Mrežni deskriptori i kurikulne mreže

Antunović, Suzana

Sveučilište u Zagrebu

ZAJEDNIČKI SVEUČILIŠNI POSLIJEDIPLOMSKI

DOKTORSKI STUDIJ MATEMATIKE

SUZANA ANTUNOVIĆ

MREŽNI DESKRIPTORI I KURIKULNE MREŽE

DOKTORSKI RAD

Mentor:

prof. dr. sc. Damir Vukičević

Zagreb, 2017

University of Zagreb

CROATIAN DOCTORAL PROGRAM IN

MATHEMATICS

Suzana Antunović

NETWORK DESCRIPTORS AND CURRICULUM NETWORKS

DOCTORAL THESIS

Supervisor:

prof. dr. sc. Damir Vukičević

Zagreb, 2017

ZAHVALE

Veliku zahvalu upućujem mentoru prof.dr.sc. Damiru Vukičeviću koji me upoznao sa svijetom znanosti, pomagao mi svojim idejama, savjetima i razumijevanjem i vodio me kroz proces nastajanja ove disertacije. Zahvaljujem se i članovima stručnog povjerenstva, prof.dr.sc. Dragutinu Svrtanu i prof.dr.sc. Tomislavu Došliću, te kolegama sa Seminara za diskretnu matematiku na savjetima, primjedbama i idejama. Posebno hvala upućujem Ivanu Marušiću na nezamjenjivoj tehničkoj podršci i pomoći.

Najveća hvala mojim roditeljima koji su me pratili na svakom koraku mog privatnog i profesionalnog sazrijevanja. Bez njihove potpore i utjehe vjerojatno ne bi bilo ni ovoga rada. Hvala svim mojim prijateljima i dragim ljudima koji su bili uz mene i bodrili me tijekom ovog procesa.

Konačno, ovaj rad posvećujem osobi s kojoj sam posljednje četiri godine dijelila strahove, strepnje, umor, iscrpljnost, olakšanje, probleme, uspjehe i neuspjehe, osobi koju sam zbog njegove snage i volje doživjela u potpuno novom svjetlu i na koju sam jako ponosna. Veselim se što ćemo svoje velike životne uspjehe moći proslaviti zajedno. Robi, ovaj rad je za tebe.

Sažetak

U disertaciji su izloženi rezultati istraživanja iz više područja teorije mreža. U prvom dijelu proučavane su ekstremalne vrijednosti poopćenih mrežnih deskriptora transmisije, međupoloženosti, vršne produktivnosti i vršne profitabilnosti. U dva promatrana slučaja uzeta je u obzir pretpostavka da vrhovi u mreži koji se nalaze na većim udaljenostima komuniciraju manje od onih na manjim udaljenostima. U prvom slučaju količina komunkacije među vrhovima utežena je s $d(u,v)^{\lambda}$ za $\lambda < 0$, a u drugom slučaju s $\lambda^{d(u,v)}$ za $\lambda \in (0,1)$, pri čemu je $d(u,v)$ udaljenost između vrhova u i v. Analizirane su gornje i donje ograde vrijednosti deskriptora. Nadalje, definirane su kurikulne mreže i analizirana neka njihova svojstva. Riječ je o jednostavnim usmjerenim grafovima u kojima vrhovi predstavljaju edukacijske jedinice, a usmjereni brid između dva vrha označava da je poznavanje jedne jedinice potrebno za učenje druge. Definirane su mjere pomoću kojih je moguće napraviti evaluaciju valjanosti redoslijeda edukacijskih jedinica. Mjere su analizirane s više različitih gledišta i određeni su grafovi koji odgovaraju najmanje i najviše složenim nastavnim planovima. Predložen je algoritam za određivanje optimalne ekspozicije edukacijskih jedinica u odnosu na odabranu mjeru složenosti. Konačno, detaljno je analiziran problem detekcije zajednica u kurikulnim mrežama i predloženi su algoritmi za rješavanje tog problema.

Abstract

In this thesis results of research from several different areas of complex networks theory have been introduced. First, generalized versions of network descriptors such as transmission, betweeness centrality, networkness and
network surplus have been defined and analyzed. Two different approaches that were studied both take into account the assumption that vertices in networks, which are closer to each other, communicate more than the vertices on greater distances. In the first case the amount of communication between vertices has been amended by $d(u,v)^{\lambda}$ where $\lambda < 0$ and in the second case by $\lambda^{d(u,v)}$, where $d(u,v)$ represents the distance between vertices u and v, and $\lambda \in (0,1)$. Upper and lower bounds for minimal and maximal values of these descriptors have been analyzed. Furthermore, special kind of networks has been defined. Curriculum network is a simple, directed graph in which vertices represent educational units, and directed arc from u to v indicates that understanding of a unit u is necessary for learning and understanding of unit v. Measures for evaluating the quality of curriculum content sequencing have been proposed and analyzed from sever different points of view. Graphs corresponding to most and least complex curricula have been found. Some real–world curriculum networks and their properties have been analyzed.
Finally, the problem of community detection in curriculum networks has been analyzed and two algorithms for solving this problem have been suggested.

Sadržaj

Poglavlje 1

UVOD

Proučavanje kompleksnih mreža temelji se na konceptima teorije grafova. Mreža je skup vrhova povezanih bridovima. Brid između dva vrha implicira postojanje nekog odnosa ili veze između dva povezana vrha [115, 116]. Često se pojam mreža koristi kao sinonim za graf iako se formalno pojam graf odnosi na sam matematički objekt, dok se pojam mreža odnosi na realizaciju tog objekta u stvarnom svijetu. Mreža je pojednostavljena reprezentacija koja reducira sustav na apstraktnu strukturu zadržavajući pritom samo neke osnovne podatke o početnom sustavu. Vrhovi i bridovi mogu biti označeni nekim dodatnim informacija da bi se sačuvalo više informacija o sustavu te mogu predstavljati različite entitete i njihovu povezanost ovisno o problemu koji želimo analizirati. Tako, na primjer, u društvenim mrežama, vrhovi mogu predstavljati osobe, a bridovi poznanstvo ili prijateljstvo dviju osoba, međusobnu suradnju, neprijateljstvo, geografsku blizinu i slično. U tehnološkim mrežama vrhovi mogu predstavljati računala, a bridovi fizičku povezanost kabelom. U biologiji, točnije prehrambenim mrežama, vrhovi predstavljaju vrstu životinja, a brid između dvije vrste implicira da je jedna vrsta predator u odnosu na drugu. U računarstvu se često dijagram toka nekog algoritma prikazuje u obliku grafa kojem su vrhovi naredbe (instrukcije), a redoslijed izvršavanja

naredbi prikazan je pomoću bridova. Nadalje, mrežama se mogu reprezentirati i razne računalne strukture podataka, umrežavanje i paralelizam računala i njihov sekvencijalni rad.

Problem koji se često spominje kao začetak teorije grafova je problem königsberških mostova (*eng. The Königsberg Bridge Problem*)[119]. Grad Königsberg u današnjoj Rusiji bio je smješten na obje strane rijeke Pregel i sadržavao je dva velika otoka koji su bili povezani s kopnom i jedan s drugim pomoću sedam mostova. Problem je zahtijevao da se osmisli šetnja gradom tijekom koje bi se svaki most prešao točno jednom. Početna i krajnja točka šetnje nisu nužno morale biti iste. Euler je 1736. dokazao da ovaj problem nema rješenja i tako postavio temelje razvoju teorije grafova.

Slika 1.1: Problem königsberških mostova: **(a)** mapa grada na kojoj je svaki dio kopna označen različitom bojom. **(b)** problem prikazan pomoću grafa u kojem vrhovi predstavljaju dijelove kopna, a bridovi svaki od 7 mostova. (preuzeto iz [83])

Mnogi se problemi i objekti mogu modelirati pomoću kompleksnih mreža pa nerijetko istraživanja o mežama prodiru iz matematike u fiziku [5, 57, 26, 90], biologiju [82, 161], kemiju [166, 123, 130], klimatologiju [142, 172, 52], računalnu znanost, sociologiju [23, 25, 28, 59], epidemiologiju [18, 19, 72] i druge grane. Ideje iz teorije mreža primijenjene su na analizu metaboličkih [69] i genetskih regulatornih mreža, modeliranje i vizualizaciju velikih bežičnih komunikacijskih mreža, razvoj strategija cijepljenja radi kontrole širenja bolesti [111, 112], identifikaciju kritičnih točaka u prometu gdje se stvaraju gužve [6, 13] i u širokom spektru drugih praktičnih problema.

Prvi korak u analiziranju mreže često je vizualizacija. Ona predstavlja koristan alat za analizu podataka jer nam dopušta da odmah uočimo važna strukturalna svojstva mreže koja bi možda bilo teško uočiti da se radi samo o sirovim podacima. Znanstvenici iz različitih grana znanosti su tijekom godina razvili opsežan skup alata, matematičkih, računalnih i statističkih, za analizu, modeliranje i bolje razumijevanje mreža. Budući je većina alata napravljena tako da koristi mreže u njihovom apstraktom obliku (u obliku grafa) mogu se primjeniti na gotovo sve sustave i probleme koje je moguće reprezentirati kao mreže. Mnogi od tih alata počinju s jednostavnom mrežom i nakon određenih kalkulacija dolaze do informacija kao što su, na primjer, koji je vrh najbolje povezan s drugima ili duljina najduljeg puta u mreži. Drugi alati koriste modele mreža za matematička previđanja o procesima koji se događaju u mreži kao što su putevi kojima će teći promet na Internetu ili načini kako će se širiti epidemija neke bolesti [116]. Razvijaju se računalni modeli i programski alati koji imaju za cilj pružanje uvida u razne dinamičke fenomene na mrežama [158]. Primjerice, moždanu aktivnost moguće je proučavati kroz skup neuralnih mreža i interpretirati je u okviru često korištenih mjera strukturalne i funkcionalne povezanosti [134]. Nadalje, predviđanje i analiziranje metaboličkih procesa u velikim biokemijskim mrežama predstavlja izazov za

znanstvenike iz područja bioinformatike i biologije [22]. Računalni i matematički modeli imaju ogromnu ulogu u procjeni i kontroli širenja zaraznih bolesti, kao što je bio slučaj s virusom gripe $H1N1$ 2011. godine [154]. Bolesti se šire kroz mrežu kontakata među pojedincima. Uzorci i učestalost takvih kontakata mogu se prikazati u obliku kompleksne mreže i dosta pažnje je posvećeno empirijskim istraživanjima i proučavanju strukture takvih mreža. Pojavljuje se i nova vrsta zaraze u obliku računalnih virusa, programa koji se sami multipliciraju i prelaze s računala na računalo na sličan način na koji se patogeni šire među ljudima ili životinjama [116]. Bilo da je riječ o širenju bolesti ili prosljeđivanju neke informacije kroz mrežu, identificiranje vrhova koji su najutjecajniji i najaktivniji u svom djelovanju je važan korak prema optimiziranom korištenju resursa ili efikasnom širenju informacija [87]. Nerijetko su za takve situacije razvijeni posebni paketi koji su specijalizirani za konkretne sustave i probleme, primjerice *Brain Connectivity Toolbox* u sklopu programskog alata *MATLAB* ili *GLEaMviz*, javno dostupni programski sustav koji simulira širenje zaraznih bolesti na globalnoj razini [154].

Posljednjih godina svjedoci smo novog trenda u istraživanju mreža jer se fokus s manjih grafova i istraživanja svojstava pojedinih vrhova ili bridova pomiče na statistička i dinamička svojstva većih mreža i grafova s nekoliko stotina tisuća ili milijuna vrhova, temporalne mreže [75, 80], prostorne mreže [20, 31, 46], društvene mreže [137]. Za procvat ove grane znanosti uvelike je zaslužan napredak u polju informacijske i komunikacijske tehnologije koji nam omogućava sakupljanje i razmjenu informacija te brzu i efikasnu analizu prikupljenih podaka, što je rezultiralo podrobnijim proučavanjem modernih mreža sa nekoliko stotina milijuna vrhova kao što su Facebook, Twitter, World Wide Web, Internet i druge.

