Analiza omrežij

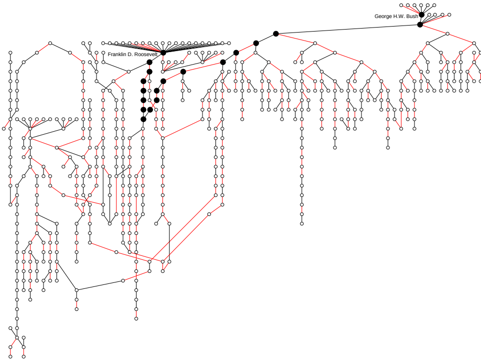
8. Aciklična omrežja

Vladimir Batagelj

Univerza v Ljubljani, FMF, matematika

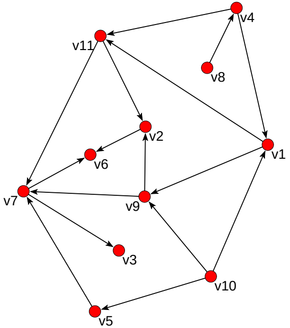
Interdisciplinarni doktorski študijski program Statistika
Ljubljana, april 2014

Kazalo



wiki: http://pajek.imfm.si/doku.php?id=event:pd
April 24, 2014/ april 2014

Aciklična omrežja



acyclic.paj

Omrežje $\mathcal{G} = (\mathcal{V}, \mathbf{R})$, $\mathbf{R} \subseteq \mathcal{V} \times \mathcal{V}$ je *aciklično*, če v njem ni nobenega (pravega) cikla.

$$\overline{\mathbf{R}} \cap I = \emptyset$$

$\overline{\mathbf{R}}$ je tranzitivna ovojnica relacije $\mathbf{R}$ – relacija *dosegljivosti*.
Včasih dopuščamo zanke $\overline{\mathbf{R} \setminus I} \cap I = \emptyset$.
Primeri acikličnih omrežij so: omrežja sklicevanj, rodovniki, projektna omrežja, ...

V dejanskih acikličnih omrežjih običajno obstaja lastnost $p : \mathcal{V} \to \mathbb{R}$ (najpogosteje čas), ki je usklajena s povezavami

$$(u, v) \in \mathbf{R} \Rightarrow p(u) < p(v)$$

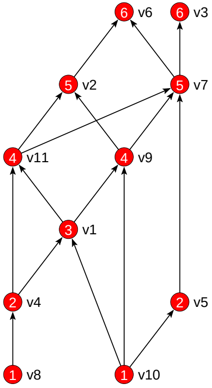
Osnovne lastnosti

Če je $\mathcal{G} = (\mathcal{V}, \mathbf{R})$ acikličen in $\mathcal{U} \subseteq \mathcal{V}$, je tudi $\mathcal{G}|\mathcal{U} = (\mathcal{U}, \mathbf{R}|\mathcal{U})$, $\mathbf{R}|\mathcal{U} = \mathbf{R} \cap \mathcal{U} \times \mathcal{U}$ acikličen.
Če je $\mathcal{G} = (\mathcal{V}, \mathbf{R})$ acikličen, je tudi $\mathcal{G}' = (\mathcal{V}, \mathbf{R}^{-1})$ acikličen. Dualnost.

Množica *začetkov* $\mathrm{Min}_{\mathbf{R}}(\mathcal{V}) = \{v : \neg\exists u \in \mathcal{V} : (u, v) \in \mathbf{R}\}$ in množica *koncev* $\mathrm{Max}_{\mathbf{R}}(\mathcal{V}) = \{v : \neg\exists u \in \mathcal{V} : (v, u) \in \mathbf{R}\}$ sta v končnem acikličnem omrežju neprazni.

Tranzitivna ovojnica $\overline{\mathbf{R}}$ aciklične relacije $\mathbf{R}$ je aciklična.
Relacija Q je *ogrodje* relacije $\mathbf{R}$ ntk. je $Q \subseteq \mathbf{R}$, $\overline{Q} = \overline{\mathbf{R}}$ in je relacija Q minimalna – iz nje ne moremo odstraniti nobene povezave, ne da bi 'pokvarili' drugo enakost.
Za splošne relacije (grafe) lahko obstaja več ogrodij; za aciklične pa je enolično določeno $Q = \mathbf{R} \setminus \mathbf{R} * \overline{\mathbf{R}}$.

