} \to \mathcal{T}$ opisuje potovalni čas prek povezave in naj časovna količina $i \in A_{\mathcal{T}}$ nad izbranim polkolobarjem $(A, \oplus, \odot, \mathbf{0}, \mathbf{1})$ opisuje neko drugo informacijo o povezavi.

V tem primeru želimo izračunati vrednost

$$(f, n)(t) = \left(\min_{\tau \geq t} (a(\tau) + \tau), \bigoplus_{\sigma = \mathrm{Argmin}_{\substack{\tau \geq t \\ \sigma \geq t}} (a(\tau)+\tau)} i(\sigma) \right).$$

Prva komponenta f ostane enaka kot v polkolobarju prvih potovanj (enačba (6.3)) in pove prvi čas, ob katerem lahko pridemo prek povezave po trenutku t. V drugi komponenti n seštevamo (v izbranem polkolobarju A) vrednosti na povezavah, na katerih dosežemo minimalen prvi čas in ki se začnejo po času t.

Izraz ne izgleda preveč lepo, zato najprej definiramo transformacijo

$$\begin{aligned} (a, i) &\mapsto (a', i), \\ a'(t) &= a(t) + t. \end{aligned}$$

Od tod vidimo, da želimo

$$(f, n)(t) = \left(\min_{\tau \geq t} a'(\tau), \bigoplus_{\sigma = \mathrm{Argmin}_{\tau \geq t} a'(\tau)} i(\sigma) \right). \tag{6.4}$$

Zadnji izraz enostavneje zapišemo s pomočjo seštevanja v ustreznem posplošenem geodezičnem polkolobarju $G_{\mathcal{T} \times A}$. Enačbo (6.4) lahko zapišemo kot

$$(f, n)(t) = \bigboxplus_{\tau \geq t} (a'(\tau), i(\tau)). \tag{6.5}$$

Velja $f \in \mathbb{T}$ in $n \in A_{\mathcal{T}}$.

6.6.1 Operacije v potovalnih polkolobarjih

Transformacija iz enačbe (6.5) časovnima količinama a, ki predstavlja potovalni čas, in i, ki predstavlja neko drugo informacijo o povezavi, priredi par (f, n), ki pripada množici

$$G_A(\mathcal{T}) = \{(f, n);\ f \in \mathbb{T},\ n \in A_{\mathcal{T}}\}.$$

Definicija 6.21. Na množici parov funkcij $G_A(\mathcal{T})$ definiramo *seštevanje* $\oplus$ in *množenje* $\diamond$ z

$$\big((f,n) \oplus (g,m)\big)(t) = (f,n)(t) \boxplus (g,m)(t),$$
$$\big((f,n) \diamond (g,m)\big)(t) = \big((g \circ f)(t), n(t) \odot m(f(t))\big).$$

Operacija $\boxplus$ je seštevanje v posplošenem geodezičnem polkolobarju $G_{\mathcal{T}\times A}$ (definicija 6.19). Operacija $\odot$ je množenje v polkolobarju A.

Definiciji seštevanja in množenja lahko preberemo takole: Če imamo na izbiro vzporedni povezavi, izberemo tisto, ki se prej konča, in ohranimo vrednost, ki je dodana izbrani povezavi. Če se obe povezavi končata ob istem času, seštejemo obe dodani vrednosti.

Če imamo staknjeno potovanje na dveh zaporednih povezavah, se potovanje konča takrat, ko se konča odsek prek druge povezave. Odsek prek druge povezave se lahko začne le po času, ko lahko prvič pridemo prek prve povezave (čas $f(t)$). Vrednost na drugi komponenti staknjenih povezav je vrednost, ki je napisana na prvi povezavi, če gremo na pot po času t, krat vrednost, ki je napisana na drugi povezavi, če gremo na pot po času $f(t)$.

Prva komponenta para pove prvi čas, ob katerem lahko pridemo v izbrano vozlišče. Druga komponenta za prvo potovanje pove vrednost, odvisno od polkolobarja A. Če je A kombinatorični polkolobar, druga komponenta pove število potovanj prvih odsekov. Če je A polkolobar najkrajših poti, druga komponenta pove najkrajšo dolžino potovanja prvih odsekov. Če je A geodezični polkolobar, nam druga komponenta (urejeni par) pove dolžino in število najkrajših potovanj prvih odsekov.

Trditev 6.4. *Množica $G_A(\mathcal{T})$ je za operaciji iz definicije 6.21 polkolobar. Nevtralni element je par konstantnih funkcij $(\infty, \mathbf{0})$. Enota za množenje je $(id, \mathbf{1})$. Druga komponenta enote je konstantna funkcija.*

Dokaz. Asociativnost, komutativnost in nevtralni element za seštevanje sledijo iz lastnosti posplošenega geodezičnega polkolobarja.

Pokažimo, da je $(id, \mathbf{1})$ enota za množenje

$$\big((f,n) \diamond (id,\mathbf{1})\big)(t) = \big(f(t), n(t) \odot \mathbf{1}\big) = (f,n)(t),$$
$$\big((id,\mathbf{1}) \diamond (f,n)\big)(t) = \big(f(t), \mathbf{1} \odot n(t)\big) = (f,n)(t).$$

in da je $(\infty, \mathbf{0})$ ničla za množenje

$$\big((f,n) \diamond (\infty,\mathbf{0})\big)(t) = (\infty, n(t) \odot \mathbf{0}) = (\infty, \mathbf{0}),$$
$$\big((\infty,\mathbf{0}) \diamond (f,n)\big)(t) = (f(\infty), \mathbf{0} \odot n(\infty)\big) = (\infty,\mathbf{0}), \text{ saj je } f \text{ povečujoča.}$$

Pokažimo še asociativnost množenja in oba distributivnostna zakona. Najprej asociativnost:

$$\begin{aligned}\big(\big((f,n)\diamond(g,m)\big)\diamond(h,r)\big)(t) &= \big((g\circ f)(t), n(t)\odot m(f(t))\big)\diamond(h(t),r(t))\\ &= \big((h\circ g\circ f)(t), n(t)\odot m(f(t))\odot r((g\circ f)(t))\big)\\ \big((f,n)\diamond\big((g,m)\diamond(h,r)\big)\big)(t) &= (f,n)(t)\diamond\big((h\circ g)(t), m(t)\odot r(g(t))\big)\\ &= \big((h\circ g\circ f)(t), n(t)\odot m(f(t))\odot r(g(f(t)))\big).\end{aligned}$$

Očitno v obeh primerih dobimo enak rezultat, torej asociativnost množenja velja. Preverimo še distributivnost:

$$\begin{aligned}((h,r)\diamond(f,n))(t) &= ((f\circ h)(t), r(t)\odot n(h(t))),\\ ((h,r)\diamond(g,m))(t) &= ((g\circ h)(t), r(t)\odot m(h(t)))\end{aligned}$$

in

$$((h,r)\diamond(f,n)\oplus(h,r)\diamond(g,m))(t) = (\min(f(h(t)), g(h(t))),$$
$$\left\{\begin{array}{ll} r(t)\odot n(h(t)), & f(h(t)) < g(h(t))\\ r(t)\odot(n(h(t))\oplus m(h(t))), & f(h(t)) = g(h(t))\\ r(t)\odot m(h(t)), & f(h(t)) > g(h(t))\end{array}\right\}.$$

Uporabili smo distributivnost v polkolobarju A. Poglejmo, kaj dobimo na drugi strani enakosti za distributivnost:

$$((f,n)\oplus(g,m))(t) = \left(\min(f(t),g(t)), \left\{\begin{array}{ll} n(t), & f(t) < g(t)\\ (n\oplus m)(t), & f(t) = g(t)\\ m(t), & f(t) > g(t)\end{array}\right\}\right),$$

kar množimo z leve $(h,r)(t)\diamond$ in dobimo

$$\left((\min(f,g)\circ h)(t), r(t)\odot\left\{\begin{array}{ll} n(h(t)), & f(h(t)) < g(h(t))\\ (n\oplus m)(h(t)), & f(h(t)) = g(h(t))\\ m(h(t)), & f(h(t)) > g(h(t))\end{array}\right\}\right).$$

Torej leva distributivnost velja. Če množimo $((f,n)\oplus(g,m))(t)$ z desne strani $\diamond(h,r)(t)$ pa dobimo

$$\left((h\circ\min(f,g))(t), \left\{\begin{array}{ll} n(t), & f(t) < g(t)\\ (n\oplus m)(t), & f(t) = g(t)\\ m(t), & f(t) > g(t)\end{array}\right\} \odot r(\min(f(t),g(t)))\right),$$

kar je enako rezultatu naslednjih računov

$$((f,n) \diamond (h,r))(t) = ((h \circ f)(t), n(t) \odot r(f(t))),$$
$$((g,m) \diamond (h,r))(t) = ((h \circ g)(t), m(t) \odot r(g(t))),$$

kar seštejemo s $\oplus$ v

$$\left(\min(h(f(t)), h(g(t))), \begin{cases} n(t) \odot r(f(t)), & h(f(t)) < h(g(t)), \\ n(t) \odot r(f(t)) \oplus m(t) \odot r(g(t)), & h(f(t)) = h(g(t)), \\ m(t) \odot r(g(t)), & h(f(t)) > h(g(t)) \end{cases} \right).$$

Desna distributivnost velja, saj so f, g in h naraščajoče funkcije in je množenje $\odot$ v polkolobarju A distributivno.

Torej veljata oba distributivnostna zakona in je $G_A(\mathcal{T})$ res polkolobar. □

Definicija 6.22. Naj bo A kombinatorični (povezanostni, geodezični itd.) polkolobar. Polkolobarju

$$(G_A(\mathcal{T}), \oplus, \diamond, (\infty, \mathbf{0}), (id, \mathbf{1}))$$

pravimo *potovalni kombinatorični (povezanostni, geodezični itd.)* polkolobar.

6.6.2 Vmesnost glede na potovanja prvih odsekov

Kot smo že omenili, imajo pari funkcij (f, m) iz potovalnih polkolobarjev različne pomene. Najbolj enostavna uporaba potovalnih polkolobarjev je posplošitev vmesnosti iz razdelka 3.4.

Definicija 6.23. *Vmesnost glede na potovanja prvih odsekov* po času t definiramo kot

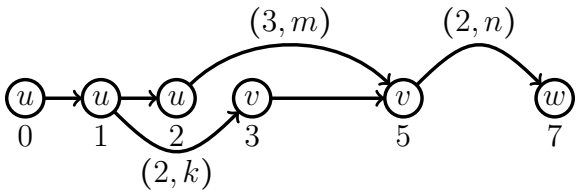
$$\mathfrak{b}_v(t) = \frac{1}{(n-1)(n-2)} \sum_{\substack{u,w \in \mathcal{V} \\ |\{v,u,w\}|=3}} \frac{n_{uw(v)}(t)}{n_{uw}(t)}.$$

Z $n_{uw}(t)$ smo označili število potovanj prvih odsekov od u do w. Časovna količina $n_{uw(v)}(t)$ predstavlja število potovanj prvih odsekov od u do w, ki potekajo skozi v.

V statičnih omrežjih predpostavimo, da vsa komunikacija poteka prek najkrajših poti. Tukaj predpostavimo, da poteka le prek potovanj prvih odsekov. V časovnem omrežju v splošnem ne velja, da vsako prvo potovanje od u do w vsebuje prvo potovanje od u do v in prvo potovanje od v do w. Kot protiprimer poglejmo omrežje na sliki 6.3. Uteži na povezavah so potovalni časi in število (načinov) povezav. Potovalni čas prek povezave (u, v) je enak 2 ob času 1 in 3 ob času 2. Med vozliščema v in w je potovalni čas enak 2 ob času 5. Izven teh časov povezave ne obstajajo.

Med vozliščema u in v obstaja k prvih potovanj, ki se končajo ob času 3. Med vozliščema v in w obstaja n prvih potovanj. Med vozliščema u in w obstaja $(m + k)\cdot n$ prvih potovanj. Težava nastane, ker ne razlikujemo med čakanjem v vozlišču v in potovanjem po povezavi. Povezave (u, v) z utežjo $(3, m)$ v polkolobarju ne upoštevamo, saj ni med prvimi potovanji od u do v in torej ni vsebovana v potovanju prvih odsekov od u do w.

Če želimo upoštevati vsa prva potovanja, pride do težav. Podobno se zgodi, če želimo poiskati potovanja, v katerih čakalni časi niso dovoljeni ali so vnaprej predpisani. Oba problema ostajata nerešena.



Slika 6.3: Prvo potovanje ne vsebuje nujno samo prvih odsekov.

Vrednosti $n_{uw}(t)$ in $n_{uw(v)}(t)$ lahko dobimo iz zaprtja $\mathbf{B}$ časovne vrednostne matrike omrežja nad potovalnim kombinatoričnim polkolobarjem na podoben način kot to storimo v statičnem primeru. Matrika $\mathbf{B}$ vsebuje pare časovnih količin oblike $\big(f_{uv}(t), n_{uv}(t)\big)$. Vrednost $f_{uv}(t)$ je enaka prvemu času, ko lahko iz vozlišča u pridemo v vozlišče v, če potovanje začnemo ob času t. Vrednost $n_{uv}(t)$ pove število potovanj prvih odsekov, ki se začnejo v vozlišču u po času t in se končajo v vozlišču v ob času $f_{uv}(t)$.

Ko poznamo matriko $\mathbf{B}$, izračunamo

$$n_{uw(v)}(t) = n_{uv}(t) \cdot n_{vw}(f_{uv}(t)),$$

če velja $f_{uw}(t) = f_{vw}(f_{uv}(t))$. Sicer je vrednost $n_{uw(v)}(t)$ enaka $(\infty, 0)$.

Poglavje 7

Časovna omrežja brez potovalnih in čakalnih časov

V tem poglavju bomo pokazali, da lahko večino običajnih pristopov in algoritmov za analizo statičnih omrežij naravno razširimo na časovna omrežja brez potovalnih in čakalnih časov. Pri tem uporabljamo ustrezne enostavne časovne polkolobarje (definicija 6.14).

Postopki, ki smo jih razvili za analizo časovnih omrežij brez potovalnih in čakalnih časov, so dostopni kot Pythonska knjižnica TQ (Temporal Quantities – časovne količine) na spletni strani `http://pajek.imfm.si/doku.php?id=tq`. Razvijamo tudi uporabniku prijaznejši program Ianus (po zgledu programa Pajek).

Omejimo se na omrežja, ko so potovalni in čakalni časi enaki nič. Operacije v enostavnih časovnih polkolobarjih so definirane po točkah in vsi algoritmi za časovne matrike konvergirajo pod istimi pogoji kot za statične matrike. Računi s časovnimi matrikami so enakovredni računom na končnem zaporedju statičnih matrik. Zato, da lahko ustvarimo programsko podporo za delo s časovnimi omrežji, se omejimo na časovne količine, ki so odsekoma konstantne funkcije. Potem jih lahko opišemo v obliki [časovni interval I_i, vrednost v_i časovne količine a na intervalu I_i],

$$a = \big((I_i, v_i)\big)_{i=1}^{k}.$$

V splošnem so intervali I_i lahko različnih tipov (polodprti, zaprti, odprti). Odločili smo se, da izberemo intervale oblike $[s_i, f_i)$. Tudi vrednost v_i je lahko strukturirana. Na primer $v_i = (w_i, c_i, \tau_i)$ predstavlja utež w_i, kapaciteto c_i in potovalni čas τ_i ali $v_i = (d_i, n_i)$ predstavlja dolžino najkrajše poti d_i in število najkrajših poti n_i.

Veljati mora $s_i < f_i$ za $i = 1, \ldots, k$ in $f_{i-1} \leq s_i$ za $i = 2, \ldots, k$.

V naši predstavitvi je čas diskreten in omejen, torej $\mathcal{T} = [t_{min}, t_{max}] \subset \overline{\mathbb{N}}$.

Take časovne količine predstavimo z zaporedjem urejenih trojic

$$a = \big((s_i, f_i, v_i)\big)_{i=1}^{k}.$$

V predstavljenih primerih bomo uporabljali Pythonski zapis za časovne količine. Količini a in b v Pythonu predstavimo kot seznam trojic:

```
a = [(1, 5, 2), (6, 8, 1), (11, 12, 3), (14, 16, 2), (17, 18, 5),
     (19, 20, 1)]
b = [(2, 3, 4), (4, 7, 3), (9, 10, 2), (13, 15, 5), (16, 21, 1)]
```

Trojice seznama so naraščajoče urejene po prvem členu s_i. Časovna količina a ima na intervalu $[1,5)$ vrednost 2, na intervalu $[6,8)$ vrednost 1, na intervalu $[11,12)$ vrednost 3 itd. Izven opisanih intervalov ima vrednost enako ničli polkolobarja. Časovni količini a in b grafično predstavimo na sliki 7.1, zgoraj.