Sve se više pažnje posvećuje sigurnosti i otpornosti kompleksnih mreža [74, 160]. Otpornost (*eng. resilience*) je sposobnost sustava da prilagodi svoju aktivnost i zadrži funkcionalnost kada se dogode greške ili problemi u radu [43]. Kompleksne

mreže kao što su Internet, financijska tržista, komunikacijske, prometne, električne mreže čine esencijalni dio modernih društava. Poremećaj u radu koji bi poremetio, na primjer, dovod električne energije mogao bi imati teške posljedice, stoga je sigurnost i otpornost takvih mreža na slučajne i namjerne napade od presudne važnosti [113]. Mogućnost rušenja mreže napadom na jedan ili nekoliko vrhova ili bridova jedan je od ključnih problema koji se pokušavaju spriječiti i riješiti. Jednom kada dođe do poremećaja u radu ili protoku informacija kroz mrežu, posljedice za društvo su često velike. Uzmimo za primjer računalni virus "Code Red" koji je 2001. godine zarazio oko 360 tisuća servera, a nastala šteta procijenjena je na više stotina milijuna dolara [7]. Još jedan primjer poremećaja u radu sustava koji je ostavio velike posljedice je nestanak struje koji se dogdio 1996. godine u Sjevernoj Americi. Zbog preopterećenja električne mreže uzrokovanog pretjeranim korištenjem klimatizacijskih uređaja tijekom neuobičajeno toplog ljeta, došlo je do nestanka električne energije u zapadnoj Kanadi, dijelu Sjedinjenih Američkih Država i dijelu sjevernog Meksika. U periodu od skoro mjesec dana nije normaliziran sustav opskrbe što je direktno utjecalo na više od 2 milijuna ljudi [157].

Razvoj interdisciplinarnog pristupa koji se temelji na suradnji znanstvenika iz polja informacijskih znanosti, inženjerstva, statističke i nelinearne fizike, primijenjene matematike i društvenih znanosti donosi nove uvide i koncepte u proučavanje sigurnosti kompleksnih mreža [113].

1.1 Vrste i primjeri kompleksnih mreža

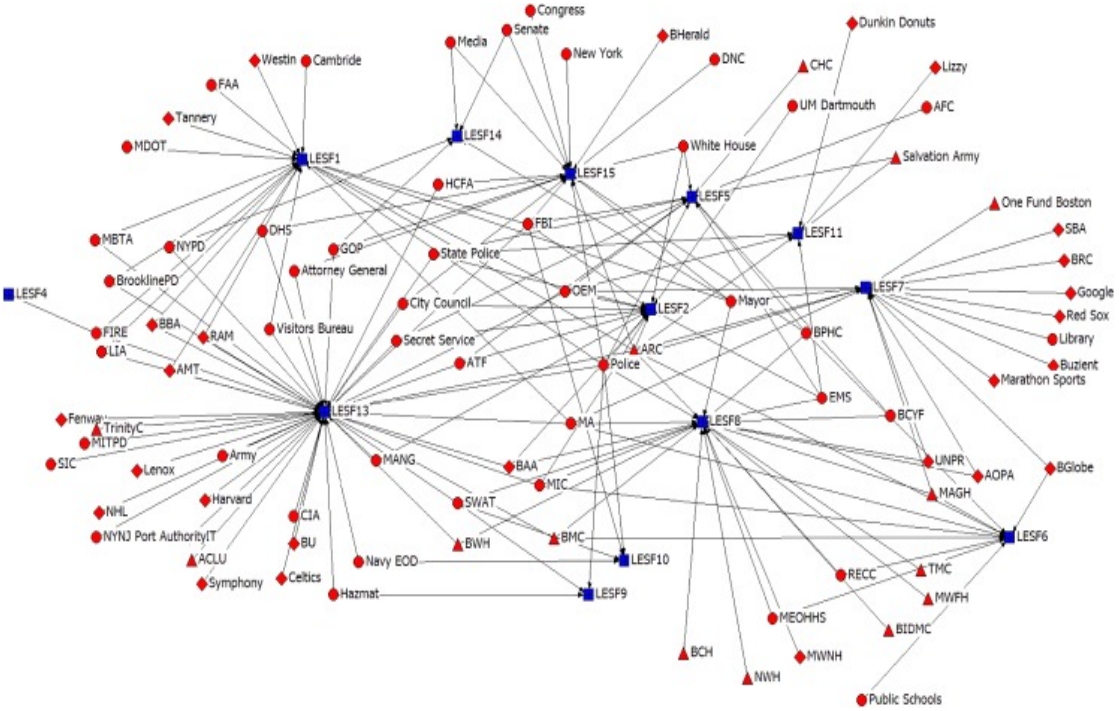
Kompleksne mreže najčešće se dijele u četiri skupine: tehnološke, društvene, informacijske i biološke [116]. U ovom poglavlju definirani su i opisani neki primjeri često proučavanih mreža za svaku od navedenih skupina.

Tehnološke mreže čine osnovu današnjih modernih tehnoloških društava. Zasigurno najpoznatiji primjer tehnološke mreže je Internet. Internet je svjetski raširena mreža fizičkih podatkovnih veza (kabela) među računalima i srodnim uređajima. Njegova struktura nije kontrolirana od strane neke centralne organizacije, sustav se sam organizira kombiniranim djelovanjem mnogih lokalnih i autonomnih računalnih sustava. Poruke se Internetom šalju razlomljene u pakete, a sustav funkcionira tako da ukoliko se u mrežu dodaju novi vrhovi ili bridovi ili ako se izbrišu ili pokvare neki postojeći, usmjernici automatski prilagođavaju rutu slanja paketa u skladu s novim stanjem sustava [116]. Iako je to odlična karakteristika sa strane robusnosti i fleksibilnosti mreže, predstavlja problem u istraživanju strukture Interneta pa se većina zaključaka donosi na temelju eksperimentalnih mjerenja. Neki od značajnijih primjera tehnoloških mreža, osim Interneta, su telefonske mreže, mreže prijenosa električne energije [60], prometne mreže [51, 104] i distribucijske mreže [64, 39]. Telefonska mreža sastoji se od fiksnih i bežičnih veza kojima se prenose telefonski pozivi. Mreža za prijenos električne energije, u ovom kontekstu, je mreža visokonaponskih dalekovoda koji omogućavaju prijenos električne energije. Prometne mreže uključuju zrakoplovne linije, cestovni, željeznički i pomorski promet. Primjer prometne mreže prikazan je na Slici 1.2. Distribucijske mreže uključuju naftovode i plinovode, kanalizacijske linije i rute kojima se koriste dostavljači ili distributeri proizvoda.

Slika 1.2: Mreža brzih željezničkih linija u Europi 2016. godine (preuzeto sa www.wikipedia.com, original PNG: Bernese media)

Društvene mreže su mreže u kojima vrhovi predstavljaju osobe ili ponekad grupe ljudi, a bridovi predstavljaju neku vrstu socijalne interakcije među njima. Zadnjih desetljeća velik razvoj doživjele se online društvene mreže kao što su Facebook i Twitter. Facebook je online društvena mreža u kojoj su vrhovi osobe, točnije njihovi profili ili mrežne stranice, a veze se stvaraju ukoliko dvije osobe definiraju svoj odnos kao prijateljstvo [153]. Teško je odrediti točan broj vrhova u mreži pošto se on brzo mijenja, ali po posljednjim procjenama Facebok je u prosincu 2016. godine imao oko 1860 milijuna aktivih korisnika [141]. Posebna vrsta društvenih mreža su mreže pripadnosti (*affiliation networks*) u kojima vrhovi predstavljaju osobe koje su povezane bridom ako pripadaju istoj društvenoj grupi.

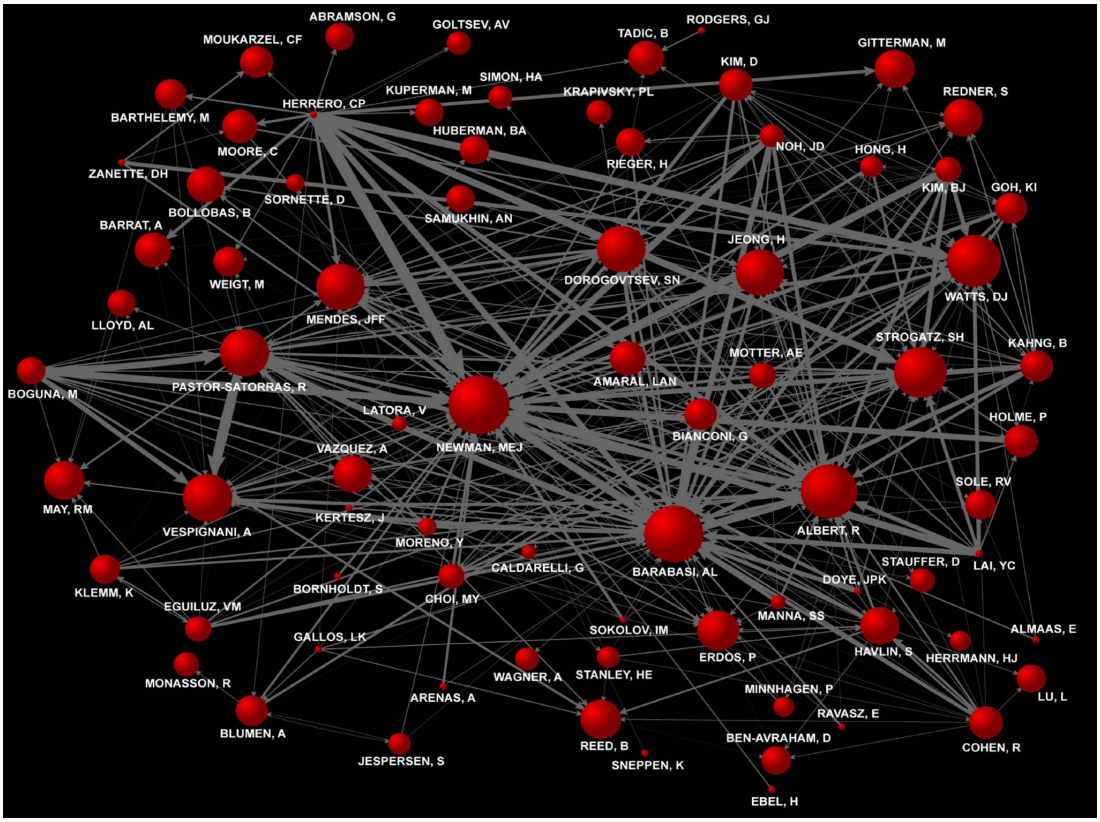
Jedna od takvih je mreža suradnje državnih i lokalnih sigurnosnih i zdravstvenih institucija u sklopu Bostonskog sustava za upravljanje u nuždi prikazana na Slici 1.3. U akademskim krugovima, vrlo je važna mreža koautorstva u kojoj vrhovi predstavljaju znanstvenika ili grupu znanstvenika koji su povezani bridom ako su zajedno objavili znanstveni članak [14, 40]. Veliki doprinos istraživanju društvenih mreža dao je Stanley Miligram svojim eksperimentom u kojem je pokazao "efekt malog svijeta", svojstvo kompleksne mreže prema kojem je prosječna najkraća udaljenost između dva vrha relativno mala [110, 151].