Globina

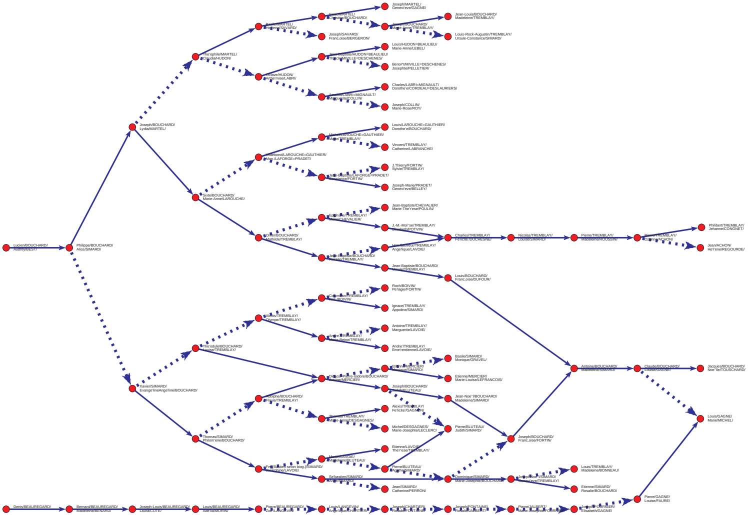


Preslikava $h : \mathcal{V} \to \mathbb{N}^+$ je *globina*, če so vse razlike na najdaljši poti in vrednost v začetku enake 1.

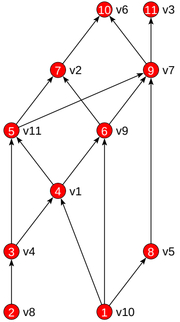
$\mathcal{U} \leftarrow \mathcal{V}$; $k \leftarrow 0$
while $\mathcal{U} \neq \emptyset$ **do**
  $\mathcal{T} \leftarrow \mathrm{Min}_R(\mathcal{U})$; $k \leftarrow k + 1$
  for $v \in \mathcal{T}$ **do** $h(v) \leftarrow k$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \mathcal{T}$

Risanje po ravneh. Macro `Layers`. Druge globine. Algoritem Sugiyama.

Prikaz parnega grafa Bouchardovega rodovnika



Pravilna oštevilčenja

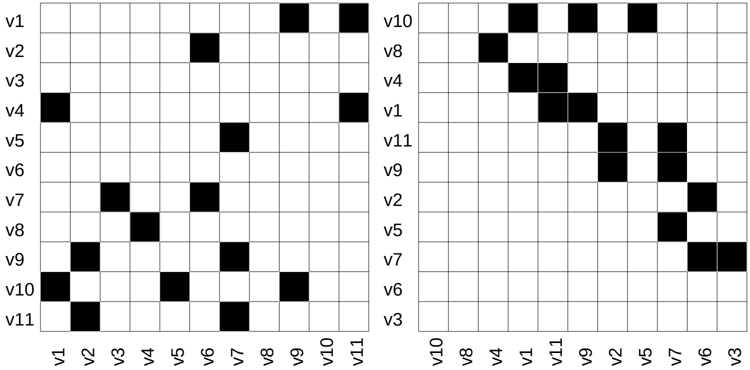


Injektivna, z relacijo **R** usklajena preslikava $h : \mathcal{V} \to 1..|\mathcal{V}|$ je *pravilno oštevilčenje*. 'Topološko urejanje'

$\mathcal{U} \leftarrow \mathcal{V}$; $k \leftarrow 0$
while $\mathcal{U} \neq \emptyset$ **do**
  izberi $v \in \mathrm{Min}_{\mathbf{R}}(\mathcal{U})$; $k \leftarrow k + 1$
  $h(v) \leftarrow k$
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v\}$

Matrični prikaz acikličnega omrežja glede na pravilno oštevilčenje ima ničelni spodnji trikotnik.

. . . Pravilna oštevilčenja




```
read or select  [Acyclic.paj]
Network/Acyclic Natwork/Depth Partition/Acyclic
Partition/Make Permutation
File/Network/Export as Matrix to EPS/Using Permutation [acy.eps]
```

Pravilna oštevilčenja in izračun vrednosti

Naj bo funkcija $f : \mathcal{V} \to \mathbb{R}$ definirana na sosedih takole:

- $f(v)$ je znana za $v \in \mathrm{Min}_{\mathbf{R}}(\mathcal{V})$
- $f(v) = F(\{f(u) : u\mathbf{R}v\})$

Če vrednosti funkcije f računamo v vrstnem redu določenem z nekim pravilnim oštevilčenjem, dobimo vse vrednosti v enem prehodu – saj so za vsak $v \in \mathcal{V}$ vrednosti, ki jih potrebujemo pri izračunu $f(v)$ že določene.