V seznamski predstavitvi časovnih količin operaciji $\oplus$ in $\odot$ iz časovnega polkolobarja izračunamo kot

$$(a \oplus b)(t) = \begin{cases} a(t) \oplus b(t), & t \in T_a \cap T_b, \\ a(t), & t \in T_a \setminus T_b, \\ b(t), & t \in T_b \setminus T_a \end{cases}$$

in

$$(a \odot b)(t) = a(t) \odot b(t),\ t \in T_a \cap T_b.$$

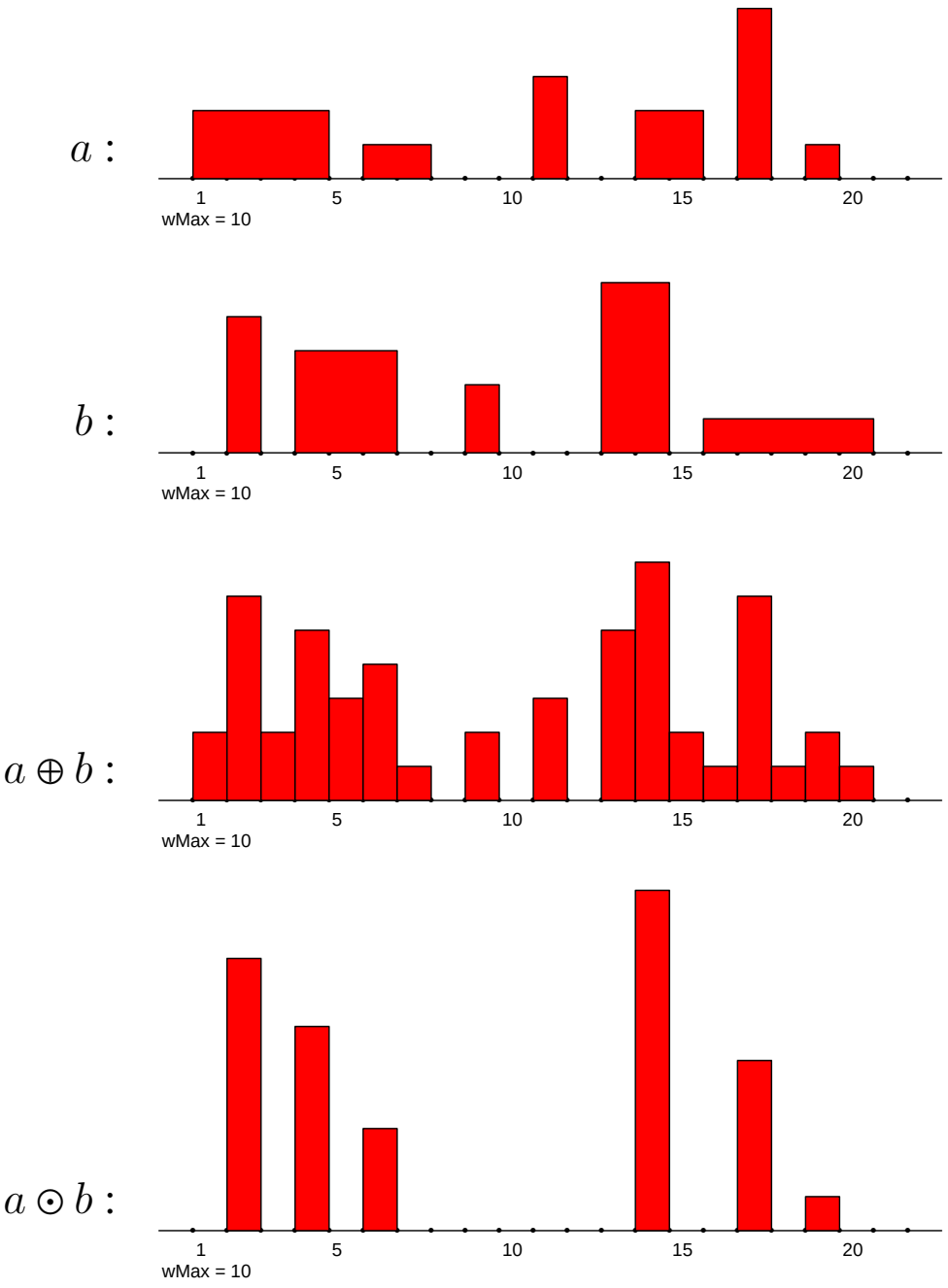
Rezultate zopet izrazimo s seznami trojic.

Psevdokoda algoritmov za seštevanje *sum* in množenje *prod* časovnih količin, zapisanih v tej obliki, je opisana v algoritmih 4 za seštevanje in 5 za množenje. Oba postopka delujeta nad izbranim enostavnim časovnim polkolobarjem (kombinatorični, povezanostni, geodezični itd.). Operaciji seštevanja in množenja izbranega nečasovnega polkolobarja izračunata funkciji *sAdd* in *sMul*. Algoritma temeljita na zlivanju urejenih seznamov.

Funkcija *length*(a) vrne dolžino (število elementov) seznama a. Funkcija *get*(a) vrne zadnji element seznama a in prestavi kazalec na naslednji element. Če je seznam prazen, vrne stražarja $(\infty, \infty, 0)$. Izraz $(s, f, v) \leftarrow e$ razdeli element e v njegove dele. Izraz c.*append*(e) doda element e na konec seznama c. Funkcija *standard*(a) v seznamu a združi sosedne časovne intervale z enako vrednostjo v en sam interval.

Vsota s in produkt p časovnih količin a in b, ki ju izračunamo z opisanima algoritmoma, sta grafično prikazana na spodnji polovici slike 7.1. Predstavitev v Pythonu je

```
s = [(1, 2, 2), (2, 3, 6), (3, 4, 2), (4, 5, 5), (5, 6, 3),
     (6, 7, 4), (7, 8, 1), (9, 10, 2), (11, 12, 3), (13, 14, 5),
     (14, 15, 7), (15, 16, 2), (16, 17, 1), (17, 18, 6),
     (18, 19, 1), (19, 20, 2), (20, 21, 1)]
p = [(2, 3, 8), (4, 5, 6), (6, 7, 3), (14, 15, 10), (17, 18, 5),
     (19, 20, 1)]
```



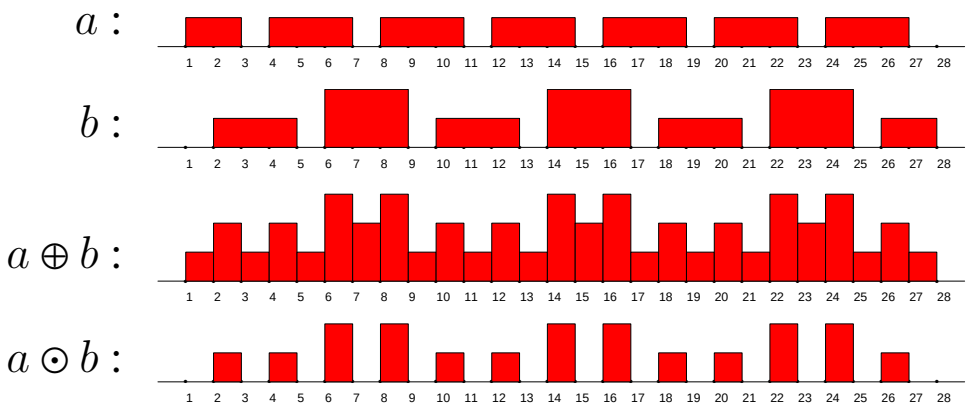
Slika 7.1: Seštevanje in množenje časovnih količin.

Algoritem 4 Seštevanje časovnih količin.

```
1:  function sum(a, b)
2:      if length(a) = 0 then return b
3:      if length(b) = 0 then return a
4:      c ← [ ]; (s_a, f_a, v_a) ← get(a); (s_b, f_b, v_b) ← get(b)
5:      while (s_a < ∞) ∨ (s_b < ∞) do
6:          if s_a < s_b then
7:              s_c ← s_a; v_c ← v_a
8:              if s_b < f_a then f_c ← s_b; s_a ← s_b
9:              else f_c ← f_a; (s_a, f_a, v_a) ← get(a)
10:         else if s_a = s_b then
11:             s_c ← s_a; f_c ← min(f_a, f_b); v_c ← sAdd(v_a, v_b)
12:             s_a ← s_b ← f_c; f_d ← f_a
13:             if f_d ≤ f_b then (s_a, f_a, v_a) ← get(a)
14:             if f_b ≤ f_d then (s_b, f_b, v_b) ← get(b)
15:         else
16:             s_c ← s_b; v_c ← v_b
17:             if s_a < f_b then f_c ← s_a; s_b ← s_a
18:             else f_c ← f_b; (s_b, f_b, v_b) ← get(b)
19:         c.append((s_c, f_c, v_c))
20:     return standard(c)
```

Algoritem 5 Množenje časovnih količin.

```
1:  function prod(a, b)
2:      if length(a) · length(b) = 0 then return [ ]
3:      c ← [ ]; (s_a, f_a, v_a) ← get(a); (s_b, f_b, v_b) ← get(b)
4:      while (s_a < ∞) ∨ (s_b < ∞) do
5:          if f_a ≤ s_b then (s_a, f_a, v_a) ← get(a)
6:          else if f_b ≤ s_a then (s_b, f_b, v_b) ← get(b)
7:          else
8:              s_c ← max(s_a, s_b); f_c ← min(f_a, f_b); v_c ← sMul(v_a, v_b)
9:              c.append((s_c, f_c, v_c))
10:             if f_c = f_a then (s_a, f_a, v_a) ← get(a)
11:             if f_c = f_b then (s_b, f_b, v_b) ← get(b)
12:     return standard(c)
```



Slika 7.2: Seštevanje in množenje časovnih količin – rast prostorske zahtevnosti.

Naj bo $l_a = \mathit{length}(a)$ in $l_b = \mathit{length}(b)$. Če imata operaciji $\oplus$ in $\odot$ iz osnovnega polkolobarja konstantno časovno zahtevnost $\mathcal{O}(1)$, sta zahtevnosti seštevanja in množenja časovnih količin $\mathcal{O}(l_a + l_b)$. Primer na sliki 7.2 pokaže, da je vsota v najslabšem primeru skoraj štirikratne dolžine obeh argumentov in produkt skoraj dvakrat toliko dolg kot argumenta. Ker smo privzeli $\mathcal{T} = [t_{min}, t_{max}] \subset \overline{\mathbb{N}}$, je dolžina seznama, ki opisuje časovne količine, omejena z $L = t_{max} - t_{min}$. Torej sta časovni zahtevnosti operacij v enostavnem časovnem polkolobarju največ reda $\mathcal{O}(L)$.

V trenutni različici knjižnice TQ uporabljamo predstavitev omrežja $\mathcal{N}$ z vrednostno matriko $\mathbf{A} = [a_{uv}]_{u,v \in \mathcal{V}}$,

$$a_{uv} = \begin{cases} w(u,v), & (u,v) \in \mathcal{L}, \\ 0, & \text{sicer.} \end{cases}$$

Časovna količina $w(u,v)$ predstavlja uteži na povezavi (u,v). Vrednost 0 je nevtralni element enostavnega časovnega polkolobarja (časovna količina, ki je enaka 0 za vse $t \in \mathcal{T}$).

Natančnejši opis časovnega omrežja dobimo z uporabo časovnih vektorjev za opis lastnosti vozlišč.

7.1 Enostavne operacije

7.1.1 Skupna vrednost

Včasih bomo uporabljali *združeno* ali *skupno vrednost* dane časovne količine $a = \big((s_i, f_i, v_i)\big)_{i=1}^{k}$ iz enostavnega časovnega kombinatoričnega polkolobarja. Skupno

vrednost Σa definiramo kot

$$\Sigma a = \sum_{i=1}^{k} (f_i - s_i) \cdot v_i.$$

Izračunamo jo s funkcijo *total*. Na primer $\Sigma a = 23$ in $\Sigma b = 30$. Vedno velja $\Sigma a + \Sigma b = \Sigma(a + b)$.

7.1.2 Časovne stopnje

Produkt časovne matrike in časovnega vektorja definiramo enako kot za statične matrike in vektorje. Obstajata levi in desni produkt časovnega vektorja in časovne matrike.

Naj bo $\mathbf{A}$ časovna matrika velikosti $n \times m$, vektor x časovni vektor dolžine n in vektor y časovni vektor dolžine m. Levi produkt matrike $\mathbf{A}$ z vektorjem y je vektor $x = (y^T \odot \mathbf{A})^T$. Desni produkt matrike $\mathbf{A}$ z vektorjem x je vektor $y = \mathbf{A} \odot x$. Operaciji delujeta nad polkolobarjem časovnih matrik.

V knjižnici TQ smo obe množenji časovne matrike s časovnim vektorjem implementirali kot funkciji *MatVecMulL*(A, x) in *MatVecMulR*(A, x).

Za statično omrežje s sosednostno matriko $\mathbf{A}$ izračunamo vektorja vhodnih in izhodnih stopenj vozlišč z množenjem matrike in vektorja nad kombinatoričnim polkolobarjem. Enako velja za časovna omrežja

$$\mathrm{indeg} = \mathbf{e} \odot \mathbf{A} \qquad \text{in} \qquad \mathrm{outdeg} = \mathbf{A} \odot \mathbf{e},$$

kjer so $\mathbf{e}$ stolpec dolžine $n = |\mathcal{V}|$ z vsemi vrednostmi enakimi časovni količini 1, indeg časovni vektor vhodnih stopenj in outdeg časovni vektor izhodnih stopenj vozlišč. Operacija $\odot$ je množenje v enostavnem časovnem kombinatoričnem polkolobarju.

7.1.3 Dejavnost in privlačnost

Operaciji seštevanje in množenje časovnih količin lahko uporabimo za enostavno analizo časovnega omrežja. Naj bo $\mathbf{A} = [a_{uv}]_{u,v \in \mathcal{V}}$ vrednostna matrika časovnega omrežja in a_{uv} časovne količine, ki imajo pozitivne realne vrednosti. Pomen vrednosti a_{uv} je intenzivnost dejavnosti vozlišča u proti vozlišču v.

Definicija 7.1. *Dejavnost* skupine vozlišč $\mathcal{V}_1$ proti skupini vozlišč $\mathcal{V}_2$ definiramo kot

$$\mathrm{act}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{u \in \mathcal{V}_1} \sum_{v \in \mathcal{V}_2} a_{uv}.$$

Operacije opravljamo nad enostavnim časovnim kombinatoričnim polkolobarjem. *Dejavnost vozlišča* act(u) je enaka

$$\text{act}(u) = \text{act}(\{u\}, \mathcal{V} \setminus \{u\}) = \sum_{v \in \mathcal{V} \setminus \{u\}} a_{uv}.$$

Definicija 7.2. Da pri pomembnosti vozlišča $u \in \mathcal{V}$ upoštevamo tudi njegov položaj v omrežju, definiramo *koeficient privlačnosti* vozlišča u, att(u), kot

$$\text{att}(u) = \frac{1}{\Delta} \sum_{v \in \mathcal{V} \setminus \{u\}} \frac{a_{vu}}{\text{act}(v)}.$$

Ulomek $\frac{a_{vu}}{\text{act}(v)}$ predstavlja delež dejavnosti vozlišča v, ki si jo deli z vozliščem u. Iz $0 \leq \frac{a_{vu}}{\text{act}(v)} \leq 1$ izhaja, da velja

$$\sum_{v \in \mathcal{V} \setminus \{u\}} \frac{a_{vu}}{\text{act}(v)} \leq \Delta.$$

Torej velja $0 \leq \text{att}(u) \leq 1$ za vse $u \in \mathcal{V}$. Največjo vrednost privlačnosti dobijo vozlišča, ki so koren zvezde v neusmerjenem omrežju ali koren proti središču usmerjene zvezde v usmerjenem omrežju.

7.1.4 Zaprtje

Če je osnovni polkolobar $(A, \oplus, \odot, 0, 1)$ zaprt, je zaprt tudi polkolobar matrik nad njim. Za časovno količino a nad zaprtim polkolobarjem velja $T_{a^\star} = \mathcal{T}$. Prirejen Fletcherjev algoritem (algoritem 1, stran 10) za izračun (strogega) zaprtja v absorpcijskem časovnem polkolobarju je zapisan v algoritmu 6. Časovna zahtevnost algoritma 6 je $\mathcal{O}(n^3 \cdot L)$.

Algoritem 6 Zaprtje časovne matrike nad absorpcijskim časovnim polkolobarjem.

```
1: function MatClosure(R, strict = False)
2:     n ← nRows(R)
3:     C ← R
4:     for k = 1 to n do
5:         for u = 1 to n do
6:             for v = 1 to n do
7:                 C[u, v] ← sum(C[u, v], prod(C[u, k], C[k, v]))
8:         if ¬strict then C[k, k] ← sum(1, C[k, k])
9:     return C
```

7.1.5 Prisotnost vozlišč in povezav

V prejšnjih razdelkih smo opazovali časovna omrežja, v katerih so vozlišča ves čas prisotna. Prisotnost vozlišč ob različnih časih lahko opišemo s funkcijo $T : \mathcal{V} \to A_{\mathcal{T}}$,

$$T(u) = \big((s_i, f_i, 1)\big)_{i=1}^{k} \quad \text{za } u \in \mathcal{V},$$

ki pove, da je vozlišče u prisotno v omrežju v časovnih intervalih $[s_i, f_i)$, $i = 1, 2, \ldots, k$.

Najmanjšo potrebno prisotnost T_{Min}, ki ustreza pogoju usklajenosti (zveza (6.1), stran 45) za vozlišče $u \in \mathcal{V}$, določimo iz povezav časovnega omrežja s pomočjo enačbe

$$T_{Min}(u) = \bigcup_{\substack{\ell \in \mathcal{L}: \\ u \in \mathrm{ext}(\ell)}} \mathit{binary}(a_\ell).$$

Izraz ext(ℓ) označuje množico krajišč povezave ℓ in a_ℓ je časovna količina, ki pripada povezavi ℓ. Funkcija *binary* postavi vse neničelne vrednosti v časovni količini na 1. V knjižnici TQ prisotnost T_{Min} izračunamo s funkcijo *minTime*.

Prisotnost q lahko uporabimo za izračun podomrežja danega časovnega omrežja, ki je opisano z matriko $\mathbf{A}$. Podomrežje vsebuje samo vozlišča, ki so prisotna v q, in povezave, ki ustrezajo pogoju usklajenosti glede na q.

Najprej definiramo postopek *extract*$(p, a) = b$, kjer je p dvojiška časovna količina in a splošna časovna količina, kot

$$b(t) = a(t) \odot p(t) = \begin{cases} a(t), & t \in T_p \cap T_a, \\ 0, & \text{sicer.} \end{cases}$$

Naj bo $\mathbf{B}$ časovna matrika podomrežja, ki je določeno s prisotnostjo q. Vrednost v matriki $\mathbf{B}$, ki ustreza povezavi $\ell(u, v) \in \mathcal{L}$, določimo kot

$$b_\ell = \mathit{extract}(q(u) \cap q(v), a_\ell).$$

V knjižnici TQ je ta operacija zapisana v funkciji *MatExtract*(q, A).