Slika 1.3: Mreža suradnje državnih i lokalnih institucija nakon eksplozije bombe na Bostonskom maratonu u travnju 2013. godine [77]. Krug označava javnu organizaciju. Trokut označava neprofitnu organizaciju. Romb označava privatnu organizaciju. Kvadrat označava LESF (Local Emergency Support Functions)

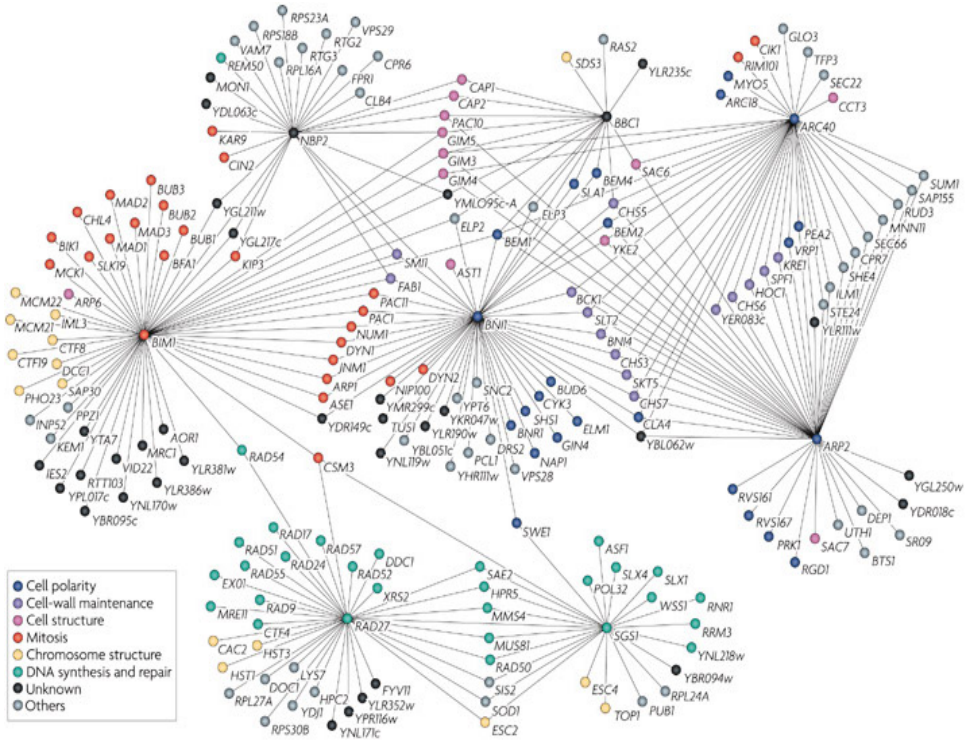
Informacijske mreže većinom su konstruirane s ciljem diseminacije znanja i pronalaženja informacija. Jedan od poznatijih primjera zasigurno je World Wide Web [1, 4]. U ovoj mreži vrhovi su mrežne stranice, a bridovi su hiperveze koje nam omogućuju navigaciju s jedne na drugu stranicu. World Wide Web se često pogrešno koristi kao sinonim za Internet, a zapravo predstavlja samo jednu od usluga pomoću koje se ostvaruje razmjena podataka putem Interneta. Mrežu su osamdesetih godina prošlog stoljeća kreirali znanstvnici u laboratoriju CERN u Ženevi za međusobnu razmjenu informacija [78], ali je vrlo brzo nadišla svoju prvotnu namjeru. U ovoj skupini često se spominju i mreža citata [94] (primjer prikazan na Slici 1.4). Većina znanstvenih radova sadrži reference na prijašnje radove pa je moguće konstruirati mrežu u kojoj su vrhovi znanstveni radovi, a brid između dva vrha A i B označava da rad A sadrži referencu na rad B. Navedene mreže primjeri su usmjerenih mreža.

Osnovni tipovi *bioloških mreža* su metaboličke mreže [69], mreže interakcija među proteinima [81], genetske regulatorne mreže [68], hranidbene mreže [54, 79] i neuralne mreže [37]. Kemijski spojevi u živim stanicama povezani su biokemijskim reakcijama koje pretvaraju jedan kemijski spoj u drugi. Svi spojevi u stanici su dijelovi zamršene biokemijske mreže reakcija koja se naziva metabolička mreža. Neuroni u mozgu su duboko povezani jedni s drugima što rezultira složenim mrežama koje su prisutne u strukturnim i funkcionalnim aspektima mozga [37].



Slika 1.4: Mreža citata generirana na temelju članaka objavljenih u PR časopisima na temu kompleksne mreže [129]. Prikazani su bridovi s težinama iznad određenog praga. Širina brida proporcionalna je težini, a veličina vrha proporcionalna je sumi težina incidentnih bridova.

Svi organizmi su povezani jedni s drugima kroz proces hranjenja. Ako se određena vrsta hrani drugom vrstom ili je plijen u odnosu na nju, međusobno su povezane u zamršenom hranidbenom lancu. Stabilnost tih interakcija jedno je od ključnih pitanja u ekologiji [101] koje je osobito važno s obzirom na potencijalni gubitak vrsta uzrokovan globalnim klimatskim promjena.



Slika 1.5: Mreža genetskih interakcija gljivica [29]. Vrhovi predstavljaju gene, a bridovi interakcije. Prikazano je 291 interakcija između 204 gena.

Mnogi sustavi koji su znanstvenicima zanimljivi sastavljeni su od pojedinačnih dijelova ili komponenti koje su na neki način povezane. Većina aspekata vezanih za takve sustave vrijedna je proučavanja. Moguće je proučavati prirodu svake zasebne komponente ili vrstu interakcija među njima, ali i uzorke koji se pojavljuju prilikom povezivanja komponenti. Mreže su stoga opširan, ali moćan alat za reprezentaciju i bolje razumijevanje takvih sustava [116].

1.2 Osnovni pojmovi teorije grafova

U ovoj disertaciji korištena je standardna terminologija teorije grafova bazirajući se na [155] i [27].

Graf je uređeni par $G = (V, E)$, gdje je $V(G) = V$ neprazan skup čije elemente nazivamo vrhovima, a $E(G) = E$ je podskup skupa neuređenih parova elemenata iz V čije elemente nazivamo bridovima. Ako za $e \in E$ vrijedi $e = \{u, v\}$, onda vrhove u i v nazivamo krajevima brida e i oni nisu nužno različiti. Graf se ponekad definira i kao uređena trojka $G = (V, E, \varphi)$, gdje je V neprazan skup čije elemente nazivamo vrhovima, E je skup čije elemente nazivamo bridovima, a φ je preslikavanje koje svakom bridu $e \in E$ pridružuje neuređeni par (ne nužno različitih) vrhova. Za $e = \{u, v\}$ kažemo da su u i v incidentni s e, te da su u i v susjedi i pišemo $e = uv$. Brid čiji se krajevi podudaraju nazivamo petljom, a dva ili više bridova s istim parom krajeva nazivamo višestrukim bridovima. Ukoliko u grafu nema ni petlji ni višestrukih bridova za graf kažemo da je jednostavan. Kompleksne mreže koje proučavamo u ovoj disertaciji su jednostavni grafovi. G je prazan graf ako je $E(G) = \emptyset$. Uvodimo oznake

$$v(G) = |V(G)| \text{ - broj vrhova u } G$$
$$e(G) = |E(G)| \text{ - broj bridova u } G$$

Jednostavan graf s n vrhova u kojem je svaki par vrhova spojen bridom nazivamo potpunim grafom i označavamo s K_n. Primjeri jednostavnih grafova koje ćemo korisiti su ciklusi i putovi. Ciklus C_n na n vrhova definiran je kao graf sa skupom vrhova $V = \{1, 2, ..., n\}$ i skupom bridova $E = \{\{i, i+1\} : i < n\} \cup \{1, n\}$. Put P_n na n vrhova definiran je skupom vrhova $V = \{1, 2, ..., n\}$ i skupom bridova $E = \{\{i, i+1\} : i < n\}$. U jednostavnom grafu put je potpuno određen nizom svojih vrhova $v_0v_1...v_k$. Kažemo da je v_0 početak, a v_k kraj puta P, a ponekad i za v_0 i za v_k kažemo da su krajevi. Za vrhove $v_1, ..., v_{k-1}$ kažemo da su unutarnji

vrhovi puta. Dva vrha $u, v \in V$ grafa G su povezana ako u G postoji uv-put. Udaljenost $d_G(u, v)$ dvaju vrhova u i v u G je duljina najkraćeg uv-puta. Kada je iz konteksta jasno o kojem grafu je riječ pišemo samo $d(u, v)$.

Neka su G i H grafovi. Kažemo da je H podgraf od G i pišemo $H \subseteq G$ ako je $V(H) \subseteq V(G)$ i $E(H) \subseteq E(G)$, a svaki brid iz H ima iste krajeve u H kao što ih ima u G. Tada kažemo i da je G nadgraf od H. Podgraf $H \subseteq G$ za koji je $V(H) = V(G)$ zovemo razapinjući podgraf.

Neka je $G = (V, E)$ i $V' \subseteq V$. Podgraf od G čiji je skup vrhova $V \backslash V'$, a skup bridova se sastoji od bridova iz E čija su oba kraja u $V \backslash V'$ označavat ćemo s $G \backslash V'$. Ako je $V' = \{v\}$ umjesto $G \backslash V'$ pisat ćemo $G \backslash \{v\}$. Podgraf od G dobiven izbacivanjem brida e iz E označavat ćemo s $G - e$, a graf dobiven dodavanjem brida $e \notin E$ grafu G s $G + e$.

Skup svih susjednih vrhova vrha $u \in V(G)$ označavamo s $N(u)$. Ako je G jednostavan graf, onda definiramo stupanj vrha u, u oznaci $d_u = d(u)$, kao broj susjeda od u. Za vrh u kažemo da je izoliran ako je $d(u) = 0$, a kažemo da je u list ako je $d(u) = 1$. Vrijedi sljedeća propozicija.

Propozicija 1.1 *Neka je $G = (V, E)$ graf. Vrijedi*

$$\sum_{u \in V(G)} d(u) = 2e(G).$$

Dijametar grafa G, u oznaci $diam(G)$, je najveća duljina najkraćeg puta za sve parove vrhova koji su međusobno povezani. Graf G je povezan ako su svaka dva njegova vrha povezana nekim putem. Povezanost među vrhovima je relacija ekvivalencije, pa stoga postoji particija skupa vrhova V na klase ekvivalencije i te klase nazivamo komponente povezanosti od G. Ako graf ima samo jednu komponentu povezanosti onda kažemo da je povezan, a inače kažemo da je nepovezan.

Graf u kojem nema ciklusa nazivamo acikličkim grafom ili šumom, a povezani aciklički graf nazivamo stablom. Za stablo G korijen stabla je bilo koji čvrsto

odabrani vrh. Vrijedi Teorem o karakterizaciji stabla:

Teorem 1.2 *Neka je $G = (V, E)$ jednostavan graf. Tada su sljedeće tvrdnje ekvivalentne.*

i) G je stablo;

ii) za svaka dva vrha $u, v \in V$ postoji jedinstveni uv-put u G;

iii) za svaki brid $e \in E$, $G - e$ je nepovezan;

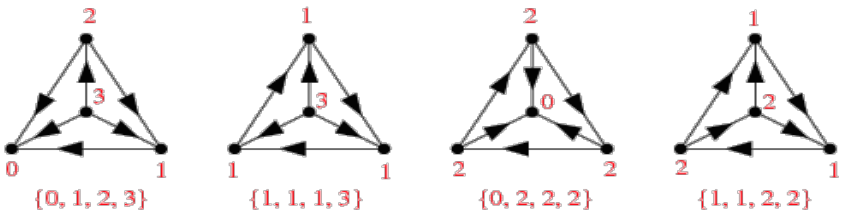
iv) dodavanjem bilo kojeg brida $e \notin E$ između postojećih vrhova $u, v \in V$, $G+e$ sadrži točno jedan ciklus;

v) vrijedi $v(G) = e(G) + 1$.

Razapinjuće stablo grafa G je razapinjući podgraf od G koji je stablo. Dijkstrino stablo s korijenom u ili stablo najkraćih putova s korijenom u grafa G je razapinjuće stablo grafa G takvo da je put od vrha u do proizvoljnog vrha v u G najkraći uv-put u G [49, 45]. Za jednostavan graf G kažemo da je vršno tranzitivan ako vrijedi $\forall u, v \in V(G)$, $\exists g \in Aut(G)$ tako da je $g(u) = v$.