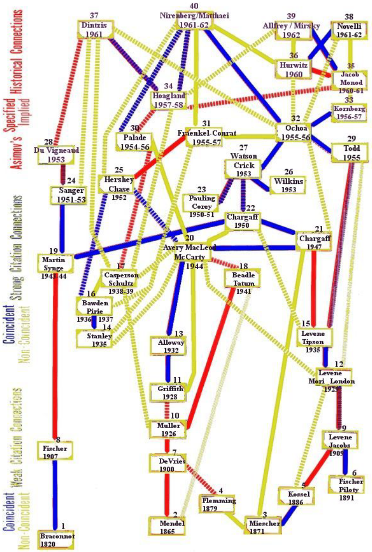
Pravilna oštevilčenja – primer CPM

CPM (Critical Path Method): Projekt je sestavljen iz posameznih opravil. Vozlišča predstavljajo stanja projekta, povezave pa opravila. Za vsako opravilo (u, v) poznamo čas njegovega trajanja $t(u, v)$. Neko opravilo se lahko začne izvajati šele, ko so vsa opravila, ki se končajo v njegovem začetku zaključena. Projektno omrežje je aciklično. Zanima nas, koliko najmanj časa je potrebno za izvedbo projekta.
Naj bo $T(v)$ čas najzgodnejšega zaključka vseh opravil v stanju v.

$$T(v) = 0, \qquad v \in \mathrm{Min}_{\mathbf{R}}(\mathcal{V})$$
$$T(v) = \max_{u:u\mathbf{R}v}(T(u) + t(u, v))$$

`Network/Acyclic Network/Critical Path Method - CPM`

Omrežja sklicevanj



Analiza omrežij sklicevanj se je pričela leta 1964 s članko Garfield et al. Leta 1989 sta Hummon in Doreian predlagala tri mere pomembnosti – uteži na povezavah, ki omogočajo računalniško določitev (naj)pomembnejših delov omrežja sklicevanj. Za dve izmed njih obstajata zelo učinkovita postopka za njun izračun.

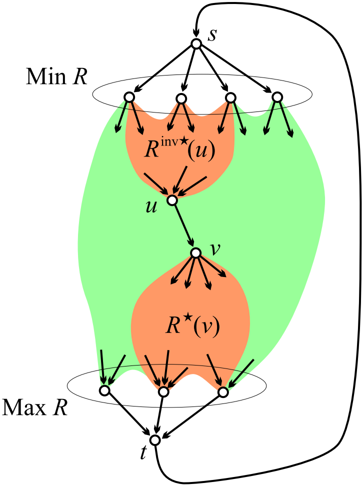
... omrežja sklicevanj

V dani množici enot/vozlišč $\mathcal{U}$ (članki, knjige, druga dela, itd.) vpeljemo *relacijo sklicevanja*/množico usmerjenih povezav $\mathbf{R} \subseteq \mathcal{U} \times \mathcal{U}$

$$u\mathbf{R}v \equiv v \text{ se sklicuje na } u$$

ki določa *omrežje sklicevanj* $\mathcal{N} = (\mathcal{U}, \mathbf{R})$.
Relacija sklicevanja je običajno *irrefleksivna* (ni zank) in (skoraj) *aciklična*. V nadaljevanju bomo privzeli, da ima ti dve lastnosti.
V dejanskih omrežjih sklicevanj se včasih pojavijo manjše (praviloma 2 ali 3 vozlišča) krepke komponente. Tako omrežje najenostavneje pretvorimo v aciklično tako, da skrčimo krepke komponente in odvržemo zanke, ki pri tem nastanejo. Obstajajo tudi druge možnosti.
Omrežje sklicevanj je koristno dopolniti v *standardno* obliko, tako da mu dodamo skupni *izvor* $s \notin \mathcal{U}$ in skupni *ponor* $t \notin \mathcal{U}$. Izvor s je neposredno povezan v vse minimalne elemente relacije $\mathbf{R}$; ponor t pa iz vseh maksimalnih elementov relacije $\mathbf{R}$. Dodamo še *povratno* povezavo (t, s).

Štetje poti – Search path count method



Metoda *štetja poti* – *search path count method* (SPC) temelji na števcih $n(u, v)$, ki štejejo število različnih poti iz izvora s v ponor t, ki gredo čez povezavo (u, v). Za izračun števcev $n(u, v)$ vpeljemo dve pomožni količini: $n^-(v)$ šteje število poti iz izvora s v vozlišče v, in $n^+(v)$ šteje število poti iz vozlišča v v ponor t.

Hitri postopek za izračun števila poti

Iz osnovnih načel kombinatorike izhaja

$$n(u,v) = n^-(u) \cdot n^+(v), \qquad (u,v) \in \mathbf{R}$$

kjer je

$$n^-(u) = \begin{cases} 1 & u = s \\ \sum_{v:v\mathbf{R}u} n^-(v) & \textit{sicer} \end{cases}$$

in

$$n^+(u) = \begin{cases} 1 & u = t \\ \sum_{v:u\mathbf{R}v} n^+(v) & \textit{sicer} \end{cases}$$

Ta zveza je osnova za hiter izračun števcev $n(u,v)$ – najprej vozlišča grafa topološko uredimo in nato uporabljamo v dobljenem vrstnem redu gornjo zvezo. Postopek ima časovno zahtevnost reda $O(m)$, $m = |\mathbf{R}|$. Topološka urejenost zagotavlja, da so v gornji zvezi vse količine že izračunane, ko jih potrebujemo.