7.2 Časovna razbitja, dosegljivost, šibka in krepka povezanost

Časovna razbitja opišemo s časovnimi količinami $a = \big((s_i, f_i, v_i)\big)_{i=1}^{k}$, kjer vrednosti $v_i \in \mathbb{N}$ pomenijo pripadnost razredom (skupinam) razbitja. Vrednost $v_i = j$ pomeni, da enota, ki jo opisuje količina a, pripada razredu j v časovnem intervalu $[s_i, f_i)$. S časovnimi razbitji opišemo spreminjanje komponent časovnega omrežja.

V časovnem omrežju, ki ga opisuje dvojiška časovna matrika $\mathbf{A}$, izračunamo zaprtje $\mathbf{A}^\star$ nad časovnim povezanostnim polkolobarjem. Matrika $\mathbf{A}^\star$ predstavlja matriko relacije dosegljivosti (če je vrednost v trenutku t na mestu uv enaka 1, je takrat vozlišče v dosegljivo iz vozlišča u, sicer pa ne).

Časovno matriko šibke povezanosti $\mathbf{W}$ dobimo kot

$$\mathbf{W} = (\mathbf{A} \cup \mathbf{A}^T)^\star \tag{7.1}$$

in časovno matriko krepke povezanosti $\mathbf{S}$ kot

$$\mathbf{S} = \mathbf{A}^\star \cap (\mathbf{A}^\star)^T, \tag{7.2}$$

kjer operaciji $\cup$ in $\cap$ ustrezata $\oplus$ in $\odot$ iz časovnega povezanostnega polkolobarja, ki ju izvedemo po komponentah.

Naj bo $\mathbf{R} = \overline{\mathbf{A}} = \mathbf{A} \odot \mathbf{A}^\star$ časovna matrika, ki opisuje relacijo stroge dosegljivosti v danem omrežju. Potem časovna vektorja *inReach* = *inDeg*$(\mathbf{R})$ in *outReach* = *outDeg*$(\mathbf{R})$ vsebujeta časovne količine, ki opisujejo število vozlišč v danem trenutku, iz katerih je dano vozlišče v dosegljivo (*inReach*$[v]$) oziroma so dosegljiva iz danega vozlišča v (funkcija *outReach*$[v]$).

7.2.1 Časovna šibka povezanost

Funkcija *weakConnMat*$(\mathbf{A})$ za dano časovno matriko omrežja $\mathbf{A}$ izračuna pripadajočo matriko šibke povezanosti $\mathbf{W}$ (enačba (7.1)). Vsaka časovna rezina $\mathcal{N}(t)$, $t \in \mathcal{T}$, matrike $\mathbf{W}$ je ekvivalenčna relacija, ki jo lahko strnjeno opišemo s pripadajočim časovnim razbitjem.

Časovno matriko ekvivalenčne relacije $\mathbf{E}$ spremenimo v pripadajoče časovno razbitje $\mathbf{p}$ tako, da upoštevamo dejstvo, da na vsakem časovnem intervalu ekvivalentna (v našem primeru šibko povezana) vozlišča dobijo enako vrednost na tem intervalu v produktu matrike z vektorjem. Vzamemo produkt nad kombinatoričnim polkolobarjem in izberemo vektor z $i-$tim elementom enakim 2^i. Potem ima vsak element produkta v dvojiškem zapisu enice na mestih, ki ustrezajo indeksom vozlišč v isti komponenti.

Postopek je implementiran kot funkcija *eqMat2Part*$(\mathbf{E})$, ki je opisana v algoritmu 7. Algoritem ima le teoretično vrednost. Uporabili bi ga lahko le na zelo majhnih omrežjih. V knjižnici TQ namesto vrednosti 2^i vzamemo π_i, kjer je π neka permutacija, in izračun nekajkrat ponovimo ter primerjamo rezultate.

Ko dobimo časovno razbitje, razrede preoštevilčimo z zaporednimi naravnimi števili s pomočjo funkcije $renumPart(p)$ (algoritem 8).

Algoritem 7 Časovno ekvivalenčno relacijo zapiši kot časovno razbitje.

```
1: function eqMat2Part(E)
2:     SetSemiring(combinatorial)
3:     v ← [[(0, ∞, 2^i)] for i = 1 to nRows(E)]
4:     p ← MatVecMulR(E, v)
5:     return renumPart(p)
```

Algoritem 8 Preoštevilči razrede časovnega razbitja.

```
1: function renumPart(p)
2:     C ← { }; q = [ ]
3:     for a ∈ p do
4:         r ← [ ]
5:         for (s_a, f_a, c_a) ∈ a do
6:             if c_a ∉ C then C[c_a] ← 1 + length(C)
7:             r.append((s_a, f_a, C[c_a]))
8:         q.append(r)
9:     return q
```

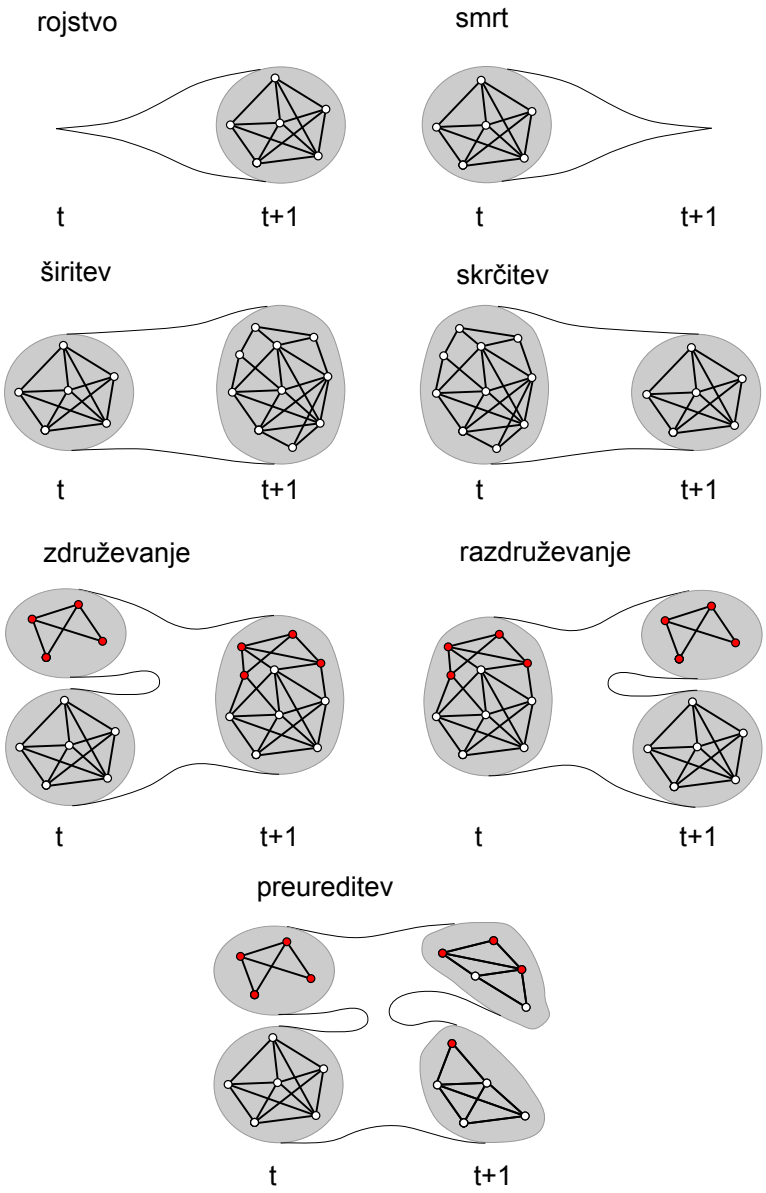
7.2.2 Časovna krepka povezanost

Postopek *strongConnMat*($\mathbf{A}$) za dano časovno matriko omrežja $\mathbf{A}$ določi pripadajočo časovno matriko krepke povezanosti $\mathbf{S}$ (enačba (7.2)). Ustrezen produkt dvojiških časovnih matrik $\mathbf{A}$ in $\mathbf{B}$ izračunamo s funkcijo *MatInter*($\mathbf{A}, \mathbf{B}$). Za določitev razbitja na krepke komponente spet uporabimo funkcijo *eqMat2Part* na matriki krepke povezanosti $\mathbf{S}$.

Časovna zahtevnost algoritmov za izračun razbitja na časovne šibko in krepko povezane komponente je $\mathcal{O}(n^3 \cdot L)$. V knjižnici TQ obe matriki in razbitji izračunamo s pomočjo strogega zaprtja nad enostavnim časovnim povezanostnim polkolobarjem.

7.2.3 Druga časovna razbitja iz ekvivalenčnih relacij

Postopek, opisan v razdelku 7.2.1, je uporaben za poljubno časovno ekvivalenčno relacijo na danem omrežju. Predela jo v pripadajoče časovno razbitje. V nadaljnjem raziskovanju nameravamo poskusiti razviti postopke za tolmačenje razvoja skupin (razredov), določenih s časovnim razbitjem glede na (izpopolnjeno) klasifikacijo sprememb iz [68, 69], predstavljeno na sliki 7.3.



Slika 7.3: Spremembe skupin.

Poglavje 8

Pomembnosti v časovnih omrežjih brez potovalnih in čakalnih časov

O časovnih stopnjah vozlišč in o tem, kako jih izračunamo, smo govorili že v razdelku 7.1.2.

8.1 Časovna nakopičenost

Časovno nakopičenost definiramo na enak način, kot smo to storili v razdelku 3.2 na strani 22. Ker se omrežje s časom spreminja, se spreminjata tudi število sosedov in maksimalna stopnja.

Štetje trikotnikov v časovnem omrežju ostane enako kot v statičnem primeru in je zapisano v algoritmu 9 v vrsticah 6–8. Najprej zagotovimo, da so na diagonali časovne sosednostne matrike ničle (torej v omrežju ni zank), potem pa izračunamo matriko $\mathbf{S}$.

Ukaz *VecConst*(n, v) ustvari vektor velikosti n s časovnimi količinami, enakimi v. Funkcija *MatBin* vse vrednosti v trojicah časovne matrike postavi na 1 (iz vrednostne naredi sosednostno matriko). S funkcijo *MatSetDiag*$(\mathbf{A}, c)$ vse diagonalne elemente v matriki $\mathbf{A}$ spremenimo v vrednost c. Funkcija *MatSym*$(\mathbf{A})$ izračuna časovno matriko $\mathbf{S} = \mathbf{A} \oplus \mathbf{T}$. Funkciji *VecSum* in *VecProd* po komponentah seštejeta / zmnožita časovna vektorja. Podobno vlogo ima tudi *VecInv*(a) = $[$*invert*$(a_i),\ i = 1, \ldots n]$. Funkcija *invert*$(a) = [(s, f, 1/v)$ **for** $(s, f, v) \in a]$ deluje v enostavnem časovnem kombinatoričnem polkolobarju. Funkcija *MatProd* izračuna produkt časovnih matrik. Ker potrebujemo le diagonalne vrednosti matrike $\mathbf{SAS}$, smo napisali še posebno funkcijo *MatProdDiag*, ki izračuna le diagonalne elemente produkta matrik.

Za izračun nakopičenosti vozlišč je treba rezultat normalizirati. Število sosedov vozlišča v dobimo z izračunom stopnje v vrstici 9 (ne upoštevamo usmerje-

nosti povezav). Maksimalno število sosedov Δ lahko razumemo na dva načina: lahko izberemo maksimalno število sosedov v določeni časovni točki (*type* = 2) ali maksimalno število sosedov za celotno življenjsko dobo omrežja (*type* = 3). Če izberemo *type* = 1, algoritem izračuna klasično nakopičenost vozlišč. Pri izračunu maksimalne časovne stopnje Δ smo uporabili seštevanje nad maxmin polkolobarjem $(\overline{\mathbb{R}}, \max, \min, -\infty, \infty)$.

Časovna zahtevnost algoritma 9 je $O(n^3 \cdot L)$.

Algoritem 9 Časovna nakopičenost.

function *clusCoef*$(A, \textit{type} = 1)$
type = 1 - standard clustering coefficient
type = 2 - corrected clustering coefficient / temporal degMax
type = 3 - corrected clustering coefficient / overall degMax
    SetSemiring(*combinatorial*)
    $n \leftarrow$ *nRows*(A); $ve \leftarrow$ *VecConst*$(n, [(0, \infty, -1)])$
    $B \leftarrow$ *MatSetDiag*(*MatBin*$(A), \mathbf{0}$)
    $S \leftarrow$ *MatBin*(*MatSym*(B))
    deg $\leftarrow$ *MatVecMulR*$(S,$ *VecConst*$(n, \mathbf{1}))$
    if *type* = 1 **then**
        fac $\leftarrow$ *VecProd*(*deg*, *VecSum*(*deg*, ve))
    else
        SetSemiring(*maxmin*); $\delta \leftarrow \mathbf{0}$
        for $d \in$ *deg* **do** $\delta \leftarrow$ *sum*(δ, d)
        if *type* = 3 **then**
            $\Delta \leftarrow \max([v$ **for** $(s, f, v) \in \delta])$
            $\delta \leftarrow [(0, \infty, \Delta)]$
        SetSemiring(*combinatorial*)
        degm $\leftarrow$ *VecSum*(*deg*, ve); *fac* $\leftarrow \mathbf{0}$
        for $d \in$ *degm* **do** *fac.append*(*prod*(δ, d))
    tri $\leftarrow$ *MatProdDiag*(*MatProd*$(S, B), S)$
    return *VecProd*(*VecInv*(*fac*), *tri*)

8.2 Časovna dostopnost

Časovna dostopnost je definirana kot v razdelku 3.3 na strani 23. Kot v statičnih omrežjih tudi v časovnih omrežjih za izračun dostopnosti najprej izračunamo matriko $\mathbf{D} = [d_{uv}]_{u,v \in \mathcal{V}}$ dolžin najkrajših poti d_{uv} med vozliščema u in v. Izračunamo jo kot zaprtje časovne matrike $\mathbf{A}$ nad enostavnim časovnim polkolobarjem

najkrajših poti. Vrednosti časovnih količin v matriki $\mathbf{A}$ so lahko poljubna nenegativna realna števila.

Vektor dostopnosti vozlišč izračunamo tako, da najprej seštejemo časovne razdalje do drugih vozlišč nad enostavnim časovnim kombinatoričnim polkolobarjem. Postopek za izračun časovne dostopnosti je zapisan v algoritmu 10, ki ima časovno zahtevnost $\mathcal{O}(n^3 \cdot L)$.

Algoritem 10 Časovna dostopnost.

```
1:  function closeness(A, type = 2)
2:  # type: 1 - output, 2 - all, 3 - input
3:      s ← startTime(A); f ← finishTime(A); n ← nRows(A)
4:      SetSemiring(path)
5:      D ← MatClosure(A, strict = True)
6:      SetSemiring(combinatorial)
7:      k ← (2 − |type − 2|) · (n − 1); fac ← [(0, ∞, k)]
8:      for v = 1 to n do
9:          d ← 0
10:         for u = 1 to n do
11:             if u ≠ v then
12:                 if type < 3 then d ← sum(d, D[v, u])
13:                 if type > 1 then d ← sum(d, D[u, v])
14:         cl[v] ← prod(fac, invert(d))
15:     return cl
```

8.3 Časovna vmesnost

Časovno vmesnost definiramo na enak način kot v statičnih omrežjih (razdelek 3.4, stran 25),

$$b(v) = \frac{1}{(n-1)(n-2)} \sum_{\substack{u,w \in \mathcal{V} \\ |\{v,u,w\}|=3}} \frac{n_{uw}(v)}{n_{uw}}.$$

Tako kot v statičnem omrežju tudi v časovnem omrežju najprej izračunamo zaprtje $\mathbf{C} = [(d_{uv}, n_{uv})]_{u,v \in \mathcal{V}}$ prirejene časovne matrike $\mathbf{G}$ omrežja nad enostavnim časovnim geodezičnim polkolobarjem. Prirejeno časovno matriko $\mathbf{G} = [(d,n)_{uv}]_{u,v \in \mathcal{V}}$ dobimo iz časovne vrednostne matrike $\mathbf{A}$ tako, da vsak neničelni element matrike $\mathbf{A}$ zamenjamo s časovno količino, ki ima vrednost $(1,1)$ za vse čase, ob katerih je izvirna časovna količina različna od 0:

$$(d,n)_{uv}(t) = \begin{cases} (1,1), & \ell(u,v) \in \mathcal{L},\ t \in T_\ell, \\ 0 & \text{sicer.} \end{cases}$$

V časovnem omrežju sta razdalja d in število najkrajših poti n časovni količini.

S funkcijo *betweenness*, zapisano v algoritmu 11, transformiramo matriko omrežja $\mathbf{A}$ v matriko $\mathbf{G}$ in izračunamo njeno strogo zaprtje $\mathbf{C}$ nad enostavnim časovnim geodezičnim polkolobarjem. V nadaljevanju uporabljamo operacije enostavnega časovnega kombinatoričnega polkolobarja in izračunamo časovni vektor vmesnosti b. Funkcija *between* (algoritem 12) iz časovnih količin $d[u, v]$ in $n[u, v]$ izračuna vrednost, ki ustreza stavku

if $d[u, w] = d[u, v] + d[v, w]$ **then** $n[u, v] \cdot n[v, w]/n[u, w]$.

Spet uporabimo zlivanje seznamov. Časovna zahtevnost algoritma 11 je $\mathcal{O}(n^3 \cdot L)$.