U Poglavlju 3 bavimo se kurikulnim mrežama koje su jednostavni usmjereni grafovi. **Usmjereni graf** ili **digraf** D je graf G u kojem svaki brid ima smjer od početka prema kraju. D se još zove i orijentacija od G i označava $D = \overrightarrow{G}$. Brid s početkom u, a krajem v je uređeni par (u, v). Kažemo da je $e = uv$ luk ili usmjereni brid od u prema v. Dva ili više lukova s istim početkom i krajem zovu se višestruki lukovi. Digraf D možemo definirati i kao uređenu trojka (V, E, φ), gdje je $\varphi : E \to V \times V$ funkcija koja svakom luku e pridružuje uređeni par vrhova (u, v) (moguće i jednakih) koji se zovu početak i kraj od e. Jednostavni digraf je onaj koji nema višestrukih lukova i petlji. Pridruženi digraf $D(G)$ grafa G je digraf dobiven iz G zamjenom svakog brida dvama suprotno orijentiranim lukovima s istim krajevim. Pripadni graf $G(D)$ digrafa D je graf dobiven iz D brisanjem svih strelica, tj. tako da bridove u D tretiramo kao neuređene parove vrhova. Turnir je

digraf čiji je pripadni graf jednostavan i potpun. Na Slici 1.6 su prikazani primjeri turnira.



Slika 1.6: Primjer turnira na 4 vrha. Brojevi u zagradi predstavljaju niz outstupnjeva vrhova u grafu.

Mnogi pojmovi koje smo definirali za neusmjerene grafove se automatski prenose na digrafove. Primjerice, usmjerni put je jednostavni digraf čiji su vrhovi linearno poredani tako da postoji usmjereni brid s početnim vrhom u i krajnjim vrhom v ako i samo ako je vrh v odmah iza vrha u u poretku vrhova. Dijametar digrafa je duljina najkraćeg usmjerenog puta između dva najudaljenija povezana vrha. Digraf D je slabo povezan (ili kraće povezan) ako je pripadni graf povezan, a jako povezan ako za svaka dva vrha $u, v \in V$ postoji usmjereni put od u do v i usmjereni put od v do u. Komponenta digrafa D je komponenta (povezanosti) pripadnog grafa od D, a jaka komponenta digrafa D je maksimalni jako povezan poddigraf od D. Za razliku od grafova, u digrafovima razlikujemo dvije vrste stupnjeva vrha. Definiramo instupanj (ulazni stupanj) vrha v, u oznaci $d^+(v) = d_{in}(v)$, kao broj lukova s krajem v i outstupanj (izlazni stupanj) vrha v, u oznaci $d^-(v) = d_{out}(v)$, kao broj lukova s početkom v. Vrijedi sljedeća propozicija:

Propozicija 1.3 *U digrafu G vrijedi*

$$\sum_{v \in V(G)} d^+(v) = e(G) = \sum_{v \in V(G)} d^-(v).$$

Kažemo da je digraf tranzitivno zatvoren ako za svaki usmjereni put od vrha u do vrha v postoji usmjereni brid uv. Digraf je acikličan ako ne sadrži diciklus.

Poglavlje 2

Mrežni deskriptori

Važan alat za analizu mreža, posebno društvenih i komunikacijskih, su mjere centralnosti definirane na vrhovima grafa. Osmišljene su da bi se lakše odredila i interpretirala važnost i položaj vrha u mreži. Većina mjera temelji se na analizi najkraćih putova koji povezuju dva vrha. Jedna od njih je i međupoloženost, dobro poznat koncept koji nam govori kolika količina komunikacije prolazi kroz određeni vrh [21]. Vrhovi s visokim vrijednostima međupoloženosti interpretiraju se kao važni i moćni jer kontroliraju tijek informacija u mreži. Uklanjanje takvih vrhova iz mreže bi uvelike poremetilo komuunikaciju među ostalim vrhovima [32]. Bridna međupoloženost prvi put je definirana i korištena u kontekstu otkrivanja zajednica u kompleksnim mrežama [65]. Za brid uv, bridna međupoloženost (eng. *edge betweenness*) definirana je sa

$$b(uv) = \sum_{\{k,l\} \in V(G)} \frac{s_{uv}^{kl}}{s^{kl}}$$

pri čemu je s_{uv}^{kl} broj najkraćih putevi između vrhova k i l koji prolaze bridom uv, a s^{kl} je ukupan broj najkraćih puteva između k i l. Međupoloženost $c(u)$ vrha u

definirana je kao suma bridnih međupoloženosti svih bridova incidentnih s u [38]:

$$c(u) = \sum_{v \in N(u)} b(uv)$$

gdje je $N(u)$ skup svih susjeda vrha u. Još jedan važan koncept u komunikacijskim mrežama je transmisija. Za vrh $u \in V$ definirana je kao

$$t(u) = \sum_{u \in V} d(u, v)$$

pri čemu je $d(u, v)$ udaljenost između vrhova u i v. U kontekstu komunikacijskih mreža, međupoloženost možemo interpretirati kao količinu informacija koje prolaze preko vrha u, a transmisiju možemo tumačiti kao trošak vrha u za mrežu [159]. Poznato je da su obje mjere povezane s Wienerovim indeksom [164] koji je definiran kao

$$W(G) = \frac{1}{2} \sum_{(u,v) \in V^2} d(u, v).$$

Pojam je uveo Harry Wiener 1947. godine pod imenom "broj staza" (eng. *path number*) [164]. Riječ je o najstarijem topološkom indeksu povezanom s molekularnim grananjem na temelju kojeg su kasnije razvijeni brojni drugi topološki indeksi kemijskih grafova [133]. U radu [38] dokazano je da vrijedi

$$\sum_{u \in V} c(u) = \sum_{u \in V} t(u) = 2W(G) \tag{2.1}$$

Wienerov indeks usko je povezan i s mjerom centralnosti vrha (eng. *closeness centrality*) koja se često koristi u sociometriji i teoriji društvenih mreža [56].

Koristeći ove dobro poznate koncepte, definirana su dva nova mrežna deskriptora: vršna produktivnost i vršna profitabilnost. Oba deskriptora možemo interpretirati u kontekstu mjerenja produktivnosti određenog vrha [159]. Vršna produktivnost vrha u definirana je kao omjer međupoloženosti i transmisije

$$\rho(u) = \frac{c(u)}{t(u)},$$

dok je vršna profitabilnost definirana kao razlika tih dviju mjera

$$\nu(u) = c(u) - t(u).$$

Interpretacija svih definiranih mrežnih deskriptora u kontekstu komunikacijskih mreža temelji se na pretpostavci da je količina komunikacije između dva proizvoljna vrha jednaka, tj. da ne ovisi o njihovoj udaljenosti.

2.1 $d(u,v)^\lambda$ - težinski mrežni deskriptori

Realistično je pretpostaviti da će vrhovi koji se nalaze na manjim udaljenostima komunicirati više nego oni na većim. U skladu s tim, definicije mrežnih deskriptora utežene su s $d(u,v)^\lambda$ za $\lambda < 0$, generalizirajući slučaj $\lambda = -1$ predstavljen u [30]. Definiramo:

$$t_\lambda(u) = \sum_{v \in V \setminus \{u\}} d(u,v) \cdot d(u,v)^\lambda = \sum_{v \in V \setminus \{u\}} d(u,v)^{1+\lambda},$$

$$b_\lambda(uv) = \sum_{\{k,l\} \in \binom{V}{2}} \frac{s_{uv}^{kl}}{s^{kl}} \cdot d(k,l)^\lambda,$$

$$c_\lambda(u) = \sum_{v \in [u]} b_\lambda(uv).$$

Primjetimo da je funkcija t_λ rastuća funkcija u terminima udaljenosti za $\lambda > -1$, a padajuća za $\lambda < -1$. Nadalje, definiramo:

$$\rho_\lambda(u) = \frac{c_\lambda(u)}{t_\lambda(u)},$$

$$\nu_\lambda(u) = c_\lambda(u) - t_\lambda(u).$$

Za $\lambda = 0$ dobijemo standardne definicije transmisije, međupoloženosti, vršne produktivnosti i vršne profitabilnosti [38, 159]. U ovom poglavlju ograničili smo se

na slučaj kada je $\lambda < 0$ jer pozitivne vrijednosti λ vode nerealnoj pretpostavci da udaljeni vrhovi komuniciraju više nego oni koji su blizu. Analogno kao u [159] definiramo

$$mc_\lambda(G) = \min\{c_\lambda(u) : u \in V\} \quad Mc_\lambda(G) = \max\{c_\lambda(u) : u \in V\}$$
$$mt_\lambda(G) = \min\{t_\lambda(u) : u \in V\} \quad Mt_\lambda(G) = \max\{t_\lambda(u) : u \in V\}$$
$$m\rho_\lambda(G) = \min\{\rho_\lambda(u) : u \in V\} \quad M\rho_\lambda(G) = \max\{\rho_\lambda(u) : u \in V\}$$
$$m\nu_\lambda(G) = \min\{\nu_\lambda(u) : u \in V\} \quad M\nu_\lambda(G) = \max\{\nu_\lambda(u) : u \in V\}$$

Cilj je odrediti gornje i donje ograde ovih vrijednosti te graf i vrh u grafu za koji se one postižu. Dokažimo prvo tri pomoćne leme koje ćemo koristiti u daljnjim dokazima.

Lema 2.1 *Za svaki graf G i $\lambda < 0$ vrijedi*

$$\sum_{u\in V} t_\lambda(u) = \sum_{u\in V} c_\lambda(u).$$

Dokaz. Za transmisiju, tvrdnja slijedi direktno iz definicije. Dokažimo da tvrdnja vrijedi za međupoloženost. Vrijedi

$$\begin{aligned}\sum_{u\in V} c_\lambda(u) &= \sum_{u\in V}\sum_{v\in N(u)}\sum_{\{k,l\}\subseteq V} \frac{s_{uv}^{kl}}{s^{kl}} d(k,l)^\lambda \\ &= \sum_{\{k,l\}\subseteq V} \frac{d(k,l)^\lambda}{s^{kl}} \sum_{u\in V}\sum_{v\in N(u)} s_{uv}^{kl}.\end{aligned}$$

Za dani par vrhova $\{k,l\}$, $\sum_{u\in V}\sum_{v\in N(u)} s_{uv}^{kl}$ je broj parova (u,v) vrhova takvih da je $d(u,v) = 1$ i da najkraći put između k i l prolazi bridom uv. Duljina svakog od s^{kl} najkraćih putova od k do l je $d(k,l)$, pa na svakom od tih putova možemo odabrati $d(k,l)$ parova $\{u,v\}$ za koje je $d(u,v) = 1$. Stoga vrijedi

$$\sum_{u\in V}\sum_{v\in N(u)} s_{uv}^{kl} = 2 \cdot d(k,l) \cdot s^{kl}.$$

Nadalje, vrijedi

$$\sum_{u\in V} c_\lambda(u) = \sum_{\{k,l\}\subseteq V} \frac{d(k,l)^\lambda}{s^{kl}} \cdot 2d(k,l)\cdot s^{kl} = 2\sum_{\{k,l\}\subseteq V} d(k,l)^{\lambda+1}$$

i time je tvrdnja dokazana. ■

Napomena 2.2 *Obzirom na rezultat u Lemi 2.1 i jednadžbu (2.1) prirodno je označiti*

$$\sum_{u\in V} t_\lambda(u) = \sum_{u\in V} c_\lambda(u) = 2W_{\lambda+1}(G).$$

Lema 2.3 *Za pozitivne brojeve $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ vrijedi:*

$$\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} \geqslant \min\{\frac{a_i}{b_i}\}.$$

Dokaz. Vrijedi

$$\frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i} = \frac{\sum_{i=1}^{n} b_i \cdot \min\{\frac{a_i}{b_i}\} \cdot \frac{\frac{a_i}{b_i}}{\min\{\frac{a_i}{b_i}\}}}{\sum_{i=1}^{n} b_i} \geqslant \frac{\sum_{i=1}^{n} b_i \cdot \min\{\frac{a_i}{b_i}\}}{\sum_{i=1}^{n} b_i} = \min\{\frac{a_i}{b_i}\}.$$

■

Lema 2.4 *Za sve $\lambda < 0$ i za dani $n \in \mathbb{N}$, među svim grafovima s n vrhova, bilo koji graf G za koji je vrijednost $c_\lambda(G)$ maksimalna je stablo.*

Dokaz. Neka je G graf takav da je $c_\lambda(G)$ maksimalna i neka je u vrh u G za koji se postiže maksimalna mađupoloženost. Pokažimo da je G stablo. Pretpostavimo suprotno. Neka je G' Dijkstrino stablo s korijenom u vrhu u. Kako je G' stablo vrijedi da je $\frac{s_{uv}^{kl}}{s^{kl}} = 1$, za sve $k, l \in V$ koji su povezani putem koji prolazi bridom uv. Iz definicije G' jasno je da su udaljenosti između u i bilo kojeg drugog vrha iste u G i G'. No, tada je vrijednost $c_\lambda(u)$ u G' veća ili jednaka vrijednosti $c_\lambda(u)$ u G, što je protivno pretpostavci. ■

2.1.1 Vršna produktivnost

Prije nego prijeđemo na određivanje gornjih i donjih ograda za vrijednosti vršne produktivnosti, potrebna nam je sljedeća definicija.