Hummon in Doreian-ove uteži in SPC

Hummon in Doreian uteži so takole določene:

- *search path link count* (SPLC) method: utež $w_l(u, v)$ je enaka številu "*vseh možnih poti po omrežju, ki imajo začetek v izvoru*" in gredo po povezavi $(u, v) \in \mathbf{R}$.
- *search path node pair* (SPNP) method: $w_p(u, v)$ "*upošteva vse povezanosti parov vozlišč s potmi, ki gredo po povezavi* $(u, v) \in \mathbf{R}$".

Uteži SPLC lahko določimo tako, da uporabimo postopek SPC na omrežju, ki ga dobimo, če v standardni obliki omrežja povežemo izvor s s povezavo še z vsemi neminimalnimi vozlišči iz $\mathcal{U}$; za uteži SPNP moramo temu omrežju dodati še povezave iz vseh nemaksimalnih vozlišč iz $\mathcal{U}$ v ponor t.

Vozliščne uteži

Količine uporabljene za določitev povezavnih uteži w lahko uporabimo tudi za določitev ustreznih vozliščnih vrednosti t

$$
\begin{aligned}
t_c(u) &= n^-(u) \cdot n^+(u) \\
t_l(u) &= n_l^-(u) \cdot n_l^+(u) \\
t_p(u) &= n_p^-(u) \cdot n_p^+(u)
\end{aligned}
$$

Te štejejo število poti izbrane vrste skozi vozlišče u.
V programu `Pajek` dobimo hkrati uteži w in lastnost t.

```
Network/Acyclic Network/Citation Weights/Search Path Count (SPC)
Network/Acyclic Network/Citation Weights/Search Path Link Count (SPLC)
Network/Acyclic Network/Citation Weights/Search Path Node Pair (SPNP)
```

Lastnosti uteži SPC

Vrednosti števcev $n(u, v)$ določajo tok po omrežju sklicevanj – zanj velja *Kirchoffov vozliščni zakon*: v vsakem vozlišču u standardnega omrežja sklicevanj velja *vstopni tok* = *izstopni* tok:

$$\sum_{v:v\mathbf{R}u} n(v, u) = \sum_{v:u\mathbf{R}v} n(u, v) = n^-(u) \cdot n^+(u)$$

Utež $n(t, s)$ je enaka celotnemu toku skozi omrežje in ponuja naravni način normalizacije uteži

$$w(u, v) = \frac{n(u, v)}{n(t, s)} \quad \Rightarrow \quad 0 \le w(u, v) \le 1$$

in, če je C minimalni povezavni prerez, velja še $\sum_{(u,v)\in C} w(u, v) = 1$.
V velikih omrežjih lahko uteži postanejo zelo velike. Na to je potrebno biti pazljiv pri programiranju algoritma.

Analiza omrežij

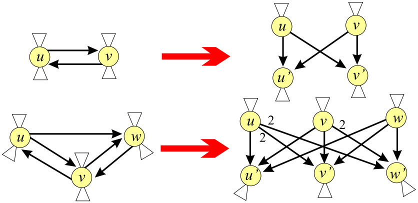
V. Batagelj

Omrežja sklicevanj s cikli

Če v omrežju obstaja cikel, obstaja med nekaterimi vozlišči neskončno število sprehodov. Nastali problem lahko poskušamo rešiti na več načinov: z vpeljavo 'staranja', ki zagotovi, da celotna utež sprehodov gre proti neki končni vrednosti; ali omejimo definicijo uteži na neko končno podmnožico sprehodov – npr. poti ali najkrajše poti. Toda to ustvari nova vprašanja: Kolikšen naj bo faktor 'staranja'? Ali je mogoče učinkovito prešteti (najkrajše) poti?
Druga možnost je, ker so omrežja sklicevanj skoraj aciklična, da jih pretvorimo v aciklična:

- skrčimo vsako ciklično skupino (netrivialno krepko komponento) v vozlišče, ali
- razklenemo cikle z odstranitvijo nekaj povezav, ali
- z uporabo ustreznih transformacij (glej naslednjo prosojnico).