Algoritem 11 Časovna vmesnost.

```
1:  function betweenness(A)
2:      n ← nRows(A); G ← MatSetVal(A, (1, 1))
3:      SetSemiring(geodetic)
4:      C ← MatClosure(G, strict = True)
5:      SetSemiring(combinatorial)
6:      fac ← [(0, ∞, 1/(n − 1)/(n − 2))]
7:      for v = 1 to n do
8:          r ← 0
9:          for u = 1 to n do
10:             for w = 1 to n do
11:                 if (C[u, w] ≠ [ ]) ∧ (u ≠ w) ∧ (u ≠ v) ∧ (v ≠ w) then
12:                     r ← sum(r, between(C[u, v], C[v, w], C[u, w]))
13:         b[v] ← prod(r, fac)
14:     return b
```

8.4 Rekurzivne mere pomembnosti

8.4.1 Pomembnost na osnovi lastnih vektorjev

Najenostavnejša numerična metoda za izračun dominantnega lastnega vektorja je potenčna metoda (poglavje 4, stran 35).

Implementacija potenčne metode za časovne matrike je zapisana kot funkcija *eigTemp* v algoritmu 13. Algoritem vrne približek za dominantni lastni vektor x, približek za dominantno lastno vrednost ev in parameter *convergence,* ki pove, ali se je algoritem končal, ker je dosegel želeno natančnost (v tem primeru ima vrednost **True**), ali ne (ima vrednost **False**). Funkcija *MatVecRight*(A, x) izračuna produkt $\mathbf{A}x$ za časovno matriko $\mathbf{A}$ in časovni vektor x, funkcija *normalize*(x)

Algoritem 12 Operacija zlivanja za izračun časovne vmesnosti.

```
1:  function between(a, b, c)
2:      if length(a) = 0 then return [ ]
3:      if length(b) = 0 then return [ ]
4:      if length(c) = 0 then return [ ]
5:      r ← [ ]
6:      (s_a, f_a, v_a) ← get(a); (s_b, f_b, v_b) ← get(b); (s_c, f_c, v_c) ← get(c)
7:      if isTuple(v_a) then (d_a, c_a) ← v_a
8:      if isTuple(v_b) then (d_b, c_b) ← v_b
9:      if isTuple(v_c) then (d_c, c_c) ← v_c
10:     while (s_a < ∞) ∨ (s_b < ∞) ∨ (s_c < ∞) do
11:         s_r ← max(s_a, s_b, s_c); f_r ← min(f_a, f_b, f_c)
12:         if f_a ≤ s_r then
13:             (s_a, f_a, v_a) ← get(a)
14:             if isTuple(v_a) then (d_a, c_a) ← v_a
15:         else if f_b ≤ s_r then
16:             (s_b, f_b, v_b) ← get(b)
17:             if isTuple(v_b) then (d_b, c_b) ← v_b
18:         else if f_c ≤ s_r then
19:             (s_c, f_c, v_c) ← get(c)
20:             if isTuple(v_c) then (d_c, c_c) ← v_c
21:         else
22:             if d_a + d_b = d_c then r.append((s_r, f_r, c_a · c_b/c_c))
23:             if f_r = f_a then
24:                 (s_a, f_a, v_a) ← get(a)
25:                 if isTuple(v_a) then (d_a, c_a) ← v_a
26:             if f_r = f_b then
27:                 (s_b, f_b, v_b) ← get(b)
28:                 if isTuple(v_b) then (d_b, c_b) ← v_b
29:             if f_r = f_c then
30:                 (s_c, f_c, v_c) ← get(c)
31:                 if isTuple(v_c) then (d_c, c_c) ← v_c
32:     return standard(r)
```

implementira časovno različico izraza $x/\|x\|_2$. Funkcija *test_dif*(x, y) poišče maksimalno vrednost (skozi čas) norme $\|x-y\|_2$, ki jo v vrstici 6 primerjamo z dopustno napako. Če smo dosegli želeno natančnost *tol,* izstopimo iz zanke. Približek za dominantno lastno vrednost izračunamo izven zanke, da se izognemo večkratnemu množenju z matriko. Funkcija *scalProd*(x, y) izračuna skalarni produkt dveh časovnih vektorjev x in y. Vrstica 9 predstavlja časovno različico računa $\lambda = x^T \mathbf{A} x$.

Algoritem 13 Časovna potenčna metoda.

```
1:  function eigTemp(A, x, tol = 10^-6, maxIter = 100)
2:      convergence ← False
3:      for i = 0 to maxIter do
4:          x_old ← x
5:          x ← normalize(MatVecRight(A, x))
6:          if test_dif(x, x_old) < tol then
7:              convergence ← True
8:              break
9:      ev ← scalProd(x, MatVecRight(A, x))
10:     return (x, ev, convergence)   ▷ x je približek za dominantni lastni vektor
                                        in ev je približek za dominantno lastno vrednost
```

Izrek 8.1. *Časovna potenčna metoda konvergira natanko tedaj, ko statična potenčna metoda konvergira za vsako matriko* $\mathbf{A}(t),\ t \in \mathcal{T}$, *časovne rezine omrežja.*

Dokaz. Seštevanje in množenje časovnih količin v enostavnih časovnih polkolobarjih ustrezata operacijam funkcij po točkah. Za vsak $t \in \mathcal{T}$ vrednosti časovnih količin opisujejo statično omrežje – časovno rezino $\mathcal{N}(t)$ časovnega omrežja $\mathcal{N}$. Zato se pogoji za konvergenco statičnih matrik po točkah prenesejo na časovne matrike. Funkcija *test_dif* v algoritmu preverja maksimalno razliko za vse čase. Ker je življenjska doba končna množica, iz konvergence po točkah izhaja tudi, da ta maksimum konvergira k 0. □

Posledica 8.1. *Naj bosta* $\lambda_1(t)$ *in* $\lambda_2(t)$ *lastni vrednosti z največjima absolutnima vrednostma matrike* $\mathbf{A}(t)$ *časovne rezine omrežja. Red konvergence je enak* $\mu = \max\{|\lambda_2(t)/\lambda_1(t)|,\ t \in \mathcal{T}\}$. *Časovna potenčna metoda konvergira za vrednosti* $\mu < 1$ *in konvergenca je počasnejša za vrednosti* μ *blizu* 1.

Dokaz. Časovna potenčna metoda po trditvi 4.1, stran 36, konvergira v časovni točki t, kadar je razmerje $|\lambda_2(t)/\lambda_1(t)| < 1$. Red konvergence izračunamo po točkah in poiščemo maksimum končne množice. □

Opozorimo, da iz dokaza izreka 8.1 izhaja, da časovna potenčna metoda lahko konvergira za neke čase $t \in \mathcal{T}$ in divergira za druge. Za izhod iz zanke podamo dva pogoja. Prvi pogoj je želena natančnost *tol*, ki ima privzeto vrednost enako 10^{-6}, drugi pogoj pa je največje število iteracij *maxIter* s privzeto vrednostjo 100.

Za izračun vhodne lastne pomembnosti (funkcija *inEig*) in izhodne lastne pomembnosti (funkcija *outEig*) uporabimo časovno potenčno metodo. Oba postopka sta zapisana v algoritmu 14. Funkcija *MatTrans*(A) izračuna transponirano matriko časovne matrike $\mathbf{A}$. Funkcija *VecConst*(n) ustvari časovni vektor razsežnosti n, ki ima vse komponente enake enoti časovnega polkolobarja. Funkcija *numInv*(a) zamenja vrednosti časovne količine z njihovimi inverzi in ohrani časovne intervale. Funkcja *numVecProd*(a, x) izračuna produkt časovne količine a in časovnega vektorja x. V vrstici 2 (ali 6) izračunamo približek za dominantni lastni par časovne vrednostne matrike $\mathbf{A}^T$ (ali $\mathbf{A}$) z začetnim približkom (časovnih) enic. V vrstici 3 (ali 7) vektor delimo z lastno vrednostjo.

Algoritem 14 Časovna lastna pomembnost.

1: **function** *inEig*(A)
2: $(x, ev, conv) \leftarrow$ *eigTemp*(*MatTrans*(A), *VecConst*$(len(A))$)
3: $x \leftarrow$ *numVecProd*(*numInv*$(ev), x)$
4: **return** $(x, conv)$
5: **function** *outEig*(A)
6: $(x, ev, conv) \leftarrow$ *eigTemp*$(A,$ *VecConst*$(len(A)))$
7: $x \leftarrow$ *numVecProd*(*numInv*$(ev), x)$
8: **return** $(x, conv)$

8.4.2 Katzova pomembnost

Za izračun rešitve sistema linearnih enačb smo uporabili Jacobijevo metodo (poglavje 4, stran 35). Implementacija Jacobijeve metode za reševanje sistema $\mathbf{A}x = b$, kjer sta $\mathbf{A}$ in b časovna matrika in časovni vektor, je zapisana v algoritmu 15 kot funkcija *jacobi*. Podamo dva pogoja za izhod iz zanke: kadar dosežemo želeno natančnost *tol* rešitve ali kadar izvedemo določeno število korakov iteracije *maxIter*. V vrstici 2 izračunamo inverz diagonalne matrike $\mathbf{D}$, v vrstici 3 pa izračunamo matriko $\mathbf{B} = -(\mathbf{L} + \mathbf{U})$ tako, da diagonalne elemente matrike $\mathbf{A}$ postavimo na vrednost 0 (kar v naši predstavitvi pomeni, da jih izpustimo) in negiramo vrednosti ostalih elementov. V vrstici 6 izračunamo naslednji približek za rešitev enačbe kot časovno različico stavka $x_n = invD(Bx + b)$. V vrsticah 7–9 preverimo, ali smo dosegli želeno natančnost, in zaključumo računanje, če smo jo.

Algoritem 15 Časovna Jacobijeva iteracija.

```
1: function jacobi(A, b, x, tol = 10^-6, maxIter = 100)  ▷ x je začetni približek za rešitev sistema enačb
2:     invD ← MatSetDiagVec(vecInv(diag(A)))
3:     B ← MatMinus(MatSetDiagZero(A))
4:     convergence ← False
5:     for i = 1 to maxIter do
6:         xn ← MatVecRight(invD, VecSum(MatVecRight(B, x), b))
7:         if test_dif(x, xn) < tol then
8:             convergence ← True
9:             break
10:        x ← xn
11:    return (xn, convergence)
```

Izrek 8.2. *Časovna Jacobijeva iteracija konvergira natanko tedaj, ko statična Jacobijeva iteracija konvergira za vse matrike* $\mathbf{A}(t),\ t \in \mathcal{T}$, *časovnih rezin omrežja.*

Dokaz. Razlaga je enaka kot v dokazu izreka 8.1, stran 76. Seštevanje in množenje časovnih količin sta definirana po točkah, zato operacije na časovnih matrikah ustrezajo operacijam na zaporedju statičnih matrik. Pogoji za konvergenco statičnih matrik se prenesejo na časovne matrike. □

Posledica 8.2. *Če so vse matrike časovnih rezin omrežja strogo diagonalno dominantne, časovna Jacobijeva iteracija konvergira za vsak časovni vektor, ki je začetni približek za rešitev linearnega sistema* $\mathbf{A}x = b$.

Dokaz. Konvergenca statičnih matrik sledi iz trditve 4.3, stran 37. □

Podobno kot časovna potenčna metoda tudi časovna Jacobijeva iteracija lahko konvergira za nekatere čase $t \in \mathcal{T}$ in divergira za druge. Parameter konvergence ima vrednost **True**, če konvergira za vse čase, za katere so vrednosti časovnih količin neničelne.

Algoritem za izračun Katzove pomembnosti (razdelek 3.5.2, stran 29) za časovna omrežja je zapisan v algoritmu 16. Jacobijeva metoda konvergira za strogo diagonalno dominantne matrike (poglavje 4, stran 35). V algoritmu za izračun Katzove pomembnosti vhodni parameter a, ki ustreza α iz definicije Katzove pomembnosti, lahko izpustimo.

Posledica 8.3. *Kadar parameter* a *ni podan, ga algoritem 16 izračuna na tak način, da zagotovi konvergenco časovne Jacobijeve iteracije.*

Dokaz. V vrsticah 4–9 parameter a izračunamo iz maksimuma vseh stolpičnih vsot (vhodnih stopenj) matrike $\mathbf{A}$, tako da je $1/a$ malo večje od tega maksimuma. Potem je matrika v enačbi (3.4, stran 29) strogo diagonalno dominantna in iteracija konvergira po posledici 8.2. □

V vrsticah 10–13 izračunamo $\mathbf{B} = \frac{1}{a}\mathbf{I} - \mathbf{A}^T$ in v vrstici 14 izračunamo približek za rešitev enačbe $\mathbf{B}t = d$ z začetnim približkom samih časovnih enic. V vrsticah 15–17 normaliziramo rešitev s primernim m.

Algoritem 16 Časovna Katzova pomembnost.

```
 1: function katz(A, a = Null)
 2:     n ← len(A)
 3:     d ← MatVecLeft(A, VecConst(n))          ▷ Časovne vhodne stopnje
 4:     if a = Null then                        ▷ Izračunaj a, če ni podan.
 5:         max ← 0
 6:         for i = 1 to len(d) do
 7:             if VecMax(d[i]) > max then
 8:                 max ← VecMax(d[i])
 9:         a ← 0.999/max
10:     B ← n × n časovna matrika
11:     for i = 1 to n do
12:         B[i][i] ← [(1, ∞, 1/a)]
13:     B ← MatDiff(B, MatTrans(A))
14:     (t, conv) ← jacobi(B, d, VecConst(n))
15:     m ← (n − 1)! a^(n−1) e^(1/a)
16:     m ← [(1, ∞, 1/m)]
17:     return (numVecProd(m, t), conv)
```

8.4.3 Bonacichevi pomembnosti α in (α, β)

Časovna različica Bonacicheve pomembnosti α (razdelek 3.5.3, stran 30) je zapisana v algoritmu 17. Parameter a v algoritmu ustreza parametru α iz definicije. Če vektor zunanjih statusov s ni podan, ga v vrstici 3 določimo kot vektor časovnih enic. Rešitev linearnega sistema izračunamo s pomočjo časovne Jacobijeve iteracije (algoritem 15).

Časovna različica Bonacicheve pomembnosti (α, β) je zapisana kot funkcija *bonacich* in je opisana v algoritmu 18. Parametra a in b v algoritmu ustrezata parametroma α in β iz definicije. Uvedemo pomožno spremenljivko $normB$, ki pove, ali je treba rešitev normalizirati, kot smo opisali na koncu razdelka 3.5.3 na

Algoritem 17 Časovna Bonacicheva pomembnost α.

1: **function** *alpha*(A, a, s = **Null**)
2: **if** s = **Null then**
3: $s \leftarrow$ *VecConst*($len(A)$)
4: **return** *jacobi*(*MatSum*(*MatEye*($len(A)$), *numMatProd*([$(1, \infty, -a)$], *MatTrans*(A))), s, *VecConst*($len(A)$))

strani 30 za primer, ko α ni podana (to storimo v vrstici 10). Časovno različico stavka $b_1 = a\mathbf{A}\mathbf{e}$ izračunamo v vrstici 6 in časovno različico stavka $\mathbf{B} = \mathbf{I} - b\mathbf{A}$ izračunamo v vrstici 7. Za izračun rešitve enačbe uporabimo časovno Jacobijevo iteracijo z začetnim približkom samih časovnih enic.

Algoritem 18 Časovna Bonacicheva pomembnost (α, β).

1: **function** *bonacich*(A, b, a = **Null**)
2: $normB \leftarrow$ **False**
3: **if** a = **Null then**
4: $a \leftarrow 1$
5: $normB \leftarrow$ **True**
6: $b1 \leftarrow$ *numVecProd*([$(1, \infty, a)$], *MatVecRight*(A, *VecConst*($len(A)$)))
7: $B \leftarrow$ *MatSum*(*MatEye*($len(A)$), *numMatProd*([$(1, \infty, -b)$], A))
8: $(x, conv) \leftarrow$ *jacobi*($B, b1$, *VecConst*($len(A)$))
9: **if** $normB$ **then**
10: $x \leftarrow$ *numVecProd*([$(1, \infty, \sqrt{len(A)})$], *normalize*($x$))
11: **return** $(x, conv)$

8.4.4 Kazala in viri (HITS)

Različica algoritma HITS za izračun kazal in virov (razdelek 3.5.4, stran 31) za časovna omrežja je zapisana v algoritmu 20.

Za izračun lastnega sistema matrike $\mathbf{A}^T\mathbf{A}$ smo uporabili učinkovitejši algoritem, v katerem ni treba računati produkta $\mathbf{A}^T\mathbf{A}$. Postopek je zapisan v algoritmu 19 kot funkcija *singTemp*. Podoben je algoritmu 13, razlika je v vrsticah 5 in 9, kjer z matriko $\mathbf{A}^T$ množimo z uporabo funkcije *MatTransVecRight*.

Vrednosti pomembnosti vozlišč kot kazala in kot vira izračunamo v algoritmu 20 tako, da najprej v vrstici 2 izračunamo lasten sistem matrike $\mathbf{A}^T\mathbf{A}$. Uporabimo začetni približek časovnih enic. Tako dobimo vrednosti virov y. V vrstici 3 iz vektorja y izračunamo vrednosti kazal, ki jih shranimo v vektor x. V vrsticah 4–6 jih umerimo glede na lastno vrednost.

Algoritem 19 Potenčna metoda za izračun dominantnega lastnega para $\mathbf{A}^T\mathbf{A}$.

1: **function** *singTemp*($A, x,$ *tol* = 10^{-6}, *maxIter* = 100)
2: *convergence* ← **False**
3: **for** $i = 0$ **to** $maxIter$ **do**
4: $x\_old \leftarrow x$
5: $x \leftarrow$ *normalize*(*MatTransVecRight*(A, *MatVecRight*(A, x)))
6: **if** *test_dif*(x, x_old) < *tol* **then**
7: *convergence* ← **True**
8: **break**
9: $ev \leftarrow$ *scalProd*(x, *MatTransVecRight*(A, *MatVecRight*(A, x)))
10: **return** ($x, ev,$ *convergence*)

Algoritem 20 Kazala in viri (algoritem HITS).