Definicija 2.5 Metlica *$B_{n,k}$ je graf dobiven tako da identificiramo list zvijezde S_{k+1} i kraj puta P_{n-k}.*

Primjetimo da vrijedi $B_{(n-1),2} = P_n$ and $B_{1,n} = S_n$. Primjer metlice prikazan je na Slici 2.1.

Teorem 2.6 *Za svaki graf G s n vrhova i za $\lambda \in \langle -1, 0\rangle$ vrijedi*

$$m\rho_\lambda(G) \geqslant \frac{\sum_{i=1}^{n-1} i^\lambda}{\sum_{i=1}^{n-1} i^{1+\lambda}}.$$

Donja ograda se postiže za krajnji vrh puta.

Dokaz. Za $\lambda \in \langle -1, 0\rangle$ minimalna vrijednost međupoloženosti i maksimalna vrijednost transmisije postižu za krajnje vrhove puta, kao što je dokazano u [8], pa isto vrijedi i za $m\rho_\lambda(G)$. ■

Teorem 2.7 *Za svaki graf G s n vrhova i za $\lambda \in \langle -\infty, -1\rangle$ vrijedi*

$$m\rho_\lambda(G) \geqslant \min_{2 \leqslant D \leqslant n-1} \frac{D^\lambda + \frac{1}{n-D}\sum_{i=1}^{D-1} i^\lambda}{D^{1+\lambda} + \frac{1}{n-D}\sum_{i=1}^{D-1} i^{1+\lambda}}.$$

Donja ograda se postiže za početni vrh metlice.

Dokaz. Neka je G graf za koji se postiže minimalna vrijednost $m\rho_\lambda(G)$ i neka je u vrh u G takav da je $\rho_\lambda(u) = m\rho_\lambda(G)$.
Vrijedi:

$$\rho_\lambda(u) = \frac{c_\lambda(u)}{t_\lambda(u)} \geqslant \frac{\sum_{v \in V\setminus\{u\}} d(u,v)^\lambda}{\sum_{v \in V\setminus\{u\}} d(u,v)^{\lambda+1}}$$

jer vrh u sigurno leži na svakom najkraćem putu između sebe i ostalih vrhova. Neka je v_D vrh koji je najudaljeniji od u i neka je $S = uv_1v_2 \dots v_D$ najkraći put od u to v_D. Neka je $k = n - D - 1$.

Nadalje, neka je $\{w_1, \dots w_k\} = V \setminus \{u, v_1, \dots, v_D\}$ skup svih vrhova koji ne leže na putu S i neka je $W = \{w_1, w_2, \dots, w_k, v_D\}$. Budući je $d(u, v_i) = i$ za svaki $i \in \{1, 2, \dots, D\}$, vrijedi:

$$m\rho_\lambda(G) = \rho_\lambda(u) \geqslant \frac{\sum\limits_{v \in V \setminus \{u\}} d(u,v)^\lambda}{\sum\limits_{v \in V \setminus \{u\}} d(u,v)^{\lambda+1}} = \frac{\sum\limits_{i=1}^{D} i^\lambda + \sum\limits_{i=1}^{k} d(u,w_i)^\lambda}{\sum\limits_{i=1}^{D} i^{\lambda+1} + \sum\limits_{i=1}^{k} d(u,w_i)^{1+\lambda}}. \tag{2.2}$$

Posljednji izraz u (2.2) možemo zapisati na sljedeći način:

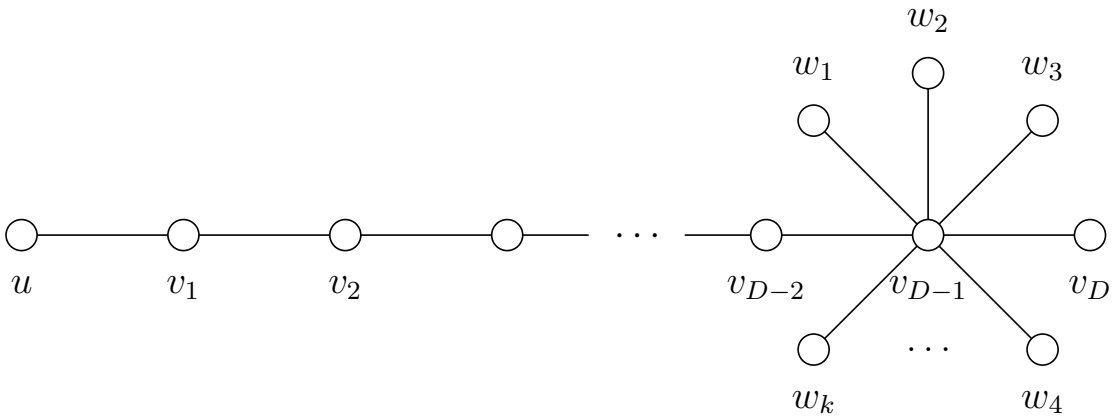
$$\frac{\sum\limits_{v \in W} \left(d(u,v)^\lambda + \frac{1}{n-D} \sum\limits_{i=1}^{D-1} i^\lambda \right)}{\sum\limits_{v \in W} \left(d(u,v)^{1+\lambda} + \frac{1}{n-D} \sum\limits_{i=1}^{D-1} i^{1+\lambda} \right)}. \tag{2.3}$$

Koristeći Lemu 2.3 zaključujemo da je minimum izraza danog u (2.3) jednak

$$\frac{x^\lambda + \frac{1}{n-D} \sum\limits_{i=1}^{D-1} i^\lambda}{x^{1+\lambda} + \frac{1}{n-D} \sum\limits_{i=1}^{D-1} i^{1+\lambda}},$$

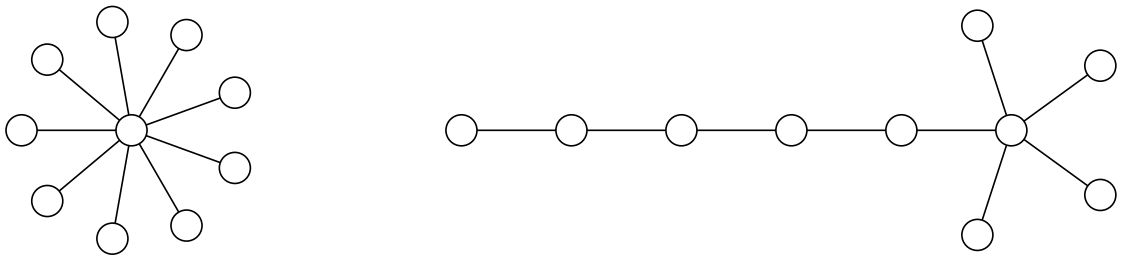
gdje je $x = d(u, v)$ za neki $v \in W$.

Dani minimum se postiže ako i samo ako je omjer $\dfrac{a_i}{b_i}$ konstantan za svaki $i \in \{1, 2, \dots, n\}$. Jedan od načina kako je to moguće postići je da je $d(u, v)$ konstantna za svaki $v \in \{w_1, \dots, w_k, v_D\}$, tj. da je $d(u, w_i) = d(u, v_D) = D$ za svaki $i \in \{1, 2, \dots, k\}$. Ovo je moguće jedino ako je w_i direktno povezan s v_{D-1} za svaki $i \leqslant k$, što dokazuje da je G zaista metlica (Slike 2.1 i 2.2). ■



Slika 2.1: Metlica za koju se postiže minimalna vrijednost $m\rho_\lambda(G)$

Napomena 2.8 *Slika 2.2 prikazuje primjere grafova koji minimiziraju vrijednost* $m\rho_\lambda(G)$ *za različite vrijednosti* λ.



$\lambda = -4$ $\lambda = -1.6$

Slika 2.2: Graf za koji se postiže minimalna vrijednost $m\rho_\lambda(G)$ prikazan za $n = 10$ i dvije različite vrijednosti λ.

Teorem 2.9 *Za svaki graf* G *s* n *vrhova i za* $\lambda < 0$ *vrijedi*

$$m\rho_\lambda(G) \leqslant 1.$$

Ograda se postiže za bilo koji vrh vršno tranzitivnog grafa.

Dokaz. Koristeći Lemu 2.1 i Lemu 2.3 zaključujemo da vrijedi:

$$\min\left\{\frac{c_\lambda(u)}{t_\lambda(u)} : u \in V\right\} \leqslant \frac{\sum\limits_{u\in V} c_\lambda(u)}{\sum\limits_{u\in V} t_\lambda(u)} = \frac{2W_{1+\lambda}(G)}{2W_{1+\lambda}(G)} = 1.$$

■

Teorem 2.10 *Za svaki graf G s n vrhova i za $\lambda < 0$ vrijedi*

$$1 \leqslant M\rho_\lambda(G) \leqslant 2^\lambda(n-2) + 1.$$

Donja ograda se postiže za bilo koji vrh vršno tranzitivnog grafa, a gornja za centralni vrh zvijezde.

Dokaz. Dokažimo prvo donju ogradu. Koristeći Lemu 2.1 zaključujemo da vrijedi:

$$\max\left\{\frac{c_\lambda(u)}{t_\lambda(u)} : u \in V\right\} \geqslant \frac{\sum\limits_{u\in V} c_\lambda(u)}{\sum\limits_{u\in V} t_\lambda(u)} = \frac{2W_{1+\lambda}(G)}{2W_{1+\lambda}(G)} = 1.$$

Za gornju ogradu, neka je $u \in V(G)$ vrh za koji se postiže maksimalna vrijednost vršne produktivnosti. Iz Leme 2.4 slijedi da je graf za koji je vrijednost $M\rho_\lambda$ maksimalna stablo. Naime, pošto je vršna produktivnost definirana kao kvocijent međupoloženosti i transmisije, želimo da brojnik ima najveću moguću vrijednost. To vrijedi kada je G stablo. Pretpostavimo da je graf koji se pojavljuje u nazivniku također stablo. Ako to nije slučaj, možemo ponoviti konstrukciju stabla G' iz Leme 2.4 da bismo dobili stablo u kojem udaljenosti između vrha u i svih ostalih vrhova ostaju iste pa samim time i vrijednost transmisije ostaje ista. Nadalje, koristeći nejednakost (2.4) koja vrijedi za $x \leqslant y$

$$\left(\frac{\sum\limits_{i=1}^{n} {a_i}^x}{n}\right)^{\frac{1}{x}} \leqslant \left(\frac{\sum\limits_{i=1}^{n} {a_i}^y}{n}\right)^{\frac{1}{y}} \tag{2.4}$$

za potencije 1 i λ dobije se:

$$\begin{aligned}
\rho_\lambda(u) &= \frac{\sum\limits_{q\in[u]} b_\lambda(uq)}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda}} \\
&\leqslant \frac{2\sum\limits_{\{v,w\}\in\binom{V\setminus\{u\}}{2}} [d(v,u)+d(u,w)]^\lambda + \sum\limits_{v\in V\setminus\{u\}} d(u,v)^\lambda}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda}} \\
&\leqslant \frac{2\sum\limits_{\{v,w\}\in\binom{V\setminus\{u\}}{2}} \left[d(v,u)^\lambda + d(u,w)^\lambda\right]\cdot\frac{2^\lambda}{2} + \sum\limits_{v\in V\setminus\{u\}} d(u,v)^\lambda}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda}} \\
&\leqslant \frac{2^\lambda(n-2)\sum\limits_{v\in V\setminus\{u\}} d(u,v)^\lambda + \sum\limits_{v\in V\setminus\{u\}} d(u,v)^\lambda}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda}} \\
&\leqslant \frac{\left[2^\lambda(n-2)+1\right]\sum\limits_{v\in V\setminus\{u\}} d(u,v)^\lambda}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda}} \leqslant 2^\lambda(n-2)+1.
\end{aligned}$$

Jednostavni izračun pokazuje da se jednakost postiže za centralni vrh zvijezde. ■

2.1.2 Vršna profitabilnost

Teorem 2.11 *Za svaki graf G s n vrhova i za $\lambda < 0$ vrijedi:*

$$m\nu_\lambda(G) \leqslant 0.$$

Ograda se postiže za bilo koji vrh u vršno tranzitivnom grafu.