Transformacija Preprint



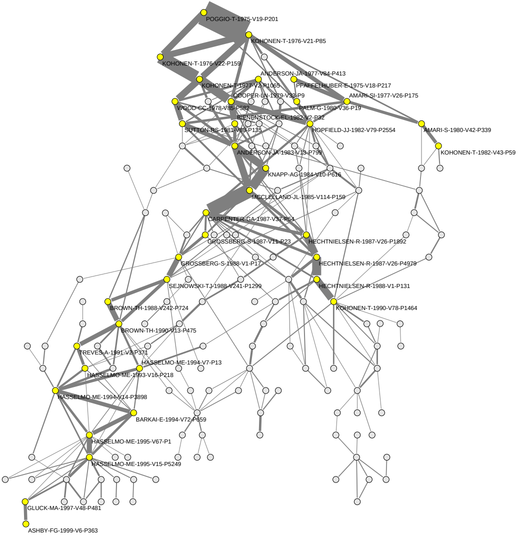
Transformacija *preprint* temelji na naslednji zamisli: Vsako vozlišče (delo) iz krepke komponente se podvoji s svojo predobjavo (preprint). Dela znotraj komponente se sklicujejo na predobjave.

Velike krepke komponente v dejanskih omrežjih sklicevanj običajno pomenijo napako v podatkih. Seveda je to odvisno od načina izgradnje omrežja. Tako, na primer ima omrežje HEP – High Energy Particle Physics z **arXiv** veliko večjih krepkih komponent, ker so kot ena enota obravnavane vse različice istega članka. Tudi v tem omrežju bi si lahko pomagali s transformacijo 'preprint'.

Omrežje SOM

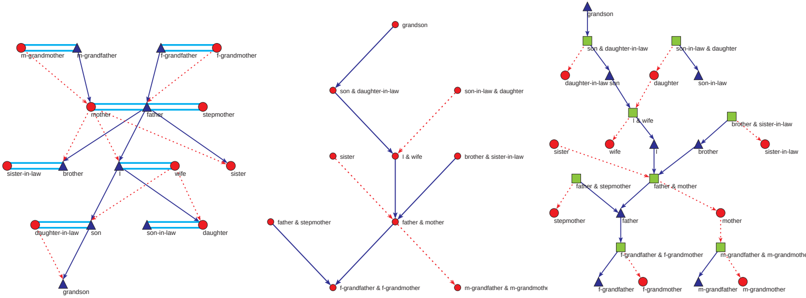
Poklejmo si omrežje sklicevanj za področje ($n = 4470$, $m = 12731$) SOM (*self-organizing maps*). Določimo uteži SPC.
Pri pregledu porazdelitve vrednosti uteži smo izbrali prag 0.007 in določili pripadajoči povezavni prerez. Odstranimo še vse male komponente ($k = 5$). Ostane ena sama komponenta. Narišemo jo po plasteh in ročno popravimo sliko.
Na sliki označimo samo pomembnejša dela – krajišča povezav z utežjo vsaj 0.05.
S slike vidimo, da v razvoju področja SOM obstajata vsaj dve smeri.

Omrežje SOM – povezavni prerez 0.007



Rodovniki

Naslednji primer acikličnih omrežij so rodovniki. V poglavju 'Omrežja vsepovsod' smo že opisali naslednje tri omrežne predstavitve rodovnikov:



Orejev graf, parni graph, in dvodelni parni graph

Mormoni, Škofja Loka, ameriški predsedniki; računalništvo, matematika.

Primerjava predstavitev rodovnikov

Parni grafi imajo več prednosti (White et al., 1999):

- v parnem grafu je manj vozlišč in povezav kot v Orejevem grafu;
- parni grafi so usmerjeni aciklični grafi;
- vsaka sklenjena veriga v parnem grafu pomeni *poroko med sorodniki*. Obstajata dve vrsti teh porok: *poroke med krvnimi sorodniki*: npr. poroka med bratrancem in sestrično. *poroke med nekrvnimi sorodniki*: npr. brata se poročita s sestrama iz druge družine.
- parni grafi so veliko prikladnejši za analize.

Dvodelni parni grafi natančneje opisujejo rodovnik – npr. omogočajo razlikovati dve poroki enega izmed bratov od enkratnih porok vsakega izmed njiju; omogočajo razkriti poroke med polbrati in polsestami.

Rodovniki so redka omrežja

Rodovnik je *navaden*, če ima vsaka oseba največ dva starša. Rodovniki so *redka* omrežja – število povezav je istega reda kot število vozlišč.
Za *navaden Orejev rodovnik* velja ($\mathcal{V}$ – vozlišča, $\mathcal{A}$ – usmerjene povezave, $\mathcal{E}$ – neusmerjene povezave):

$$|\mathcal{A}| \leq 2|\mathcal{V}|, \quad |\mathcal{E}| \leq \frac{1}{2}|\mathcal{V}|, \quad |\mathcal{L}| = |\mathcal{A}| + |\mathcal{E}| \leq \frac{5}{2}|\mathcal{V}|$$