1: **function** *hits*(A)
2: $(y, evy, conv) \leftarrow$ *singTemp*(A, *VecConst*($len(A)$))
3: $x \leftarrow$ *normalize*(*MatVecRight*(A, y))
4: $evInv \leftarrow$ *numInv*(evy)
5: $y \leftarrow$ *numVecProd*($evInv, y$)
6: $x \leftarrow$ *numVecProd*($evInv, x$)
7: **return** (x, y) ▷ x je vektor vrednosti kazal in y je vektor vrednosti virov

8.4.5 PageRank

Različica izračuna pomembnosti pageRank (razdelek 3.5.5, stran 33) za časovna omrežja je zapisana v algoritmu 21. V vrstici 4 izračunamo vektor izhodnih stopenj in v vrsticah 5–7 izračunamo matriko $\mathbf{S}$. V vrstici 5 uporabimo funkcijo *vecInvPR,* ki vrača vektor inverzov vrednosti (kadar je stopnja enaka 0, ima ustrezna komponenta vrednosti enake $\frac{1}{n}$ za vse čase $t \in \mathcal{T}$) in matriko, ki ima v vrsticah, ki ustrezajo vozliščem z ničelno izhodno stopnjo, časovne enote. Vrstica 6 spremeni originalno matriko tako, da so v vrsticah, ki ustrezajo vozliščem z ničelno izhodno stopnjo, same enice. V vrstici 7 normaliziramo matriko glede na izhodne stopnje vozlišč. To storimo s pomočjo funkcije *DiagMatProd*(x, A) namesto z množenjem matrik, da je algoritem učinkovitejši. Ta funkcija izračuna produkt diagonalne matrike, ki ima na diagonali vektor x, in matrike A. V vrstici 8 izračunamo matriko $\mathbf{M}$, pri čemer s funkcijo *constantMat* ustvarimo matriko z vsemi vrednostmi, enakimi $\frac{1}{n}$ za vse $t \in \mathcal{T}$. Levi lastni vektor matrike $\mathbf{M}$ izračunamo kot desni lastni vektor matrike $\mathbf{M}^T$. Na koncu rezultat normaliziramo. Funkcija *norm1* normalizira vektor glede na prvo normo, kar pomeni, da je vsota komponent vektorja enaka 1 za vse čase.

Algoritem 21 Časovni algoritem pageRank.

```
1: function pageRank(A, q = 0.15)
2:     n ← len(A)
3:     S ← n × n časovna matrika
4:     s ← MatVecRight(A, VecConst(n))                ▷ vektor izhodnih stopenj
5:     (S, s) ← vecInvPR(S, s)
6:     S ← MatSum(A, S)
7:     S ← DiagMatProd(s, S)
8:     M ← MatSum(numMatProd([(1, ∞, 1 − q)], S),
               numMatProd([(1, ∞, q)], constantMat(n, [(1, ∞, 1/n)])))
9:     (x, ev, conv) ← eigTemp(MatTrans(M), VecConst(n))
10:    return norm1(x)
```

Posledica 8.4. *Časovni algoritem pageRank vedno konvergira.*

Dokaz. Matrika $\mathbf{M}$ iz enačbe (3.9), stran 33, je pozitivna in ima po izreku 1.1, stran 5, enolično lastno vrednost, ki ima največjo absolutno vrednost. Torej po izreku 8.1, stran 76, časovna potenčna metoda konvergira. □

8.4.6 Nekaj besed o časovni zahtevnosti algoritmov za računanje rekurzivnih mer pomembnosti v časovnih omrežjih

Naj n označuje število vozlišč v omrežju, m število povezav in k število iteracij iterativnih algoritmov (algoritmi 13, 15 in 19). Ker smo predpostavili, da je $\mathcal{T} \subseteq \overline{\mathbb{N}}$, je dolžina časovnih količin, ki opisujejo omrežje, omejena z življenjsko dobo omrežja, ki jo označimo z L. Operaciji v osnovnem polkolobarju sta običajni operaciji v premični piki in zahtevata konstanten čas, tj. $\mathcal{O}(1)$.

V začetku poglavja 7 smo pokazali, da seštevanje in množenje časovnih količin zahtevata največ $\mathcal{O}(L)$ časa. Zato je časovna zahtevnost množenja dveh časovnih vektorjev enaka $\mathcal{O}(nL)$ in časovna zahtevnost množenja časovne matrike s časovnim vektorjem $\mathcal{O}(n^2L)$. Časovna zahtevnost množenja dveh časovnih matrik je $\mathcal{O}(n^3L)$.

Od tod sledi, da imajo vsi algoritmi za izračun rekurzivnih mer pomembnosti v časovnih omrežjih časovno zahtevnost največ $\mathcal{O}(kn^2L)$.

Časovna zahtevnost algoritma 13 sledi iz zahtevnosti operacij v časovnem polkolobarju. Funkciji v algoritmu 14 imata enako zahtevnost, saj je funkcija *eigTemp* glavni del izračuna. To velja tudi za algoritem 21. Produkt matrik v vrstici 7 bi imel časovno zahtevnost $\mathcal{O}(n^3L)$, če ne bi posebej implementirali produkta z diagonalno matriko. Tudi algoritem 15 ima časovno zahtevnost enako $\mathcal{O}(kn^2L)$, največji del prinese vrstica 6. Funkcijo *jacobi* uporabimo v vrstici 14

algoritma 16, v vrstici 4 algoritma 17 in v vrstici 8 algoritma 18. V vseh primerih funkcija *jacobi* predstavlja največji del izračunov, torej imajo tudi ti algoritmi enako časovno zahtevnost. Izračun lastnih vektorjev in lastnih vrednosti matrike $\mathbf{A}^T\mathbf{A}$ v algoritmu 19 ima z našo implementacijo enako zahtevnost. Če bi zmnožili matriki in izračunali lastne vrednosti produkta, bi časovna zahtevnost narasla na $\mathcal{O}(kn^3L)$. Rezultat te funkcije je glavni del algoritma 20, ki ima zato enako časovno zahtevnost.

8.5 Mere pomembnosti in določanje skupin v časovnih omrežjih

Izbrana časovna mera pomembnosti nam daje vpogled v spreminjanje vloge posameznih vozlišč v omrežju. Ker je v večjih omrežjih pregled vseh vozlišč prezapleten, se običajno omejimo na skupino nekaj najpomembnejših ali za nas *zanimivih* vozlišč. Pri tem imamo več možnosti:

- skupina $\mathcal{C} \subseteq \mathcal{V}$ je določena vnaprej in časovno nespremenljiva; to smo uporabili v razdelku 9.3.1;
- skupino $\mathcal{C} \subseteq \mathcal{V}$ sestavlja k v celotnem obdobju najpomembnejših vozlišč; to smo uporabili v razdelku 9.2;
- časovno skupino $\mathcal{C}(t) \subseteq \mathcal{V}$ v danem trenutku t sestavlja prvih k vozlišč glede na največje vrednosti (v trenutku t) izbrane mere pomembnosti; to smo uporabili v razdelku 9.3.2.

Analizo omejimo na vrednosti v časovnem vektorju mer pomembnosti, ki ustrezajo vozliščem iz izbrane skupine $\mathcal{C}$.

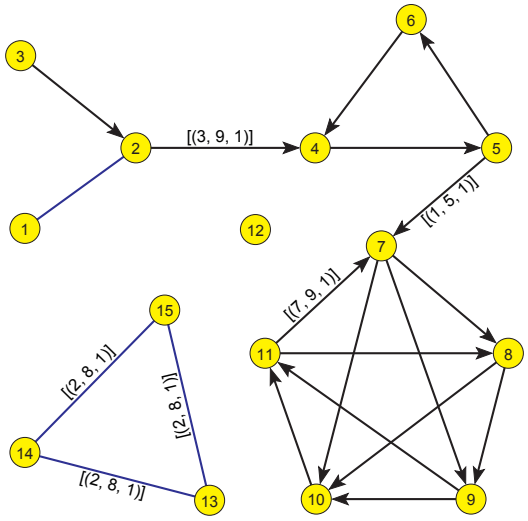
Za določanje skupin bi lahko uporabili tudi postopka prerezov (*cuts*) in otokov (*islands*), razvita v [83]. Posplošitev obeh postopkov na časovna omrežja je ena od tem, ki se jih nameravamo lotiti v nadaljnjem raziskovanju.

Poglavje 9

Analiza časovnih omrežij

9.1 Testna časovna omrežja

Testna časovna omrežja smo izbrali tako, da so dovolj majhna, da lahko bralec ročno preveri dobljene rezultate.



Slika 9.1: Prvi primer časovnega omrežja. Vse povezave, ki niso označene, imajo vrednost $[(1, 9, 1)]$.

Za časovno omrežje na sliki 9.1 smo v tabelo 9.1 zapisali pripadajoče časovne vhodne in izhodne stopnje. Vidimo, da ima vozlišče 5 na časovnem intervalu $[1, 5)$ izhodno stopnjo 2. Ker povezava $(5, 7)$ od časovne točke 5 ne obstaja več, se izhodna stopnja vozlišča na intervalu $[5, 9)$ zmanjša na 1.

Tudi rezultati obeh funkcij za izračun dosegljivosti so predstavljeni v tabeli

Tabela 9.1: Časovne vhodne in izhodne stopnje vozlišč, časovna dosegljivost, časovno šibko in krepko razbitje in časovna nakopičenost za prvi primer časovnega omrežja.

```
Vhodne stopnje
1 : [(1, 9, 1)]
2 : [(1, 9, 2)]
3 : []
4 : [(1, 3, 1), (3, 9, 2)]
5 : [(1, 9, 1)]
6 : [(1, 9, 1)]
7 : [(1, 5, 1), (7, 9, 1)]
8 : [(1, 9, 2)]
9 : [(1, 9, 2)]
10 : [(1, 9, 3)]
11 : [(1, 9, 2)]
12 : []
13 : [(2, 8, 2)]
14 : [(2, 8, 2)]
15 : [(2, 8, 2)]
```

```
Izhodne stopnje
1 : [(1, 9, 1)]
2 : [(1, 3, 1), (3, 9, 2)]
3 : [(1, 9, 1)]
4 : [(1, 9, 1)]
5 : [(1, 5, 2), (5, 9, 1)]
6 : [(1, 9, 1)]
7 : [(1, 9, 3)]
8 : [(1, 9, 2)]
9 : [(1, 9, 2)]
10 : [(1, 9, 1)]
11 : [(1, 7, 1), (7, 9, 2)]
12 : []
13 : [(2, 8, 2)]
14 : [(2, 8, 2)]
15 : [(2, 8, 2)]
```

```
Vhodna dosegljivost
1 : [(1, 9, 3)]
2 : [(1, 9, 3)]
3 : []
4 : [(1, 3, 3), (3, 9, 6)]
5 : [(1, 3, 3), (3, 9, 6)]
6 : [(1, 3, 3), (3, 9, 6)]
7 : [(1, 3, 3), (3, 5, 6), (7, 9, 5)]
8 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
9 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
10 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
11 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
12 : []
13 : [(2, 8, 3)]
14 : [(2, 8, 3)]
15 : [(2, 8, 3)]
```

```
Izhodna dosegljivost
1 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
2 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
3 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
4 : [(1, 5, 8), (5, 9, 3)]
5 : [(1, 5, 8), (5, 9, 3)]
6 : [(1, 5, 8), (5, 9, 3)]
7 : [(1, 7, 4), (7, 9, 5)]
8 : [(1, 7, 4), (7, 9, 5)]
9 : [(1, 7, 4), (7, 9, 5)]
10 : [(1, 7, 4), (7, 9, 5)]
11 : [(1, 7, 4), (7, 9, 5)]
12 : []
13 : [(2, 8, 3)]
14 : [(2, 8, 3)]
15 : [(2, 8, 3)]
```

```
Šibko razbitje
1 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
2 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
3 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
4 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
5 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
6 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
7 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
8 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
9 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
10 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
11 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
12 : []
13 : [(2, 8, 6)]
14 : [(2, 8, 6)]
15 : [(2, 8, 6)]
```

```
Krepko razbitje
1 : [(1, 9, 1)]
2 : [(1, 9, 1)]
3 : []
4 : [(1, 9, 2)]
5 : [(1, 9, 2)]
6 : [(1, 9, 2)]
7 : [(7, 9, 3)]
8 : [(1, 7, 4), (7, 9, 3)]
9 : [(1, 7, 4), (7, 9, 3)]
10 : [(1, 7, 4), (7, 9, 3)]
11 : [(1, 7, 4), (7, 9, 3)]
12 : []
13 : [(2, 8, 5)]
14 : [(2, 8, 5)]
15 : [(2, 8, 5)]
```

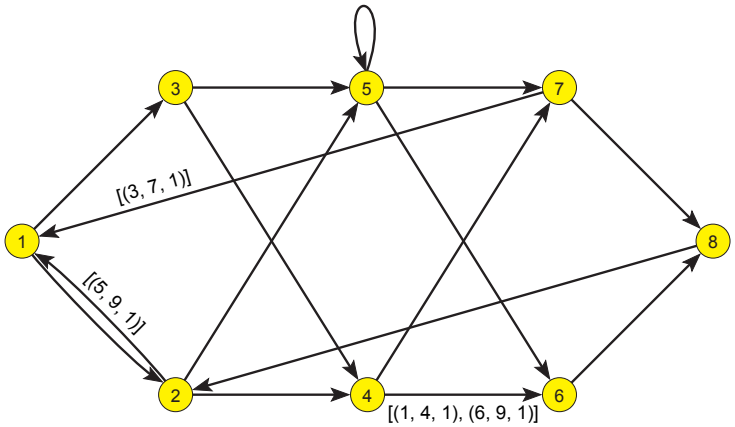
```
Klasična nakopičenost (type=1)
1 : []
2 : []
3 : []
4 : [(1, 3, 0.5), (3, 9, 0.1667)]
5 : [(1, 5, 0.1667), (5, 9, 0.5)]
6 : [(1, 9, 0.5)]
7 : [(1, 5, 0.25), (5, 9, 0.5)]
8 : [(1, 7, 0.4167), (7, 9, 0.5)]
9 : [(1, 7, 0.4167), (7, 9, 0.5)]
10 : [(1, 7, 0.4167), (7, 9, 0.5)]
11 : [(1, 9, 0.5)]
12 : []
13 : [(2, 8, 1.0)]
14 : [(2, 8, 1.0)]
15 : [(2, 8, 1.0)]
```

```
Izboljšana nakopičenost (type=2/3)
1 : []
2 : []
3 : []
4 : [(1, 3, 0.25), (3, 9, 0.125)]
5 : [(1, 5, 0.125), (5, 9, 0.25)]
6 : [(1, 9, 0.25)]
7 : [(1, 5, 0.25), (5, 7, 0.375),
     (7, 9, 0.5)]
8 : [(1, 7, 0.4167), (7, 9, 0.5)]
9 : [(1, 7, 0.4167), (7, 9, 0.5)]
10 : [(1, 7, 0.4167), (7, 9, 0.5)]
11 : [(1, 7, 0.375), (7, 9, 0.5)]
12 : []
13 : [(2, 8, 0.5)]
14 : [(2, 8, 0.5)]
15 : [(2, 8, 0.5)]
```

9.1. Iz vozlišča 6 je na časovnem intervalu $[1, 5)$ dosegljivih 8 vozlišč, na časovnem intervalu $[5, 9)$ pa so iz vozlišča 6 dosegljiva tri vozlišča (4, 5 in 6).

Zapisana sta tudi časovno šibko razbitje in časovno krepko razbitje. Vozlišče 12 je izolirano. Vozlišča 13, 14 in 15 so šibka komponenta na časovnem intervalu $[2, 8)$. Na časovnem intervalu $[1, 3)$ imamo še dve šibki komponenti: komponento $\{1, 2, 3\}$ in komponento $\{4, 5, 6, 7, 8, 9, 10, 11\}$.

Izračunali smo tudi klasično in izboljšano nakopičenost.



Slika 9.2: Drugi primer časovnega omrežja. Vse povezave, ki niso označene, imajo vrednost $[(1, 9, 1)]$.

Na sliki 9.2 je narisan drugi primer časovnega omrežja, ki je razširjena različica omrežja na sliki 3 v [4]. Za to omrežje smo izračunali časovno izhodno dostopnost vozlišč in časovno vmesnost.

Najprej zapišimo nekaj izbranih elementov strogega zaprtja $\mathbf{D}$ časovne sosednostne matrike omrežja. Vrednosti časovne količine $\mathbf{D}[3, 1]$ pomenijo, da je dolžina najkrajše poti od vozlišča 3 do vozlišča 1 na časovnem intervalu $[3, 7)$ enaka 3, na časovnem intervalu $[7, 9)$ pa 5. Izven teh intervalov vozlišče 1 ni dosegljivo iz vozlišča 3.

```
D[3,1] = [(3, 7, 3), (7, 9, 5)]
D[4,6] = [(1, 4, 1), (4, 6, 5), (6, 9, 1)]
D[6,3] = [(3, 5, 6), (5, 9, 4)]
D[7,6] = [(1, 9, 4)]
```

Časovne vrednosti izhodne dostopnosti za drugi primer časovnega omrežja so zapisane v tabeli 9.2.