Dokaz. Koristeći Lemu 2.1, usporedimo li minimalnu i prosječnu vrijednost $\nu_\lambda(u)$, vrijedi:

$$\begin{aligned}
m\nu_\lambda(u) &= \min\left\{c_\lambda(u) - t_\lambda(u) : u\in V\right\} \leqslant \frac{1}{n}\left(\sum_{u\in V}\left(c_\lambda(u)-t_\lambda(u)\right)\right) \\
&= \frac{1}{n}\left(\sum_{u\in V} c_\lambda(u) - \sum_{u\in V} t_\lambda(u)\right) = \frac{1}{n}\left(2W_{1+\lambda}(G) - 2W_{1+\lambda}(G)\right) = 0.
\end{aligned}$$

Jednakost očito vrijedi za vršno tranzitivni graf. ■

Teorem 2.12 *Za svaki graf G s n vrhova i za $\lambda \in \langle -\infty, -1\rangle$ vrijedi:*

$$m\nu_\lambda(G) \geqslant -(n-2)\cdot 2^\lambda.$$

Ograda se postiže za list zvijezde.

Dokaz. Neka je $u \in V(G)$ vrh u grafu G takav da je $m\nu_\lambda(G) = \nu_\lambda(u)$. Vrijedi:

$$\begin{aligned}\nu_\lambda(u) = c_\lambda(u) - t_\lambda(u) &\geqslant \sum_{v\in V\setminus\{u\}} d(u,v)^\lambda - \sum_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda} \\ &= \sum_{v\in V\setminus\{u\}} d(u,v)^\lambda(1-d(u,v)) \geqslant -(n-2)\cdot 2^\lambda,\end{aligned}$$

jer je funkcija $f(x) = x^l(1-x)$ rastuća na segmentu $\langle 2, +\infty\rangle$ i $(n-2)$ je maksimalni broj vrhova $v \in V$ takvih da je $d(u,v) = 2$. ■

Teorem 2.13 *Za svaki graf G s n vrhova i za $\lambda \in \langle -1, 0\rangle$ vrijedi:*

$$m\nu_\lambda(G) \geqslant \sum_{i=1}^{n-1}(i^\lambda - i^{1+\lambda}).$$

Ograda se postiže za krajnji vrh puta.

Dokaz. Za $\lambda \in \langle -1, 0\rangle$, minimalna vrijednost međupoloženosti i maksimalna vrijednost transmisije postižu se za krajnji vrh puta, kao što je dokazano u [8], pa isto vrijedi i za $m\nu_\lambda(G)$. ■

Teorem 2.14 *Za svaki graf G s n vrhova i za $\lambda < 0$ vrijedi*

$$0 \leqslant M\nu_\lambda(G) \leqslant (n-1)(n-2)\cdot 2^\lambda.$$

Donja ograda se postiže za bilo koji vrh u vršno tranzitivnom grafu, a gornja za centralni vrh zvijezde.

Dokaz. Za donju ogradu, koristeći Lemu 2.1 i usporedbu maksimalne i prosječne vrijednosti $\nu_\lambda(u)$, dobijemo

$$\begin{aligned} M\nu_\lambda(u) &= \max\left\{c_\lambda(u) - t_\lambda(u) : u \in V\right\} \geqslant \frac{1}{n}\left(\sum_{u\in V}\left(c_\lambda(u) - t_\lambda(u)\right)\right) \\ &= \frac{1}{n}\left(\sum_{u\in V} c_\lambda(u) - \sum_{u\in V} t_\lambda(u)\right) = \frac{1}{n}\left(2W_{1+\lambda}(G) - 2W_{1+\lambda}(G)\right) = 0. \end{aligned}$$

Dokažimo gornju ogradu. Analogno kao u dokazu Teorema 2.10, iz Leme 2.4 slijedi da je graf za koji se postiže gornja ograda stablo. Neka je $u \in V$ vrh u grafu G za koji se postiže maksimalna vrijednost ν_λ. Vrijedi:

$$\begin{aligned} \nu_\lambda(u) &= c_\lambda(u) - t_\lambda(u) \\ &\leqslant \sum_{v\in V\setminus\{u\}} d(u,v)^\lambda + 2\sum_{v,w\in V\setminus\{u\}} d(v,w)^\lambda - \sum_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda} \\ &\leqslant \sum_{v\in V\setminus\{u\}} d(u,v)^\lambda + 2\sum_{v,w\in V\setminus\{u\}} 2^\lambda - \sum_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda} \\ &= \sum_{v\in V\setminus\{u\}} d(u,v)^\lambda + (n-1)(n-2)\cdot 2^\lambda - \sum_{v\in V\setminus\{u\}} d(u,v)^{1+\lambda} \\ &\leqslant (n-1)(n-2)\cdot 2^\lambda. \end{aligned}$$

Jednostavni izračun pokazuje da jednakost vrijedi za centralni vrh zvijezde. ■

2.2 $\lambda^{d(u,v)}$ - težinski mrežni deskriptori

Promotrimo sada mrežne deskriptore transmisiju, međupoloženost, vršnu produktivnost i vršnu profitabilnost uz pretpostavku da je količina komunikacije između dva vrha proporcionalna $\lambda^{d(u,v)}$ za $\lambda \in \langle 0, 1\rangle$ [9]. Definiramo:

$$t^e_\lambda(u) = \sum_{v \in V \setminus \{u\}} d(u,v) \cdot \lambda^{d(u,v)},$$

$$b^e_\lambda(uv) = \sum_{\{k,l\} \in \binom{V}{2}} \frac{s^{kl}_{uv}}{s^{kl}} \cdot \lambda^{d(u,v)},$$

$$c^e_\lambda(u) = \sum_{v \in N(u)} b^e_\lambda(uv)$$

Nadalje,

$$\rho^e_\lambda(u) = \frac{c^e_\lambda(u)}{t^e_\lambda(u)},$$

$$\nu^e_\lambda(u) = c^e_\lambda(u) - t^e_\lambda(u).$$

Definiramo:

$$mc^e_\lambda(G) = \min\{c^e_\lambda(u) : u \in V\} \quad Mc^e_\lambda(G) = \max\{c^e_\lambda(u) : u \in V\}$$

$$mt^e_\lambda(G) = \min\{t^e_\lambda(u) : u \in V\} \quad Mt^e_\lambda(G) = \max\{t^e_\lambda(u) : u \in V\}$$

$$m\rho^e_\lambda(G) = \min\{\rho^e_\lambda(u) : u \in V\} \quad M\rho^e_\lambda(G) = \max\{\rho^e_\lambda(u) : u \in V\}$$

$$m\nu^e_\lambda(G) = \min\{\nu^e_\lambda(u) : u \in V\} \quad M\nu^e_\lambda(G) = \max\{\nu^e_\lambda(u) : u \in V\}$$

Cilj je pronaći gornje i donje ograde ovih vrijednosti za sve $\lambda \in \langle 0, 1\rangle$.

2.2.1 Veza između t^e_λ i c^e_λ

Analogno kao u [8], možemo interpretirati $t^e_\lambda(u)$ kao trošak koji vrh u napravi mreži, a $c^e_\lambda(u)$ kao količinu komunikacije koja prolazi vrhom u. Dokažimo da su

sume ovih dviju vrijednosti jednake.

Teorem 2.15 *Za svaki graf G vrijedi*

$$\sum_{u\in V} t^e_\lambda(u) = \sum_{k,l\in V} d(k,l)\cdot\lambda^{d(k,l)} = \sum_{u\in V} c^e_\lambda(u).$$

Dokaz. Prva jednakost slijedi iz definicije transmisije. Nadalje, vrijedi

$$\sum_{u\in V} c^e_\lambda(u) = \sum_{u\in V}\sum_{v\in N(u)}\sum_{k,l\in V}\frac{s^{kl}_{uv}}{s^{kl}}\lambda^{d(k,l)} = \sum_{k,l\in V}\frac{\lambda^{d(k,l)}}{s^{kl}}\sum_{u\in V}\sum_{v\in N(u)} s^{kl}_{uv}.$$

Za dani par vrhova $(k,l)\in V^2$, $\sum\limits_{u\in V}\sum\limits_{v\in N(u)} s^{kl}_{uv}$ je broj parova (u,v) takvih da je $d(u,v)=1$ i najkraći put između k i l prolazi bridom uv. Duljina svakog od s^{kl} najkraćih putova od k do l je $d(k,l)$ i zato na svakom takvom putu možemo odabrati $d(k,l)$ parova $\{u,v\}$ takvih da je $d(u,v)=1$. Dakle,

$$\sum_{u\in V}\sum_{v\in[u]} s^{kl}_{uv} = d(k,l)\cdot s^{kl}.$$

Konačno, imamo

$$\sum_{u\in V} c^e_\lambda(u) = \sum_{k,l\in V}\frac{\lambda^{d(k,l)}}{s^{kl}}\cdot d(k,l)\cdot s^{kl} = \sum_{k,l\in V} d(k,l)\cdot\lambda^{d(k,l)}.$$

■

2.2.2 Transmisija

Za minimalnu transmisiju dokažimo

Teorem 2.16 *Za svaki graf G s n vrhova vrijedi*

$$\min_{1\leqslant D\leqslant n-1}\frac{\lambda\left[1-(D+1)\lambda^D+D\lambda^{D+1}\right]}{(\lambda-1)^2}+(n-D-1)D\cdot\lambda^D \leqslant mt^e_\lambda(G) \qquad (2.5)$$

Donja ograda postiže se za početni vrh metlice.