Parni grafi so skoraj drevesa – odstopanja od dreves nastanejo zaradi porok med sorodniki ($\mathcal{V}_p$, $\mathcal{A}_p$ – vozlišča in povezave parnega grafa):

$$|\mathcal{V}_p| = |\mathcal{V}| - |\mathcal{E}| + n_{mult}, \quad |\mathcal{V}| \geq |\mathcal{V}_p| \geq \frac{1}{2}|\mathcal{V}|, \quad |\mathcal{A}_p| \leq 2|\mathcal{V}_p|$$

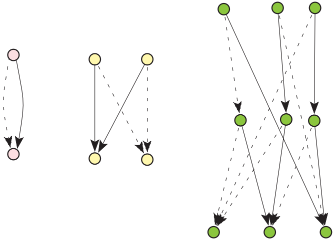
Za dvodelni parni graf pa velja:

$$|\mathcal{V}| \leq |\mathcal{V}_b| \leq \frac{3}{2}|\mathcal{V}|, \quad |\mathcal{A}_b| \leq 2|\mathcal{V}| + n_{mult}$$

Število vozlišč in povezav v Orejevih in parnih grafih

data	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}\rvert$	$\frac{\lvert\mathcal{L}\rvert}{\lvert\mathcal{V}\rvert}$	$\lvert\mathcal{V}_i\rvert$	n_{mult}	$\lvert\mathcal{V}_p\rvert$	$\lvert\mathcal{A}_p\rvert$	$\frac{\lvert\mathcal{A}_p\rvert}{\lvert\mathcal{V}_p\rvert}$
Drame	29606	8256	41814	1.69	13937	843	22193	21862	0.99
Hawlina	7405	2406	9908	1.66	2808	215	5214	5306	1.02
Marcus	702	215	919	1.62	292	20	507	496	0.98
Mazol	2532	856	3347	1.66	894	74	1750	1794	1.03
President	2145	978	2223	1.49	282	93	1260	1222	0.97
RoyalE	17774	7382	25822	1.87	4441	1431	11823	15063	1.27
Loka	47956	14154	68052	1.71	21074	1426	35228	36192	1.03
Silba	6427	2217	9627	1.84	2263	270	4480	5281	1.18
Ragusa	5999	2002	9315	1.89	2347	379	4376	5336	1.22
Tur	1269	407	1987	1.89	549	94	956	1114	1.17
Royal92	3010	1138	3724	1.62	1003	269	2141	2259	1.06
Little	25968	8778	34640	1.67	8412				1.01
Mumma	34224	11334	45565	1.66	11556				1.00
Tilltson	42559	12796	54043	1.57	16967				1.00

Prepletenost



Naj bo n število vozlišč v parnem grafu, m število povezav, k število šibkih komponent, in M število koncev (maksimalnih vozlišč – vozlišč z izhodno stopnjo 0, $M \geq 1$).
Prepletenost rodovnika imenujemo število:

$$RI = \frac{k + m - n}{k + n - 2M}$$

Za grafe brez povezav postavimo $RI = 0$.

Velja $0 \leq RI \leq 1$. $RI = 0$ natanko takrat, ko je graf gozd. Obstajajo poljubno veliki rodovniki z $RI = 1$.

Iskanje vzorcev

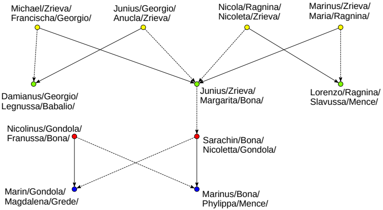
Če se izbrani *vorec* določen z danim (majhnim) grafom ne pojavlja pogosto v redkem grafu, lahko njegove pojavitve razmeroma učinkovito poiščemo s preprostim pregledom vseh možnosti.

Iskanje lahko pospešimo tako, da upoštevamo dodatne lastnosti vzorca:

- vozlišča vzorca in omrežja se morajo ujemati tudi v vrednosti izbrane lastnosti (npr. vrsta atoma v molekuli);
- ujemati se morajo uteži na povezavah (npr. moške/ženske povezave v *parnih grafih*);
- prvo vozlišče iz vzorca pripada dani skupini vozlišč.

Poroke med sorodniki v Ragusi

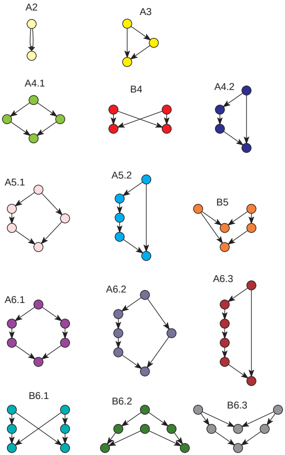
Iskanje vzorcev smo uporabili npr. pri analizi organskih molekul (iskanje ogljikovih obročev) in pri analizi porok med sorodniki v rodovnikih.