Izračunali smo tudi vmesnost za drugi primer časovnega omrežja. Najprej zapišimo nekaj izbranih elementov strogega zaprtja matrike $\mathbf{A}$ nad enostavnim časovnim geodezičnim polkolobarjem, to je časovne matrike $\mathbf{C}$. Nekaj elementov te matrike za drugi primer časovnega omrežja je:

Tabela 9.2: Izhodna dostopnost za drugi primer časovnega omrežja.

```
1 : [(1, 9, 0.4375)]
2 : [(1, 3, 0.0000), (3, 5, 0.4375), (5, 9, 0.5833)]
3 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3889)]
4 : [(1, 3, 0.0000), (3, 4, 0.4375), (4, 6, 0.3500),
     (6, 7, 0.4375), (7, 9, 0.3500)]
5 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3500)]
6 : [(1, 3, 0.0000), (3, 5, 0.2917), (5, 9, 0.3500)]
7 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3500)]
8 : [(1, 3, 0.0000), (3, 5, 0.3500), (5, 9, 0.4375)]
```

```
C[1,7] = [(1, 9, (3, 4))]
C[2,2] = [(1, 3, (4, 4)), (3, 4, (4, 6)),
          (4, 5, (4, 5)), (5, 9, (2, 1))]
C[4,6] = [(1, 4, (1, 1)), (4, 6, (5, 3)), (6, 9, (1, 1))]
C[5,5] = [(1, 9, (1, 1))]
C[6,3] = [(3, 5, (6, 2)), (5, 9, (4, 1))]
C[7,6] = [(1, 3, (4, 2)), (3, 4, (4, 6)),
          (4, 6, (4, 3)), (6, 7, (4, 6)), (7, 9, (4, 2))]
```

Vrednost časovne količine $\mathbf{C}[4,6]$ pove, da med vozliščem 4 in vozliščem 6 na časovnih intervalih $[1,4)$ in $[6,9)$ obstaja usmerjena povezava, na časovnem intervalu $[4,6)$ pa sta vozlišči povezani s tremi najkrajšimi potmi dolžine 5. Zaporedja vozlišč na teh poteh so $(4,7,8,2,5,6)$, $(4,7,1,3,5,6)$ in $(4,7,1,2,5,6)$.

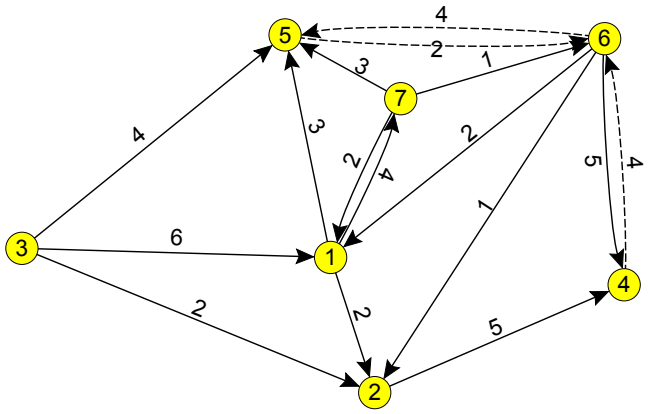
Vrednosti časovne vmesnosti za drugi primer časovnega omrežja so zapisane v tabeli 9.3.

Tabela 9.3: Časovna vmesnost za drugi primer časovnega omrežja.

```
1 : [(3,4,0.250), (4,6,0.275), (6,7,0.250), (7,9,0.143)]
2 : [(1,3,0.345), (3,4,0.405), (4,6,0.419), (6,7,0.405), (7,9,0.607)]
3 : [(1,3,0.059), (3,4,0.095), (4,6,0.105), (6,7,0.095), (7,9,0.059)]
4 : [(1,3,0.167), (3,4,0.250), (4,5,0.176), (5,6,0.105), (6,9,0.179)]
5 : [(1,3,0.167), (3,4,0.250), (4,5,0.348), (5,6,0.276), (6,9,0.179)]
6 : [(1,3,0.119), (3,4,0.095), (4,6,0.054), (6,7,0.095), (7,9,0.179)]
7 : [(1,3,0.119), (3,4,0.405), (4,5,0.469), (5,6,0.327), (6,7,0.262),
     (7,9,0.179)]
8 : [(1,3,0.309), (3,4,0.250), (4,6,0.248), (6,7,0.250), (7,9,0.524)]
```

Algoritme za izračun rekurzivnih mer pomembnosti iz razdelka 8.4 na strani 74 smo testirali na omrežju s slike 9.3. Časovne spremembe omrežja smo označili z utežmi na povezavah in črtkanimi povezavami takole:

Polne povezave so prisotne v omrežju ves opazovani čas, to je na intervalu $[1,9)$. Na časovnih intervalih $[1,3) \cup [7,9)$ je vrednost uteži na teh povezavah enaka 1, na intervalu $[3,7)$ pa je vrednost uteži enaka številu, ki je zapisano na povezavi (nekatere povezave ohranijo vrednost 1). Črtkane povezave so prisotne le znotraj intervala $[5,9)$. Na intervalu $[5,7)$ je vrednost uteži na povezavi enaka številu, ki je napisano na povezavi, na intervalu $[7,9)$ pa so vse uteži enake 1.



Slika 9.3: Tretji primer časovnega omrežja.

Časovne količine, ki opisujejo rekurzivne mere pomembnosti za testno časovno omrežje, so predolge, da bi jih zapisali v celoti. Običajno nas zanimajo le spremembe v vrstnem redu vozlišč, ki ga določa pomembnost. V tabelo 9.4 smo zapisali zaporedje vozlišč glede na vrednost izbrane pomembnosti na različnih časovnih intervalih.

	čas $[1,3)$	čas $[3,5)$	čas $[5,7)$	čas $[7,9)$
vhodni l. vekt.	4,5,2,1,7,6	5,4,7,1,2,6	4,6,5,1,2,7	6,5,4,2,1,7
izhodni l. vekt.	7,1,3,6	3,1,7,6	6,3,1,4,7,2,5	7,6,1,3,4,5,2
Katz $\alpha = 0.15$	2,5,1,4,7,6	5,4,7,1,2,6	5,6,4,1,7,2	5,6,2,1,4,7
Bonacich $\alpha = 0.85$	1,2,5,4,6,7,3	1,4,5,2,7,6,3	5,1,4,6,2,7,3	5,1,2,6,4,7,3
Bonacich $\beta = 0.15$	7,1,3,6,2	3,1,6,7,2	3,6,1,4,7,2,5	6,7,1,3,4,5,2
kazala	3,6,1,7,2	3,7,6,1,2	3,6,7,1,2,4,5	6,3,7,1,2,4,5
viri	1,2,5,4,6,7	1,5,2,4,7,6	5,1,4,2,7,6	5,1,2,4,6,7
pageRank $q = 0.15$	4,2,5,1,7,6,3	4,5,7,1,2,6,3	6,4,5,1,7,2,3	6,4,5,2,1,7,3

Tabela 9.4: Vrstni red vozlišč tretjega primera časovnega omrežja glede na časovne vrednosti pomembnosti.

Iz tabele vidimo, da nekatere mere pomembnosti ostanejo enake 0 za določena

vozlišča znotraj danega intervala. Na primer vozlišča 2, 4 in 5 na časovnem intervalu $[1, 3)$ manjkajo v vrstici, ki pripada izhodni lastni pomembnosti. To lahko opazimo tudi s slike, saj imata na tem časovnem intervalu vozlišči 4 in 5 izhodno stopnjo enako 0 in vozlišče 2 kaže le na vozlišče 4, ki ima ničelno pomembnost.

Druga zanimiva opomba je, da mere pomembnosti vrnejo podobne rezultate, če jih razdelimo v dve skupini: v prvi skupini so metode, za katere so pomembna vozlišča tista, ki imajo "več vhodnih povezav" (vhodna lastna pomembnost, Katzova pomembnost, Bonacicheva pomembnost α, viri). Druga skupina izbira vozlišča, ki imajo več "izhodnih povezav" (izhodna lastna pomembnost, Bonacicheva pomembnost (α, β), kazala, pageRank).

9.2 Omrežje Reutersovih novic o terorističnem napadu 11. septembra 2001

Omrežje novic o terorističnem napadu smo dobili iz CRA (Centering Resonance Analysis) omrežij, ki sta jih ustvarila Steve Corman in Kevin Dooley na Arizona State University [31]. Omrežje je zasnovano na vseh novicah, ki jih je o dogodku objavila agencija Reuters v zaporednih 66 dneh po napadu na ZDA 11. septembra 2001.

Vozlišča omrežja so besede in med besedama obstaja povezava, če se pojavljata v istem stavku. Utež na povezavi pove število pojavitev besed v istem stavku. Omrežje ima n = 13332 vozlišč (različnih besed, ki so se pojavljale v novicah) in m = 243447 povezav, od tega 50859 z utežjo, večjo od 1. V omrežju ni zank.

Omrežje novic, ki jih je o napadu 11. septembra objavila agencija Reuters, so uporabljali za testno omrežje na Viszards visualization sekciji konference Sunbelt XXII International Sunbelt Social Network Conference, ki je potekala v New Orleansu v ZDA med 13. in 17. februarjem 2002.

Omrežje smo iz oblike Pajek transformirali v obliko Ianus, ki jo uporabljamo v knjižnici TQ. Najprej smo določili najpomembnejša vozlišča; izračunali smo njihove skupne časovne pogostosti in nato izbrali podomrežje 50 najdejavnejših vozlišč znotraj opazovanih 66 dni. Ta vozlišča so zapisana v tabeli 9.5.

Če dobljeno omrežje narišemo, vidimo, da predstavlja skoraj poln graf. Zato smo odstranili vse časovne povezave, ki imajo utež manjšo od 10. Pripadajoči graf je predstavljen na sliki 9.4. Izolirana vozlišča smo odstranili.

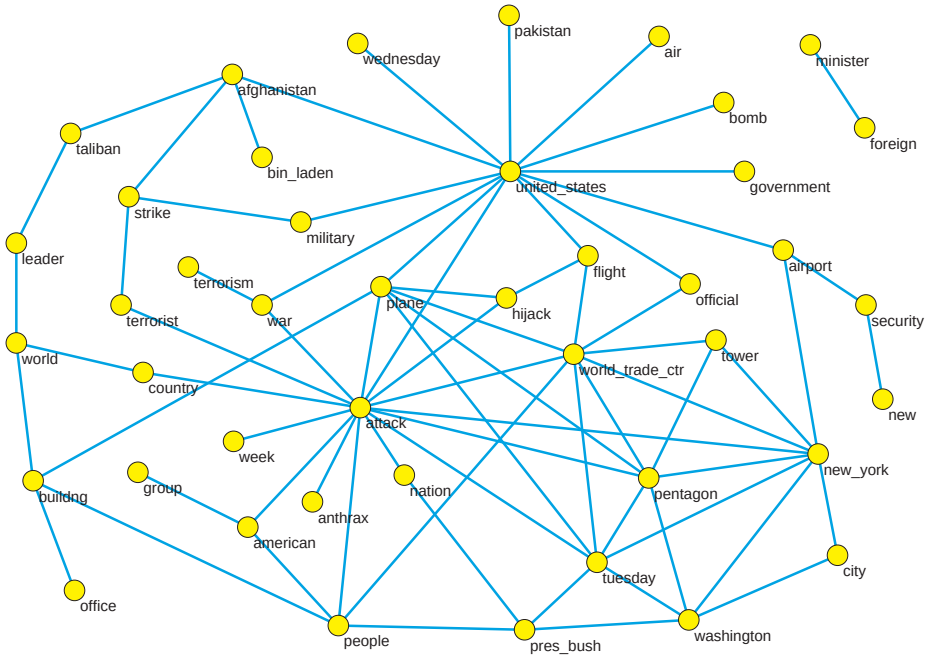
Za vsako od 50 vozlišč smo določili časovno dejavnost (razdelek 7.1.3, stran 64) in jo grafično predstavili. S pregledom rezultatov smo našli šest tipičnih vzorcev dejavnosti (slika 9.5). V vseh diagramih na sliki ležijo vrednosti na intervalu $[0, 200]$ – največja dejavnost za besedo *Wednesday* je večja od 200.

Tabela 9.5: 50 najpogostejših besed v omrežju novic o terorističnem napadu.

n	beseda	Σfreq	n	beseda	Σfreq
1	united_states	15000	26	terrorism	2212
2	attack	10348	27	day	2128
3	taliban	6266	28	week	2017
4	people	5286	29	worker	1983
5	afghanistan	5176	30	office	1967
6	bin_laden	4885	31	group	1966
7	new_york	4832	32	air	1962
8	pres_bush	4506	33	minister	1919
9	washington	4047	34	time	1898
10	official	3902	35	hijack	1884
11	anthrax	3563	36	strike	1818
12	military	3394	37	afghan	1775
13	plane	3078	38	flight	1775
14	world_trade_ctr	3006	39	tell	1746
15	security	2906	40	terrorist	1745
16	american	2825	41	airport	1741
17	country	2794	42	pakistan	1714
18	city	2689	43	tower	1685
19	war	2679	44	bomb	1674
20	tuesday	2635	45	new	1650
21	pentagon	2620	46	buildng	1634
22	force	2516	47	wednesday	1593
23	government	2380	48	nation	1589
24	leader	2375	49	police	1587
25	world	2213	50	foreign	1558

Osnovne besede so besede, ki imajo visoko število pojavitev v prvem tednu po 11. septembru in manjše, počasi padajoče vrednosti v kasnejšem obdobju. Predstavnik te skupine na sliki 9.5 je beseda *hijack*, ostali pripadniki pa so *airport, american, attack, city, day, flight, nation, New York, official, Pentagon, people, plane, police, president Bush, security, tower, United States, Washington, world* in *World Trade Center*. To so besede, ki opisujejo dogodek.

Sekundarne besede so reakcija na dogodek. V njihovih vrednostih ni velikih razlik. To skupino lahko razdelimo na tri podskupine: a) skupino *počasi zamirajočih* besed, ki jo predstavlja beseda *bin Laden (country, foreign, government, military, minister, new, Pakistan, tell, terrorism, terrorist, time, war, week)*; b)



Slika 9.4: Omrežje 11. september.

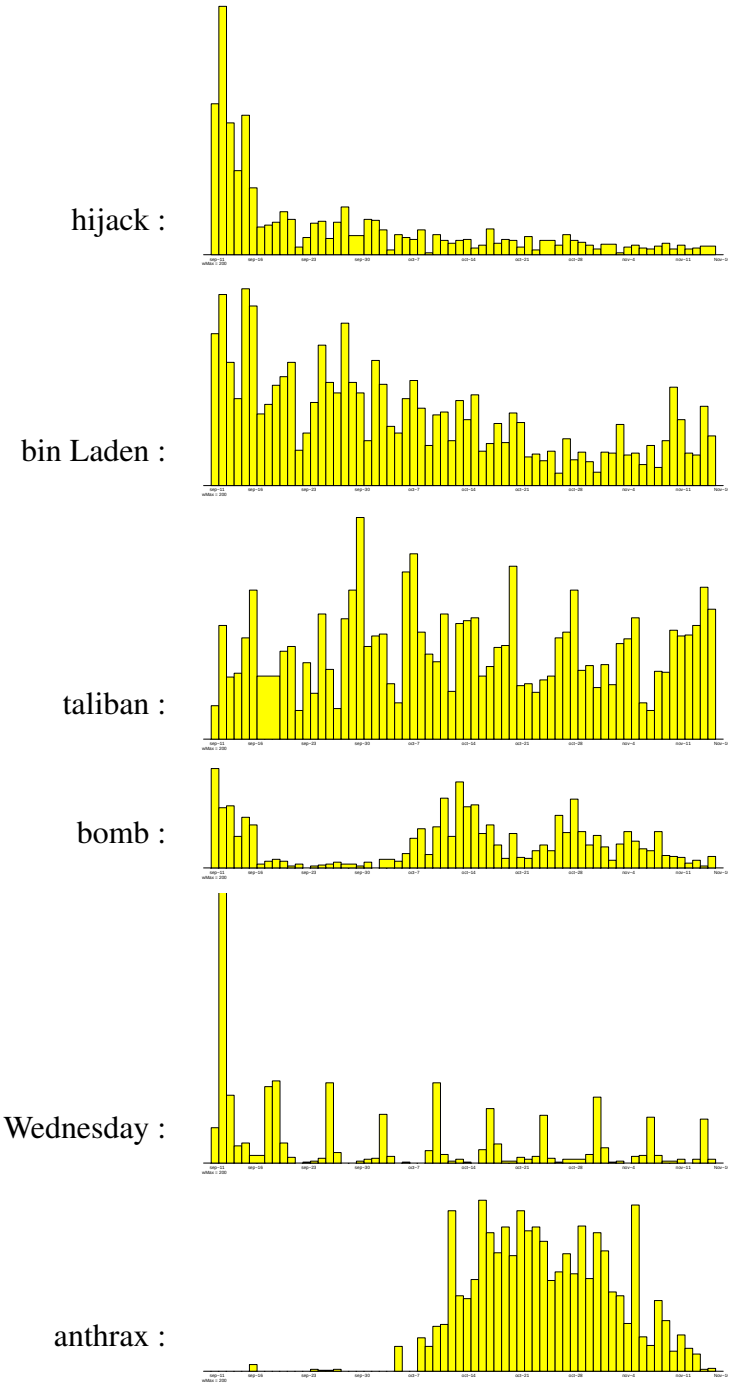
skupino *stacionarnih* besed, ki jo predstavlja beseda *taliban (afghan, Afghanistan, force, group, leader)*; in c) skupino *občasnih* besed z nekaj skoki, ki jo predstavlja beseda *bomb (air, building, office, strike, worker).*

Obstajajo trije posebni vzorci – dva *periodična* (*Wednesday* in *Tuesday*); in en *epizodičen* (*anthrax*).