Dokaz. Neka je G graf za koji se postiže minimalna vrijednost $mt^e_\lambda(G)$ i neka je $u \in V(G)$ vrh u grafu za koji je $t^e_\lambda(u) = mt^e_\lambda(G)$. Neka je v_D vrh koji je najdalji od u, tj. udaljenost između u i v_D je najveća u grafu, i neka je $S = uv_1v_2...v_D$ najkraći put od u do v. Nadalje, neka je $k = n - D - 1$ i neka je $W = \{w_1, w_2, ..., w_k\}$ skup vrhova koji ne leže na putu S. Pošto je $d(u, v_i) = i$ za svaki $i \in \{1, 2, ..., D\}$ vrijedi:

$$mt^e_\lambda(G) = t^e_\lambda(u) = \sum_{i=1}^{D} i \cdot \lambda^i + \sum_{w \in W} d(u, w) \cdot \lambda^{d(u,w)}.$$

Budući za pozitivne brojeve $a_1, a_2, ..., a_n$ takve da je $a = min\{a_1, a_2, ..., a_n\}$ vrijedi

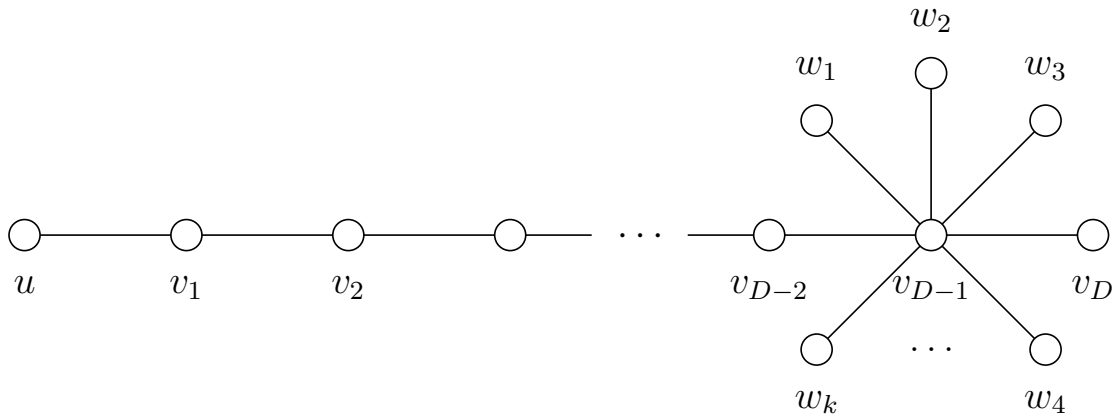
$$\sum_{i=1}^{n} a_i \geqslant \sum_{i=1}^{n} a = n \cdot a,$$

zaključujemo da je

$$mt^e_\lambda(G) = \sum_{i=1}^{D} i \cdot \lambda^i + \sum_{w \in W} d(u, w) \cdot \lambda^{d(u,w)} \geqslant \sum_{i=1}^{D} i \cdot \lambda^i + (n - D - 1)x \cdot \lambda^x$$

gdje je $x = d(u, q)$ za neki $q \in W$ za koji izraz $d(u, q) \cdot \lambda^{d(u,q)}$ ima najmanju vrijednost. To znači da će transmisija biti minimalna ako su svi vrhovi u W jednako udaljeni od u. Dokazat ćemo da je u tom slučaju $x = D$. Pretpostavimo suprotno. Promotrimo graf G' koji se dobije tako da uklonimo vrh v_D i spojimo ga s v_{x-1}. Transmisija u G' je manja nego u G, što je kontradikcija, dakle, $x = D$. Zaključujemo da je jedan od grafova za koje se postiže minimalna vrijednost transmisije metlica, odnosno da su svi vrhovi iz W direktno povezani s v_{D-1}. ■

Napomena 2.17 *Na Slici 2.3 prikazan je primjer metlice za koju se postiže minimalna vrijednost* $mt^e_\lambda(G)$.



Slika 2.3: Metlica za koju se postiže minimalna vrijednost $mt^e_\lambda(G)$.

Uz određene pretpostavke, možemo svesti slučaj $D \in \{1, .., n-1\}$ iz prethodnog Teorema na slučaj $D \in \{1, n-1, \lfloor D_{min} \rfloor, \lceil D_{min} \rceil\}$. Dokažimo:

Teorem 2.18 *Funkcija* $f(D) = \sum_{i=1}^{D} i \cdot \lambda^i + (n-D-1)D \cdot \lambda^D$ *ima lokalni minimum u točki*

$$D_1 = \frac{2 - 2\lambda + [1 + (\lambda - 1)\, n]\, Log\lambda + \sqrt{S_\lambda}}{2\,(\lambda - 1)\, Log\lambda} \tag{2.6}$$

i lokalni maksimum u točki

$$D_2 = \frac{2 - 2\lambda + [1 + (\lambda - 1)\, n]\, Log\lambda - \sqrt{S_\lambda}}{2\,(\lambda - 1)\, Log\lambda}, \tag{2.7}$$

pri čemu je

$$S_\lambda = 4\,(\lambda - 1)^2 + \left[(n-1)^2 + \lambda^2 n^2 - 2\lambda\left(n^2 - n + 2\right)\right] Log\lambda^2,$$

ako su $D_1, D_2 \in \mathbb{R}$.

Dokaz. Problem se svodi na traženje minimuma (maksimuma) funkcije

$$f(D) = \sum_{i=1}^{D} i \cdot \lambda^i + (n - D - 1)D \cdot \lambda^D.$$

Deriviranje funkcije i pojednostavljivanje izraza daje nam

$$f'(D) = \frac{\lambda^D}{(\lambda-1)^2}\left(AD^2 + BD + C\right);$$

gdje je:

$A = 2\lambda Log\lambda - Log\lambda - \lambda^2 Log\lambda;$

$B = 4\lambda - 2 - 2\lambda^2 + Log\lambda\,(\lambda - 1 + n - 2\lambda n + \lambda^2 n);$

$C = \lambda + n - 1 - 2\lambda n + \lambda^2 n - \lambda Log\lambda.$

Stacionarne točke su

$$D_1 = \frac{2 - 2\lambda + [1 + (\lambda-1)\,n]\,Log\lambda + \sqrt{S_\lambda}}{2\,(\lambda-1)\,Log\lambda}$$

i

$$D_2 = \frac{2 - 2\lambda + [1 + (\lambda-1)\,n]\,Log\lambda - \sqrt{S_\lambda}}{2\,(\lambda-1)\,Log\lambda},$$

gdje je

$$S_\lambda = 4\,(\lambda-1)^2 + \left[(n-1)^2 + \lambda^2 n^2 - 2\lambda\left(n^2 - n + 2\right)\right] Log\lambda^2.$$

Analizirajmo $f'(D)$. Pošto je $\frac{\lambda^D}{(\lambda-1)^2}$ uvijek pozitivno, rast ili pad funkcije $f(D)$ ovisi o polinomu drugog stupnja $AD^2 + BD + C$. Vodeći koeficijent $A = 2\lambda Log\lambda - Log\lambda - \lambda^2 Log\lambda > 0$ za $\lambda \in \langle 0, 1\rangle$. Zaključujemo da, pod pretpostavkom da su $D_1, D_2 \in \mathbb{R}$, funkcija $f(D)$ poprima minimalnu vrijednost za D_1 i maksimalnu vrijednost za D_2. ■

Napomena 2.19 *Označimo $D_{min} = D_1$. Ako je $D_{min} \in [1, n-1]$ i realan broj, minimum izraza $\sum_{i=1}^{D} i \cdot \lambda^i + (n - D - 1)D \cdot \lambda^D$ postiže se za neki $D \in \{1, n-1, \lfloor D_{min}\rfloor, \lceil D_{min}\rceil\}$. U suprotnom, postiže se za $D \in \{1, n-1\}$. Ova primjedba pomaže pojednostavniti izračun donje ograde iz Teorema 2.16.*

Promotrimo sada gornju ogradu. Odredili smo je za slučaj kada je $\lambda \in \langle 0, \frac{1}{2}\rangle$.

Teorem 2.20 *Za svaki graf G s n vrhova i za $\lambda \in \langle 0, \frac{1}{2}\rangle$ vrijedi*

$$mt^e_\lambda(G) \leqslant (n-1)\cdot\lambda.$$

Gornja ograda postiže se bilo koji vrh u potpunom grafu.

Dokaz. Neka je G graf za koji se postiže najveća vrijednost $mt^e_\lambda(G)$ i neka je $u \in V(G)$ vrh u grafu za koji je $t^e_\lambda(u) = mt^e_\lambda(G)$. Vrijedi:

$$mt^e_\lambda(G) = t^e_\lambda(u) = \sum_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)} \leqslant \sum_{v\in V\setminus\{u\}} \lambda = (n-1)\cdot\lambda.$$

Nejednakost vrijedi jer je za $\lambda \in \langle 0, \frac{1}{2}\rangle$ funkcija $f(x) = x\lambda^x$ padajuća. Jednakost vrijedi za potpuni graf pošto je $d(u,v) = 1$ za svaki $u, v \in V$. ■

Analizirajmo donju ogradu za $Mt^e_\lambda(G)$. Odredili smo je u posebnom slučaju za 2–povezane grafove i za $\lambda \in \langle 0, \frac{1}{2}\rangle$. Nadalje postavljamo slutnju.

Slutnja 2.21 *Za svaki graf G s $n \geqslant 3$ vrhova i za $\lambda \in \langle 0, \frac{1}{2}\rangle$ vrijedi*

$$\left\{ \begin{array}{l} \frac{\sqrt{\lambda}\left[2\sqrt{\lambda}+(n-1)\lambda^{1+\frac{n}{2}}-(n+1)\lambda^{\frac{n}{2}}\right]}{(\lambda-1)^2},\ n \text{ neparan} \\ \\ \frac{1}{2}n\lambda^{\frac{n}{2}} + \frac{\sqrt{\lambda}\left[2\sqrt{\lambda}+(n-1)\lambda^{1+\frac{n}{2}}-(n+1)\lambda^{\frac{n}{2}}\right]}{(\lambda-1)^2},\ n \text{ paran} \end{array} \right\} \leqslant Mt^e_\lambda(G) \qquad (2.8)$$

Jednakost vrijedi za bilo koji vrh u ciklusu.

Napomena 2.22 *Prethodna slutnja vrijedi u posebnom slučaju kada je G 2–povezan. Da bismo to dokazali potrebna nam je sljedeća lema.*

Lema 2.23 *Neka je $n \geqslant 3$. Neka je $\lambda \in \langle 0, \frac{1}{2}\rangle$ i neka je S skup nizova $\left(x_1, x_2, ..., x_{\lfloor n/2\rfloor}\right) \in \mathbb{N}^{\lfloor n/2\rfloor}$ takvih da je $x_1 + x_2 + ... + x_{\lfloor n/2\rfloor} = n-1$ i postoji $k \in \{1, ..., \lfloor n/2\rfloor\}$ takav*

da je $x_i \geq 2$ *za svaki* $i \leq k$ *i* $x_i = 0$ *za svaki* $i > k$.
Neka je S' *skup nizova u* S *oblika* $\left(x_1, x_2, ..., x_{\lfloor n/2 \rfloor}\right)$ *takvih da postoji* $k \in \{1, ..., \lfloor n/2 \rfloor\}$ *takav da je* $x_k \in \{0, 1\}$, $x_i = 2$ *za svaki* $1 \leq i < k$ *and* $x_i = 0$ *za svaki* $i > k$. *Neka je* T_n *definiran kao*

$$T_n\left(x_1, x_2, ..., x_{\lfloor n/2 \rfloor}\right) = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i \cdot i \cdot \lambda^i.$$

Tada je

$$\min\left\{T_n(s) : s \in S\right\} = \min\left\{T_n(s) : s \in S'\right\}.$$

Nadalje, minimalna vrijednost T_n *u* S' *je:*

$$\left\{ \begin{array}{c} 2 \cdot \sum_{i=1}^{\frac{n-1}{2}} i \cdot \lambda^i, \; n \text{ neparan} \\ 2 \cdot \sum_{i=1}^{\frac{n-2}{2}} i \cdot \lambda^i + \left(\frac{n}{2}\right) \cdot \lambda^{\frac{n}{2}}, \; n \text{ paran} \end{array} \right\}$$

Dokaz. Pretpostavimo suprotno. Neka $\left(x_1, x_2, ..., x_{\lfloor n/2 \rfloor}\right) \notin S'$ minimizira T_n u S. Tada postoji k takav da je $x_k > 2$. Primjetimo da je $k < \lfloor n/2 \rfloor$. Tada je

$$\left(x_1, x_2, ..., x_k - 1, x_{k+1} + 1, x_{k+2}, ..., x_{\lfloor n/2 \rfloor}\right) \in S.$$