Slika prikazuje tri povezane skupine porok med sorodniki v rodovniku dubrovniškega (Ragusa) plemstva. Rodovnik je predstavljen kot parni graf. Polne povezave predstavljajo odnos _ *je sin od* _; pikčaste povezave pa odnos _ *je hči od* _. V vseh treh skupinah sta se brat in sestra iz ene družine poročila s sestro in bratom iz druge družine.

`Networks/Fragment (First in Second)/Find`

Poroke med sorodniki (parni grafi na 2 do 6 vozliščih)



Na sliki so prikazani vsi mogoči vzorci porok med sorodniki na 2 do 6 vozliščih v parnih grafih. Oznake imajo naslednji pomen:

- prvi znak – število začetkov: A – en, B – dva, C – tri.
- drugi znak: število vozlišč v vzorcu (2, 3, 4, 5, ali 6).
- tretji znak: različica, če sta prva znaka enaka.

V vsakem vzorcu je število začetkov enako številu koncev. Poroke med krvnimi sorodniki imajo en začetek in en konec – vse imajo oznako A.

Primerjava rodovnikov

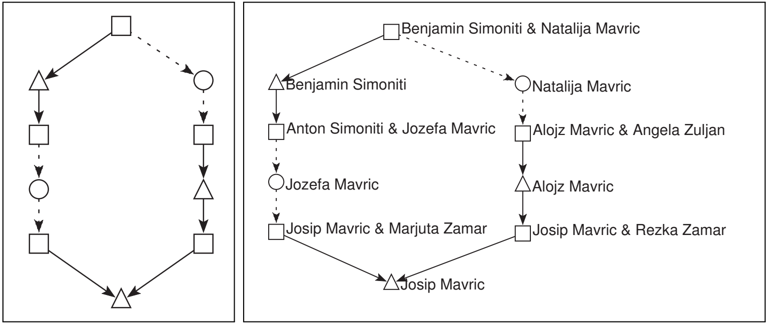
Rodovnike lahko primerjamo, tako da primerjamo porazdelitve vzorcev v njih. Za primer smo vzeli naslednje rodovnike: `Loka.ged` (Škofja Loka), `Silba.ged` (otok Silba, Hrvaška), `Ragusa.ged` (dubrovniško plemstvo med 12. in 16. stoletjem, Dremelj et al., 2002) `Turcs.ged` (turški nomadi, White et al., 1999), `RoyalE.ged` (evropske kraljevske rodbine).
Vidimo:
generacijski preskok za več kot eno generacijo je (skoraj) neverjeten – vzorci A4.2, A5.2 in A6.3 se ne pojavljajo v rodovnikih, vzorec A6.2 se pojavi 2 krat na Silbi, vzorec B6.4 se pojavi 5 krat v Ragusai in 3 krat pri turkih).
Pri turkih je zelo veliko porok vrst A4.1 in A6.1.
V vseh rodovnikih je število porok med nekrvnimi (vzorci B4, B5, C6, B6.1, B6.2, B6.3 in B6.4) sorodniki veliko večje od števila porok med krvnimi sorodniki. Razlogi za poroke med sorodniki so pogosto pogojeni s prizadevanji za ohranitev premoženja znotraj izbranih rodbin.

Normalizirane pogostosti vzorcev $\times$ 1000

pattern	Loka	Silba	Ragusa	Turcs	RoyalE
A2	0.07	0.00	0.00	0.00	0.00
A3	0.07	0.00	0.00	0.00	2.64
A4.1	0.85	2.26	1.50	**159.71**	18.45
B4	3.82	11.28	10.49	**98.28**	6.15
A4.2	0.00	0.00	0.00	0.00	0.00
A5.1	0.64	3.16	2.00	36.86	11.42
A5.2	0.00	0.00	0.00	0.00	0.00
B5	1.34	4.96	23.48	46.68	7.03
A6.1	1.98	12.63	1.00	**169.53**	11.42
A6.2	0.00	0.90	0.00	0.00	0.88
A6.3	0.00	0.00	0.00	0.00	0.00
C6	0.71	5.41	9.49	36.86	4.39
B6.1	0.00	0.45	1.00	0.00	0.00
B6.2	1.91	17.59	31.47	**130.22**	10.54
B6.3	3.32	13.53	40.96	**113.02**	11.42
B6.4	0.00	0.00	2.50	7.37	0.00
Sum	14.70	72.17	123.88	798.53	84.36

Zelo pogoste se poroke med sorodniki pri turkih; sledijo jim dubrovčani.