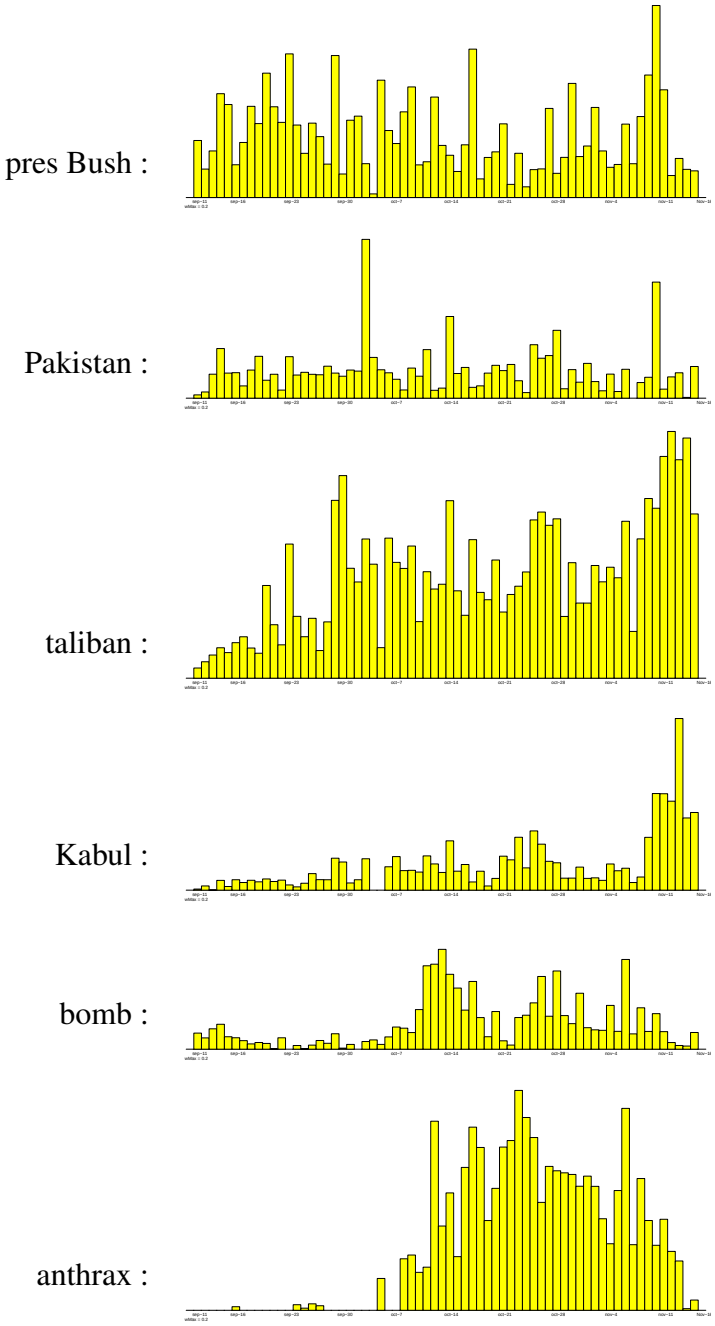
Izračunali smo tudi časovno privlačnost (razdelek 7.1.3, stran 64) in pripadajočo skupno privlačnost za vsa vozlišča omrežja. Izbrali smo 30 vozlišč z največjo skupno vrednostjo privlačnosti. Ta vozlišča so zapisana v tabeli 9.6. Na sliki 9.6 grafično predstavimo časovne vrednosti privlačnosti za izbranih šest besed. Za vse diagrame na sliki so vrednosti privlačnosti z intervala $[0, 0.2]$.

Če primerjamo vrednosti dejavnosti s slike 9.5 z vrednostmi privlačnosti s slike 9.6 na skupnih besedah (*taliban, bomb, anthrax*), opazimo, da so vrednosti korelirane (očitno velja, da ničelna dejavnost pomeni tudi ničelno privlačnost), vendar različne v podrobnostih.

Na primer besedi *taliban* in *bomb* imata na začetku obdobja majhno vrednost privlačnosti, saj sta bili obrobni osnovnim besedam. Besedi *taliban* in *Kabul* imata vedno večjo privlačnost proti koncu opazovanega obdobja.



Slika 9.5: Oblike dejavnosti vozlišč.



Slika 9.6: Vzorci privlačnosti.

Tabela 9.6: Trideset besed z največjim koeficientom privlačnosti v omrežju novic o terorističnem napadu.

n	beseda	Σatt	n	beseda	Σatt
1	united_states	12.216	16	war	2.758
2	taliban	7.096	17	force	2.596
3	attack	7.070	18	new_york	2.590
4	afghanistan	5.142	19	government	2.496
5	people	5.023	20	day	2.338
6	bin_laden	4.660	21	leader	2.305
7	anthrax	4.601	22	terrorism	2.202
8	pres_bush	4.374	23	time	2.182
9	country	3.317	24	group	2.072
10	washington	3.067	25	afghan	2.040
11	security	2.939	26	world	1.995
12	american	2.922	27	week	1.961
13	official	2.831	28	pakistan	1.943
14	city	2.798	29	letter	1.866
15	military	2.793	30	new	1.851

9.2.1 Rekurzivne mere pomembnosti v omrežju novic o terorističnem napadu

Rekurzivne mere pomembnosti (razdelek 8.4, stran 74) smo izračunali na podomrežju 50 najdejavnejših vozlišč.

Metodi za lastni pomembnosti ne konvergirata dovolj hitro z začetnim približkom samih enic. Tudi razvrstitev vozlišč pageRank ne pove prav veliko o pomembnosti vozlišč, saj precej naključno skače tako v vrednosti kot v tem, katero vozlišče ima največjo vrednost pomembnosti na časovnem intervalu.

Ostale metode razdelimo v dve skupini. Prva skupina ustreza vprašanjem: "Na katere besede najbolj kažejo novice? Kam so usmerjene? Kaj je končen cilj?" Vse metode največkrat kot najpomembnejše vrnejo besede *attack*, *afghanistan* in *anthrax*. Metode iz te skupine so Katzova pomembnost, izračunana na transponirani matriki omrežja, vrednost kazala, Bonacicheva pomembnost α, izračunana na transponirani matriki omrežja, in Bonacicheva pomembnost (α, β). Vrednost pomembnosti vozlišč se s časom zmanjšuje.

Druga skupina odgovarja na vprašanja: "Iz katerih besed se širijo novice? Kaj je začelo dogajanje?" in vse rekurzivne mere pomembnosti največkrat pripišejo največjo pomembnost besedi *United States*, razen v prvem tednu po napadu, v

katerem ima največjo pomembnost beseda *World Trade Center*. Metode iz te skupine so Katzova pomembnost, vrednost vira, Bonacicheva pomembnost α in Bonacicheva pomembnost (α, β), izračunana na transponirani matriki omrežja.

Kot primer rezultata metode iz prve skupine zapišimo število pojavitev besed z najvišjo Bonacichevo pomembnostjo α, izračunano na transponirani matriki omrežja. Največjo pomembnost ima 50-krat beseda *attack*, desetkrat beseda *afghanistan*, štirikrat beseda *anthrax* in enkrat beseda *leader*.

Kot primer iz druge skupine zapišimo število pojavitev besed z največjo Katzovo pomembnostjo. Skozi čas je 49-krat najpomembnejša beseda *United States*, sedemkrat beseda *World Trade Center*, štirikrat beseda *washington*, dvakrat besedi *taliban* in *war*, enkrat besedi *world* in *wednesday*.

9.3 Franzosijevo omrežje nasilja v Italiji

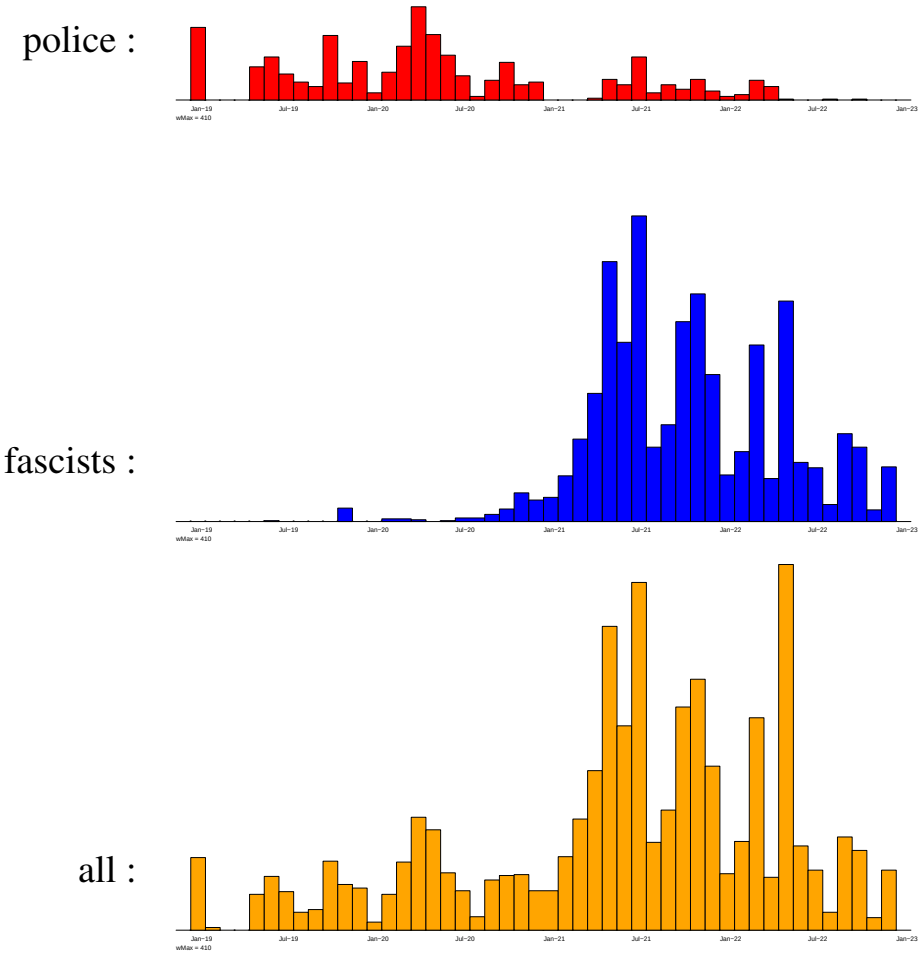
Iz časopisov, izdanih v obdobju med januarjem 1919 in decembrom 1922, je Roberto Franzosi zbral podatke o objavljenih nasilnih dejanjih – interakcijah med različnimi političnimi skupinami in drugimi skupinami ljudi v Italiji. Iz zbranih podatkov smo ustvarili Franzosijevo časovno omrežje nasilja v Italiji [44]. Vozlišča v omrežju predstavljajo vpletene skupine ljudi (na primer *people*, *police*, *fascists*, *communists*, *socialists*, *workers*). Uteži na povezavah povejo število nasilnih interakcij med skupinama (povezava (u, v) z utežjo 3 pomeni, da je skupina u storila tri nasilna dejanja nad skupino v). Časovne količine na povezavah omrežja štejejo nasilna dejanja v enem mesecu za vse mesece znotraj štirih let.

9.3.1 Dejavnost v Franzosijevem omrežju nasilja v Italiji

Na Franzosijevem časovnem omrežju nasilja smo najprej izračunali dejavnosti skupin vozlišč (razdelek 7.1.3, stran 64). Določili smo časovne količine

$$\begin{aligned} \textit{police} &= \mathrm{act}(\{\text{police}\}, \mathcal{V}) + \mathrm{act}(\mathcal{V}, \{\text{police}\}), \\ \textit{fascists} &= \mathrm{act}(\{\text{fascists}\}, \mathcal{V}) + \mathrm{act}(\mathcal{V}, \{\text{fascists}\}) \text{ in} \\ \textit{all} &= \mathrm{act}(\mathcal{V}, \mathcal{V}). \end{aligned}$$

Vrednosti rezultatov so grafično prikazane na sliki 9.7. Ko primerjamo rezultate o silovitosti nasilnih dejanj policije, nasilnih dejanj fašistov in vseh nasilnih dejanj, vidimo, da je bila večina nasilnih dejanj v prvih dveh letih (1919 in 1920) plod policijskega nasilja. Naslednji dve leti (1921 in 1922) so skoraj vsa nasilna dejanja zagrešili fašisti.



Slika 9.7: Nasilna dejavnost policije, fašistov in skupna nasilna dejavnost.

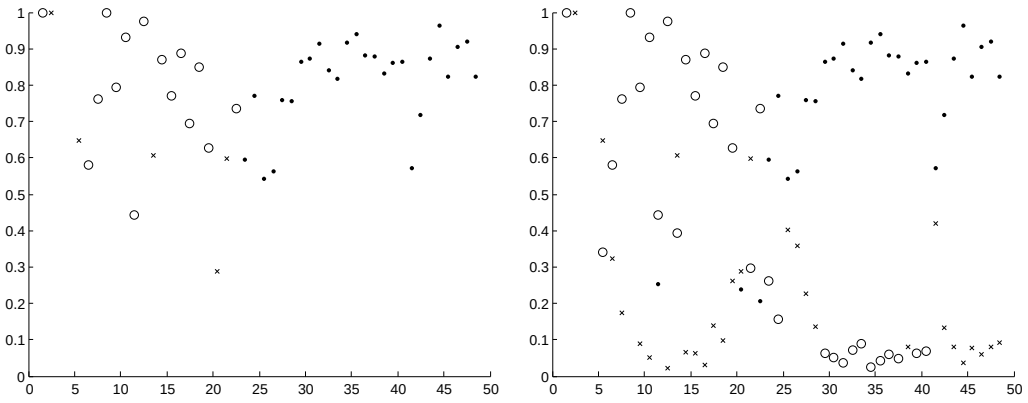
9.3.2 Rekurzivne mere pomembnosti za Franzosijevo omrežje nasilja v Italiji

Na Franzosijevem omrežju nasilja v Italiji smo uporabili tudi algoritme za izračun rekurzivnih mer pomembnosti (razdelek 8.4, stran 74).

Najlepše rezultate dobimo z algoritmom kazala in viri, kar smo pričakovali glede na naravo omrežja. Skupine lahko vidimo kot nasilneže (tiste, ki nasilje izvajajo) ali kot žrtve (tiste, nad katerimi je bilo storjeno nasilno dejanje).

Ustvarili smo časovno premico, na kateri so vidne spremembe pomembnosti vozlišč. Vrednosti kazal so prikazane na sliki 9.8. Časovne točke so meseci in višina simbolov je enaka vrednosti normalizirane (glede na prvo normo) vrednosti

kot kazala. Oblika točke (krogec, pika ali križec) ustreza skupini, ki ima v določenem mesecu najvišjo vrednost pomembnosti. S slike 9.8 jasno vidimo, da je v nekem trenutku nasilje policije prevzelo nasilje fašistov. To se zgodi v časovni točki 23, ki ustreza novembru 1920. S slike na desni strani vidimo, da je nekaj časa policija ohranila nekaj svoje nasilne dejavnosti in bila druga glede na nasilna dejanja, kasneje pa je popolnoma izginila.



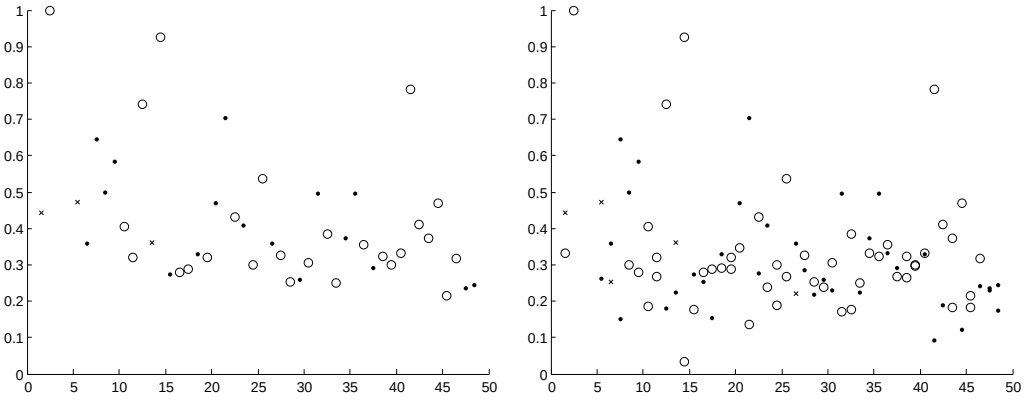
Slika 9.8: Najvišje vrednosti kazal Franzosijevega omrežja skozi čas (levo) in dve najvišji vrednosti kazal (desno). Skupina fašistov je narisana s črnimi pikami in skupina policije s krogci. Druge skupine so predstavljene križci.

Vrednosti virov so orisane na sliki 9.9. Tukaj ni jasnega trenda in izgleda, da nasilna dejanja v opazovanem obdobju niso bila omejena na posebno skupino ljudi. To se sklada z zgodovinskimi podatki.

Z ostalimi rekurzivnimi merami pomembnosti dobimo podobne rezultate, le meja ni vedno tako očitna. Za vhodno lastno pomembnost imajo najvišjo vrednost pomembnosti 24-krat *workers*, *workers (agricultural)* ali *socialists* in 14-krat *undefined*, *people* ali *protesters*. Poleg teh so še tri druge skupine dobile največjo pomembnost v mesecu (enkrat *police* in dvakrat *fascists*).

Izhodna lastna pomembnost vrne mešanico *police* (4), *protesters* (3), *?* (3), *undefined* (2), *workers* (2), *workers (agricultural)* (1) in *republicans* (1). Prva pojavitev skupine fašistov je v časovni točki 23 (november 1920). Fašisti imajo najvišjo vrednost pomembnosti do konca časovnega obdobja, razen v štirih mesecih (ko imajo največjo pomembnost skupine *the right*, *?*, *workers* ali *police*).

Pomembnost pageRank za $q = 0.15$ vrne 18-krat *fascists*, pojavitve se začnejo s časovno točko 22 (oktober 1920), ki jo potem prekinejo *workers* (3), *people* (3), *undefined* (2) in *police*. Do tega trenutka imamo mešanico *police* (6), *undefined* (5), *people* (4), *socialists* (3), *war affected* in *the right*.



Slika 9.9: Najvišje vrednosti virov Franzosijevega omrežja skozi čas (levo) in dve najvišji vrednosti virov (desno). Skupine delavcev, kmetov in socialistov so narisane s črnimi pikami. Skupine nedefiniranih, ljudi in protestnikov so predstavljene s krogci. Druge skupine so označene s križci.

Zdi se, da so nasilneži precej bolj jasni kot žrtve, zato smo izračunali Katzovo pomembnost in Bonacichevo pomembnost α na transponirani matriki omrežja.

Bonacicheva pomembnost α za $\alpha = 0.9$ vrne 17-krat *police* do časovne točke 23 (drugi z najvišjo pomembnostjo do tega časa so *thugs, undefined* in dvakrat *workers*). Po časovni točki 23 imamo 23-krat *fascists*, ostali pa so *thugs, police* in dvakrat *workers*.

Katzova mera pomembnosti 16-krat vrne *police* in po enkrat *thugs, undefined, protesters* in *?*. Po časovni točki 23 se pojavlja le skupina *fascists*. Bonacicheva pomembnost (α, β) vrne enak rezultat.

Te rezultate povzamemo v tabeli 9.7, v kateri smo zapisali število pojavitev skupin *police, fascists* in drugih skupin z najvišjo vrednostjo za različne rekurzivne mere pomembnosti. Štetje za vsako pomembnost je razdeljeno v dva stolpca: v prvem stolpcu je število pojavitev pred novembrom 1920, v drugem pa število pojavitev po novembru 1920 (vključujoče). V tabeli so zapisani podatki za mere pomembnosti, ki dajejo informacije o nasilnežih. Iz tabele jasno vidimo, da je v vseh primerih fašistično nasilje osrednje v objavljenih novicah po novembru 1920. Pred tem časom je osrednje policijsko nasilje. Zaradi ničelnih vrednosti pomembnosti na nekaterih intervalih, se vsote pojavitev v stolpcih ne ujemajo.

skupina ljudi	kazalo		pageRank		izhodni l.vekt.		Bonacich α		Bonacich α, β		Katz	
policija	15	0	6	1	4	1	17	1	16	0	16	0
fašisti	0	26	1	18	0	21	0	23	0	25	0	25
drugi	5	0	14	8	12	3	4	3	4	0	4	0

Tabela 9.7: Povzetek največjih vrednosti mer pomembnosti pred novembrom 1920 in po njem za nasilneže v Franzosijevem omrežju.

Poglavje 10

Zaključek in prihodnje delo

V disertaciji smo opisali nov algebraičen pristop k analizi časovnih omrežij, ki temelji na pojmu časovnih količin nad izbranimi polkolobarji. Definirali smo polkolobarje za analizo časovnih omrežij brez potovalnih in čakalnih časov, za iskanje prvih potovanj in za analizo časovnih omrežij, ki imajo poleg potovalnega časa še dodatne informacije na povezavah.

Naš opis časovnega omrežja se izogne eksplicitnemu zapisu prisotnosti vozlišč in povezav, kot so funkcije f_i in g_i iz opisa časovnega omrežja v obliki *time-aggregated graph* in funkcije ρ in ψ iz opisa v obliki TVG. Prisotnost oz. odsotnost vozlišč in povezav implicitno opišemo s pomočjo ničle polkolobarja. Če želimo eksplicitno povedati, kdaj sta izbrano vozlišče ali povezava prisotna, to še vedno lahko storimo z dvojiško časovno količino, v kateri enica pomeni prisotnost in ničla odsotnost povezave ali vozlišča. Najpomembnejša razlika med našim in drugimi zapisi je v tem, da načina opisa *time-aggregated graph* in TVG dovoljujeta le eno vrsto časovnih uteži na povezavah (potovalni čas). Velika prednost našega opisa je v možnosti, da na povezavi ali v vozlišču dovolimo strukturirane vrednosti. Naš opis časovnega omrežja dovoljuje, da poleg potovalnega časa na povezavi opišemo tudi dodatne informacije, ki so prav tako lahko odvisne od časa. Poznamo lahko še npr. dolžine poti, število načinov za prehod povezave, ki imajo ob različnih časih različne vrednosti.

Za časovna omrežja brez potovalnih in čakalnih časov smo predstavili algoritme, ki so posplošitve standardnih metod za analizo statičnih omrežij. Osredotočili smo se predvsem na mere pomembnosti vozlišč v omrežju. Pristop predstavlja vzdolžno alternativo tradicionalnim razbitjem časovnega omrežja na časovne rezine. Glavni prednosti pristopa sta, da lahko podatke in rezultate zapišemo s pomočjo časovnih količin, ki so zelo naraven način za opis časovnih lastnosti omrežja, in da uporabniku ni treba paziti na časovne intervale, na katerih so definirane časovne rezine. Primerne intervale izberejo operacije v časovnih polkolobarjih. Zato je analiza lahko precej bolj učinkovita.

Na časovna omrežja brez potovalnih in čakalnih časov smo posplošili večino najpogosteje uporabljanih mer pomembnosti in nekatere druge klasične prijeme v analizi omrežij. S pomočjo časovnih mer pomembnosti lahko določimo skupine najpomembnejših vozlišč v časovnem omrežju in njihov razvoj znotraj življenjske dobe omrežja. Opazimo lahko tudi menjavo vlog določenih vozlišč. Opisali smo tudi način, s katerim lahko poljubno ekvivalenčno relacijo na vozliščih časovnega omrežja zapišemo s pomočjo časovnega razbitja. Pristop smo uporabili za zapis razvoja šibkih in krepkih komponent omrežja. V prihodnosti bi lahko definirali relacije, ki bodo bolje opisale zaželene lastnosti vozlišč.

Vse opisane algoritme in še nekaj drugih smo programsko podprli s Pythonsko knjižnico TQ, ki je dostopna na spletni strani `http://pajek.imfm.si/doku.php?id=tq`. Začeli smo tudi z razvojem programa Ianus, ki bo zmožnosti knjižnice TQ nudil na uporabniku prijazen način (kot to dela program Pajek za statična omrežja).

Trenutna različica knjižnice TQ je osnovana na matrični predstavitvi časovnih omrežij, s katero ima večina algoritmov časovno zahtevnost $\mathcal{O}(n^3L)$ ali $\mathcal{O}(kn^2L)$. Zaradi precej visoke časovne zahtevnosti je uporaba prikazanih algoritmov omejena na srednje velika omrežja (velikosti do nekaj tisoč vozlišč). Ker so velika omrežja običajno redka ($m = \mathcal{O}(n)$), bi lahko v prihodnosti razvili učinkovitejše algoritme, ki bi uporabljali grafovsko (redko) predstavitev omrežja. Poleg tega bi lahko uporabili druge iterativne algoritme za računanje lastnih parov in za reševanje sistemov linearnih enačb, ki so bili razviti prav za velike redke matrike.

Na podoben način kot smo to storili za opisane mere, bi v prihodnosti lahko razvili tudi algoritme za druge mere in indekse, ki se uporabljajo v analizi statičnih omrežij.

Rezultati, ki jih dobimo s časovnimi postopki, so precej veliki (obsežni). Poiskati bo treba učinkovitejše načine za vizualizacijo in pregledovanje rezultatov.

V algoritmih v večini primerov uporabljamo kombinatorični polkolobar. V nadaljevanju bi lahko uporabili linearno algebro nad drugimi polkolobarji, če bi raziskali pomen takega izračuna v omrežju. Zanimivo bi bilo raziskati tudi pomen nedominantnih lastnih parov.

Zdi se tudi, da bi lahko nadaljevali v smeri napovedovanja razvrstitve vozlišč v omrežju za kratek čas naprej v prihodnosti, če bi uporabili teorijo perturbacij lastnih vektorjev.

Za primer, ko potovalni časi niso enaki nič, je treba razviti še računalniško podporo in nove metode, ki ne bodo izhajale iz statičnega omrežja. V tem primeru so operacije precej drugačne in izkaže se, da je problem potovanj precej težji od problema skokov. Najbolj obetavna (in uporabna) mera pomembnosti vozlišč v tem primeru izgleda nekakšna posplošitev vmesnosti na časovna omrežja s potovalnim časom. To zamisel smo nakazali že v disertaciji.

Z definicijami polkolobarjev in z operacijami, s katerimi lahko matematično

opišemo potovanja v časovnih omrežjih, smo omogočili uporabo dodatnih informacij v njihovi analizi. Pregledati bi bilo treba, katere metode iz statičnih omrežij je mogoče posplošiti tudi na ta primer, gotovo pa je bolj razburljivo poiskati mere in pojme, ki bodo prirejeni časovni razsežnosti.

Literatura

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[3] John S. Baras and George Theodorakopoulos. Path problems in networks. *Synthesis Lectures on Communication Networks*, 3(1):1–77, 2010.

[4] Vladimir Batagelj. Semirings for social network analysis. *Journal of Mathematical Sociology*, 19(1):53–68, 1994.

[5] Vladimir Batagelj. Social network analysis, large-scale. In Robert A Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 8245–8265. Springer, 2009.

[6] Vladimir Batagelj and Selena Praprotnik. An algebraic approach to temporal network analysis. *Submitted to Social Networks*, 2014.

[7] Alex Bavelas. A mathematical model of group structure. *Human Organizations*, 7:16–30, 1948.

[8] Alex Bavelas and Dermot Barrett. *An Experimental Approach to Organizational Communication*. Publications: Industrial Relations. American Management Association, 1951.

[9] Michael G. H. Bell and Yasunori Iida. *Transportation network analysis*. Chichester: Wiley, 1997.

[10] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences (Classics in Applied Mathematics)*. Society for Industrial Mathematics, 1987.

[11] Sandeep Bhadra and Afonso Ferreira. Complexity of connected components in evolving graphs and the computation of multicast trees in dynamic networks. In Samuel Pierre, Michel Barbeau, and Evangelos Kranakis, editors, *ADHOC-NOW*, volume 2865 of *Lecture Notes in Computer Science*, pages 259–270. Springer, 2003.

[12] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Trans. Internet Technol.*, 5(1):92–128, 2005.

[13] Cemal Cagatay Bilgin and Bülent Yener. Dynamic network evolution: Models, clustering, anomaly detection. *IEEE Networks*, 2006.

[14] Stefano Bistarelli. *Semirings for soft constraint solving and programming*, volume 2962. Springer Science & Business Media, 2004.

[15] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Math.*, 10(3-4):222–262, 2014.

[16] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.

[17] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[18] Phillip Bonacich and Paulette Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23(3):191–201, 2001.

[19] Stephen P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, 2005.

[20] Stephen P. Borgatti and Martin G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006.

[21] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[22] Ulrik Brandes and Thomas Erlebach. *Network Analysis: Methodological Foundations (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[23] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, pages 107–117. Elsevier Science Publishers B. V., 1998.

[24] Andries E. Brouwer and Willem H. Haemers. *Spectra of Graphs*. Universitext. Springer, 2011.

[25] Kurt Bryan and Tanya Leise. The $25,000,000,000 eigenvector: the linear algebra behind google. *SIAM Review*, 48:569–581, 2006.

[26] Angelika Bunse-Gerstner, Ralph Byers, Volker Mehrmann, and Nancy K. Nichols. Numerical computation of an analytic singular value decomposition of a matrix valued function. *Numer. Math.*, 60:1–39, 1991.

[27] Bernard Carre. *Graphs and networks*. Clarendon Press; Oxford University Press Oxford; New York, 1979.

[28] Arnaud Casteigts and Paola Flocchini. Deterministic algorithms in dynamic networks: Formal models and metrics. Technical report, Commissioned by Defense Research and Development Canada (DRDC), 2013.

[29] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[30] Cordis. `http://cordis.europa.eu/`. Accessed: February 2015.

[31] Steven R. Corman, Timothy Kuhn, Robert D. McPhee, and Kevin J. Dooley. Studying complex discursive systems. Centering resonance analysis of communication. *Human communication research*, 28(2):157–206, 2002.

[32] José R. Correa and Nicolás E. Stier-Moses. Wardrop equilibria. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[33] Hua Dai and Peter Lancaster. Numerical methods for finding multiple eigenvalues of matrices depending on parameters. *Numer. Math.*, 76(2):189–208, 1997.

[34] Wouter de Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory Social Network Analysis with Pajek (Structural Analysis in the Social Sciences)*. Cambridge University Press, 2 edition, 2012.

[35] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[36] James W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[37] James W. Demmel, Luca Dieci, and Mark J. Friedman. Computing connecting orbits via an improved algorithm for continuing invariant subspaces. *SIAM J. Sci. Comput*, 22(1):81–94, 2000.

[38] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005.

[39] Stephen Dolan. Fun with semirings: A functional pearl on the abuse of linear algebra. *SIGPLAN Not.*, 48(9):101–110, September 2013.

[40] Jack Dongarra. Common issues. In Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[41] Jack Dongarra. Preconditioning techniques. In Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[42] Afonso Ferreira. Building a reference combinatorial model for manets. *IEEE Network*, 18(5):24–29, 2004.

[43] John G. Fletcher. A more general algorithm for computing closed semiring costs between vertices of a directed graph. *Commun. ACM*, 23(6):350–351, June 1980.

[44] Roberto Franzosi. Mobilization and counter-mobilization processes: From the "red years" (1919–20) to the "black years" (1921–22) in Italy. A new methodological approach to the study of narrative data. *Theory and Society*, 26(2):275–304, 1997.

[45] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[46] Linton C. Freeman. Centrality in social networks; Conceptual clarification. *Social Networks*, 1(3):215–239, 1978.

[47] Noah E. Friedkin. Theoretical foundations for centrality measures. *The American Journal of Sociology*, 96(6):1478–1504, 1991.

[48] Betsy George and Sangho Kim. *Spatio-temporal Networks; Modeling and Algorithms.* Springer Briefs in Computer Science. Springer, 2013.

[49] Betsy George, Sangho Kim, and Shashi Shekhar. Spatio-temporal network databases and routing algorithms: A summary of results. In Dimitris Papadias, Donghui Zhang, and George Kollios, editors, *SSTD*, volume 4605 of *Lecture Notes in Computer Science*, pages 460–477. Springer, 2007.

[50] Betsy George and Shashi Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. In John F. Roddick, V. Richard Benjamins, Samira Si-said Cherfi, Roger Chiang, Christophe Claramunt, Ramez A. Elmasri, Fabio Grandi, Hyoil Han, Martin Hepp, Miltiadis D. Lytras, Vojislav B. Mišić, Geert Poels, Il-Yeol Song, Juan Trujillo, and Christelle Vangenot, editors, *Advances in Conceptual Modeling - Theory and Practice*, volume 4231 of *Lecture Notes in Computer Science*, pages 85–99. Springer Berlin Heidelberg, 2006.

[51] Chris Godsil and Gordon F. Royle. *Algebraic Graph Theory*. Graduate Texts in Mathematics. Springer New York, 2001.

[52] Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms (Operations Research/Computer Science Interfaces Series)*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[53] F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[54] Taher Haveliwala and Sepandar Kamvar. The second eigenvalue of the google matrix. Technical Report 2003-20, Stanford InfoLab, 2003.

[55] Petter Holme. Network reachability of real-world contact sequences. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 71(4):46119, 2005.

[56] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[57] Petter Holme and Jari Saramäki. *Temporal networks. Understanding Complex Systems.* Springer, 2013.

[58] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[59] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[60] E.D. Kolaczyk. Statistical analysis of network data: Methods and models. *Springer Series In Statistics*, page 386, 2009.

[61] Peter Lancaster and Miron Tismenetsky. *The Theory of Matrices: With Applications*. Computer Science and Scientific Computing Series. Academic Press, 1985.

[62] Amy N. Langville and Carl D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Rev.*, 47(1):135–161, 2005.

[63] Harold J. Leavitt. Some effects of certain communication patterns on group performance. *Journal of Abnormal and Social Psychology*, 46(1):38–50, 1951.

[64] Joseph J. Moder and Cecil R. Phillips. *Project management with CPM and PERT*. Reinhold industrial engineering and management sciences textbook series. Reinhold Pub. Corp., 2 edition, 1970.

[65] Bojan Mohar. Laplace eigenvalues of graphs – a survey. *Discrete Mathematics*, 109(1–3):171–183, 1992.

[66] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350, 2002.

[67] Vincenzo Nicosia, John Tang, Mirco Musolesi, Giovanni Russo, Cecilia Mascolo, and Vito Latora. Components in time-varying graphs. *CoRR*, abs/1106.2134, 2011.

[68] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, 2007.

[69] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[70] Parus Analytics. `http://parusanalytics.com/`. Accessed: February 2015.

[71] Nicola Perra and Santo Fortunato. Spectral centrality measures in complex networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(3):036107+, 2008.

[72] Selena Praprotnik and Vladimir Batagelj. Semirings for temporal network analysis. *Submitted to IMA Journal of Applied Mathematics*, 2015.

[73] Selena Praprotnik and Vladimir Batagelj. Spectral centrality measures in temporal networks. *Accepted to Ars Mathematica Contemporanea*, 2015.

[74] John Riordan. *Introduction to Combinatorial Analysis*. Dover Books on Mathematics. Wiley New York, 1958.

[75] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

[76] Nicola Santoro, Walter Quattrociocchi, Paola Flocchini, and Arnaud Casteigts. Time-varying graphs and social network analysis: Temporal indicators and metrics. *3rd AISB Social Networks and Multiagent Systems Symposium (SNAMAS*, pages 32–38, 2011.

[77] Scopus. `http://www.scopus.com/`. Accessed: February 2015.

[78] Tom Snijders. Siena. `http://www.stats.ox.ac.uk/~snijders/siena/`. Accessed: February 2015.

[79] Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning; a revised report. In Daniel S. Weld and Johan De Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, 1990.

[80] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge University Press, 1994.

[81] Web of Science (Knowledge). `https://webofknowledge.com/`. Accessed: February 2015.

[82] Bui B. Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

[83] Matjaž Zaveršnik. *Razčlembe omrežij (Network decompositions): doktorska disertacija.* FMF, 2004.

[84] U. Zimmerman. *Annals of Discrete Mathematics: Linear and Combinatorial Optimization in Ordered Algebraic Structures*. North-Holland, 1981.