Dvodelni parni grafi: poroka med polbratrancem in polsestrično



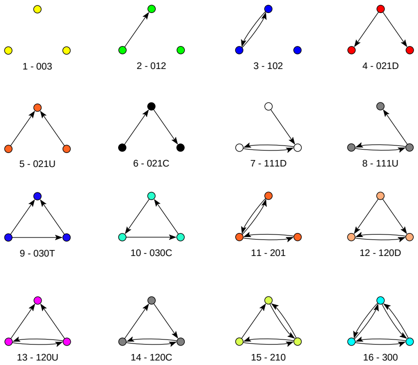
... rodovniki – druga vprašanja

Rodoslovce zanimajo še druga vprašanja:

- spremembe v pogostosti vzorcev skozi čas;
- posebnosti: velikokrat poročene osebe, osebe z veliko otrok;
- preverjanje sorodstvene povezanosti med osebama in iskanje najkrajše poti po rodovniku, če obstaja;
- določitev vse prednikov/potomcev izbrane osebe; določitev osebe, ki ima največ prednikov/potomcev;
- največja razlika v letih med možem in ženo; najstarejša/najmlajša oseba pri poroki; najstarejša/najmlajša oseba ob rojstvu otroka; . . .
- določitev najdaljše moške/ženske verige;
- odkrivanje napak pri vnosu podatkov (preverjanje pravilnosti omrežja).

Na vsa ta vprašanja je mogoče odgovoriti z orodji programa `Pajek`.

Trojice



Naj bo $\mathcal{G} = (\mathcal{V}, \mathbf{R})$ enostaven usmerjen graf brez zank. *Trojica* je podgraf porojen z izbranimi tremi vozlišči.
Obstaja 16 neizomorfnih (vrst) trojic, ki so naprej razbite na tri skupine:

- the *ničelna* trojica 003;
- *dvočleni* trojici 012 in 102; ter
- *povezane* trojice: 111D, 201, 210, 300, 021D, 111U, 120D, 021U, 030T, 120U, 021C, 030C in 120C.

Trojiški spekter

Več lastnosti grafa je mogoče izraziti z uporabo *trojiškega spektra* – porazdelitve vseh njegovih trojic. Ta je tudi osnovna sestavina za modele omrežij p^*. Običajni postopek za določitev trojiškega spektra je reda $O(n^3)$; no, v večini velikih omrežij ga je mogoče določiti precej hitreje. Algoritem temelji na naslednjem opažanju: *v velikem in redkem omrežju je večina trojic ničelnih*. Označimo s T_1, T_2, T_3 zaporedoma število ničelnih, dvočlenih in povezanih trojic. Ker je celotno število trojic enako $T = \binom{n}{3}$ in so zgornje skupine razbitje množice vseh trojic, je zamisel postopka naslednja:

- preštej vse dvočlene T_2 in vse povezane T_3 trojice z njihovimi podvrstami;
- število ničelnih trojic izračunaj po zvezi $T_1 = T - T_2 - T_3$.

... Trojiški spekter

V izvedbi postopka moramo zagotoviti, da bo vsaka neničelna trojica šteta natanko enkrat pri prehodu množice povezav. Vozlišča trojice $\{v, u, w\}$ lahko v splošnem izberemo na 6 načinov (v, u, w), (v, w, u), (u, v, w), (u, w, v), (w, v, u), (w, u, v). Problem izomorfizma lahko rešimo z uvedbo *kanonskih* izbir, ki prispevajo k štetju; ostale, nekanonske, lahko preskočimo.
Celotna zahtevnost tega algoritma je reda $O(\hat{\Delta} m)$ in potemtakem, za grafe z majhno največjo stopnjo $\hat{\Delta} \ll n$, ker je $2m \leq n\hat{\Delta}$, reda $O(n)$.

Trojice

Triad:	BA	CL	RC	R2C	TR	HC	39+	p1	p2	p3	p4
003		✓	✓		✓	✓				✓	✓
012					✓	✓	✓			✓	✓
102	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
021D			✓	✓	✓	✓	✓				✓
021U			✓	✓	✓	✓	✓			✓	✓
021C									✓		✓
111D											✓
111U								✓	✓		
030T			✓	✓	✓	✓	✓		✓		✓
030C								✓	✓		✓
201											
120D			✓	✓	✓	✓	✓			✓	✓
120U			✓	✓	✓	✓	✓	✓	✓		✓
120C							✓		✓		
210						✓	✓		✓		
300	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Triad Micro-Models:
BA: Ballance (Cartwright and Harary, '56)
CL: Clustering Model (Davis. '67)
RC: Ranked Cluster (Davis & Leinhardt, '72)
R2C: Ranked 2-Clusters (Johnsen, '85)
TR: Transitivity (Davis and Leinhardt, '71)
HC: Hierarchical Cliques (Johnsen, '85)
39+: Model that fits D&L's 742 mats N :39-72
p1-p4: Johnsen, 1986. Process Agreement Models.

Moody