Slijedi

$$\begin{aligned} 0 &\geqslant T_n\left(x_1, x_2, ..., x_{\lfloor n/2 \rfloor}\right) - T_n\left(x_1, x_2, ..., x_k - 1, x_{k+1} + 1, x_{k+2}, ..., x_{\lfloor n/2 \rfloor}\right) \\ &= k\lambda^k - (k+1)\lambda^{k+1} > 0, \end{aligned}$$

što je kontradikcija. Dakle, niz $s \in S'$ koji minimizira T_n je $(2, 2, ..., 2, 0)$ za neparan n i $(2, 2, ...2, 1)$ za parni n. Lako se vidi da je vrijednost T_n za te nizove jednaka $2 \cdot \sum_{i=1}^{\frac{n-1}{2}} i \cdot \lambda^i = \frac{\sqrt{\lambda}\left[2\sqrt{\lambda} + (n-1)\lambda^{1+\frac{n}{2}} - (n+1)\lambda^{\frac{n}{2}}\right]}{(\lambda-1)^2}$ u prvom slučaju, i $2 \cdot \sum_{i=1}^{\frac{n-2}{2}} i \cdot \lambda^i + \left(\frac{n}{2}\right) \cdot \lambda^{\frac{n}{2}} = \frac{1}{2} n \lambda^{\frac{n}{2}} + \frac{\sqrt{\lambda}\left[2\sqrt{\lambda} + (n-1)\lambda^{1+\frac{n}{2}} - (n+1)\lambda^{\frac{n}{2}}\right]}{(\lambda-1)^2}$ u drugom slučaju. ■

Dokaz Napomene 2.22. Označimo lijevu stranu izraza (2.8) sa $\mathrm{cyc}_\lambda(n)$ i pretpostavimo suprotno: da postoji 2–povezan graf G s n vrhova takav da je $Mt^e_\lambda(G) < \mathrm{cyc}_\lambda(n)$. To implicira da je $t^e_\lambda(u) < \mathrm{cyc}_\lambda(n)$, za svaki $u \in V$. Neka je $w \in V$ takav da je $t^e_\lambda(w) < \mathrm{cyc}_\lambda(n)$. Neka je w_1 vrh koji je najviše udaljen od w i neka je $d(w, w_1) = D$. Pošto je G 2–povezan vrijedi da za svaki $d < D$ postoje barem 2 vrha na udaljenosti d od w. Iz ovoga se lako vidi da je $D \leq \left\lfloor \frac{n}{2} \right\rfloor$. Označimo s x_i broj vrhova na udaljenosti i od w i promotrimo niz $(x_1, ..., x_{\lfloor n/2 \rfloor})$. Ovaj niz je očito u skupu S definiranom u Lemi 2.23. Slijedi da je $\mathrm{cyc}_\lambda(n) \leqslant t^e_\lambda(w)$ što je kontradikcija. ■

Teorem 2.24 *Za svaki graf G s n vrhova vrijedi*

$$Mt^e_\lambda(G) \leqslant \max_{1 \leqslant D \leqslant n-1} \frac{\lambda\left[1 - (D+1)\lambda^D + D\lambda^{D+1}\right]}{(\lambda - 1)^2} + (n - D - 1)D \cdot \lambda^D$$

Jednakost vrijedi za početni vrh metlice.

Dokaz. Neka je G graf za koji se postiže maksimalna vrijednost $Mt^e_\lambda(G)$ i neka je $u \in V(G)$ vrh u grafu za koji je $t^e_\lambda(u) = Mt^e_\lambda(G)$. Neka je v_D vrh koji je najudaljeniji od u i neka je $S = uv_1v_2...v_D$ najkraći put od u do v_D. Nadalje, neka je $k = n - D - 1$ i neka je $W = \{w_1, w_2, ..., w_k\}$ skup vrhova koji ne leže na putu S. Pošto je $d(u, v_i) = i$ za svaki $i \in \{1, 2, ..., D\}$ imamo da je:

$$Mt^e_\lambda(G) = t^e_\lambda(u) = \sum_{i=1}^{D} i \cdot \lambda^i + \sum_{w \in W} d(u, w) \cdot \lambda^{d(u,w)}.$$

Za pozitivne brojeve $a_1, a_2, ..., a_n$ takve da je $a = max\{a_1, a_2, ..., a_n\}$ vrijedi

$$\sum_{i=1}^{n} a_i \leqslant \sum_{i=1}^{n} a = n \cdot a,$$

pa zaključujemo da je

$$Mt^e_\lambda(G) = \sum_{i=1}^{D} i \cdot \lambda^i + \sum_{w \in W} d(u, w) \cdot \lambda^{d(u,w)} \leqslant \sum_{i=1}^{D} i \cdot \lambda^i + (n - D - 1)x \cdot \lambda^x$$

gdje je $x = d(u, q)$za neki $q \in W$ za koji izraz $d(u, q) \cdot \lambda^{d(u,q)}$ ima najveću vrijednost. To znači da će transmisija imati maksimalnu vrijednost ako su svi vrhovi iz W jednako udaljeni od u. Dokažimo da je u tom slučaju $x = D$. Pretpostavimo suprotno. Promotrimo graf G' dobiven tako da uklonimo vrh v_D i spojimo ga na v_{x-1}. Graf G' ima veću transmisiju nego G što je kontradikcija. Zaključujemo da je jedan od grafova za koje je transmisija maksimalna metlica, to jest, svi vrhovi iz W moraju biti direktno povezani s v_{D-1}. ■

Napomena 2.25 *Neka je $D_{max} = D_2$ iz Teorema 2.18. Ako vrijedi da je $D_{max} \in [1, n-1] \in \mathbb{R}$, tada se maksimum izraza $\sum_{i=1}^{D} i \cdot \lambda^i + (n - D - 1)D \cdot \lambda^D$ postiže za neki $D \in \{1, n-1, \lfloor D_{max} \rfloor, \lceil D_{max} \rceil\}$. U suprotnom, postiže se za $D \in \{1, n-1\}$. Ova primjedba može pojednostavniti izračun gornje ograde u Teoremu 2.24.*

2.2.3 Međupoloženost

Dokažimo prvo leme koje ćemo koristiti u daljnjim dokazima.

Lema 2.26 *Za svaki $\lambda \in \langle 0, 1\rangle$ i za dani broj $n \in \mathbb{N}$, među svim grafovima s n vrhova, graf G za koji se postiže maksimalna vrijednost $c_\lambda^e(G)$ je stablo.*

Dokaz. Neka je G graf takav da je vrijednost $c_\lambda^e(G)$ maksimalna i neka je $u \in V(G)$ vrh u grafu za koji se ta vrijednost postiže. Pokazat ćemo da je G stablo. Pretpostavimo suprotno. Promotrimo Dijkstrino razapinjuće stablo G' dobiveno na slijedeći način. Počevši od vrha u, u svakom koraku odaberemo vrh v koji je najbliži vrhu u (udaljenost izmedju vrhova u i v je najmanja) i još nije dio stabla. Pošto je G' stablo, vrijedi $\frac{s_{uv}^{kl}}{s^{kl}} = 1$ za svaki $k, l \in V$ koji su povezani putem koji prolazi bridom uv. Po načinu na koji je konstruirano stablo G', očito je da udaljenosti između u i v, za svaki $v \in V$, ostaju iste. To znači da je $c_\lambda^e(u)$ veći u G' nego u G što je kontradikcija s pretpostavkom. ■

Lema 2.27 *Za svaki graf G s n vrhova i za $\lambda \in \langle 0,1\rangle$ vrijedi*

$$\sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} \geqslant \sum_{i=1}^{n-1} \lambda^i = \frac{\lambda^D - \lambda}{\lambda - 1}.$$

Dokaz. Neka je G graf s n vrhova i neka su u i v vrhovi povezani najdužim putem u grafu, tj. $d(u,v) = diam(G)$. Neka je W skup svih vrhova koji ne leže na putu od u do v. Promotrimo graf G' koji se dobije tako da otkinemo bilo koji vrh $w \in W$ i zalijepimo ga na v. Ovim postupkom povećali smo udaljenosti između vrhova, i samim time, pošto je $\lambda \in \langle 0,1\rangle$, smanjili smo vrijednost sume. Nastavimo li ponavljati ovaj proces dolazimo do zaključka da će suma biti minimalna ako je G put, tj. ako vrijedi

$$\sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} \geqslant \sum_{i=1}^{n-1} \lambda^i = \frac{\lambda^D - \lambda}{\lambda - 1}.$$

■

Dokažimo sada sljedeće teoreme.

Teorem 2.28 *Za svaki graf G s n vrhova i $\lambda \in \langle 0,1\rangle$ vrijedi*

$$\frac{\lambda^D - \lambda}{\lambda - 1} \leqslant mc^e_\lambda(G).$$

Donja ograda postiže se za krajnji vrh puta.

Dokaz. Neka je G graf za koji je vrijednost $mc^e_\lambda(G)$ minimalna i neka je $u \in V(G)$ vrh takav da je $c^e_\lambda(u) = mc^e_\lambda(G)$. Koristeći Lemu 2.27 dobijemo

$$mc^e_\lambda(G) \geqslant \sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} \geqslant \sum_{i=1}^{n-1} \lambda^i = \frac{\lambda^D - \lambda}{\lambda - 1}.$$

■

Što se tiče gornje ograde, riješili smo problem za $\lambda \in \langle 0, \frac{1}{2}\rangle$.

Teorem 2.29 *Za svaki graf G s n vrhova i za $\lambda \in \langle 0, \frac{1}{2}\rangle$ vrijedi*

$$mc_\lambda^e(G) \leqslant (n-1) \cdot \lambda$$

Gornja ograda se postiže za bilo koji vrh u potpunom grafu.

Dokaz. Koristeći Teorem 2.15, možemo ograničiti prosječnu međupoloženost na sljedeći način:

$$\frac{1}{n}\sum_{u\in V} c_\lambda^e(u) = \frac{1}{n}\sum_{k,l\in V} d(k,l)\cdot\lambda^{d(k,l)} \leqslant \frac{1}{n}\sum_{k,l\in V}\lambda = \frac{1}{n}\cdot n(n-1)\cdot\lambda = (n-1)\cdot\lambda.$$

Pošto je minimalna vrijednost međupoloženosti manja ili jednaka prosječnoj, tvrdnja je dokazana. Jednakost vrijedi za potpuni graf pošto je $d(k,l) = 1$ za bilo koja dva vrha $k, l \in V$. ■

Teorem 2.30 *Za svaki graf G s n vrhova i $\lambda \in \langle 0, 1\rangle$ vrijedi*

$$Mc_\lambda^e(G) \leqslant (n-1)[\lambda + \frac{1}{2}(n-2)\cdot\lambda^2].$$

Jednakost vrijedi za centralni vrh zvijezde.

Dokaz. Koristeći Lemu 2.26 možemo zaključiti da je traženi graf stablo. Neka je G stablo takvo da je vrijednost $c_\lambda^e(G)$ maksimalna i neka je $u \in V(G)$ vrh u grafu za koji se ta vrijednost postiže. Neka je P skup svih neuređenih parova vrhova $v, w \in V \setminus \{u\}$ takvih da najkraći put od v do w prolazi vrhom u. Vrijedi:

$$\begin{aligned} Mc_\lambda^e(G) &= c_\lambda^e(u) = \sum_{v\in[u]}\sum_{k,l\in V}\frac{s_{uv}^{kl}}{s^{kl}}\lambda^{d(k,l)} = \sum_{v\in V\setminus\{u\}}\lambda^{d(u,v)} + \sum_{\{v,w\}\in P}\lambda^{d(v,w)} \\ &\leqslant (n-1)\cdot\lambda + \frac{1}{2}(n-1)(n-2)\cdot\lambda^2 \\ &= (n-1)\left[\lambda + \frac{1}{2}(n-2)\cdot\lambda^2\right]. \end{aligned}$$

Maksimalna vrijednost međupoložnosti postiže se za centralni vrh zvijezde pošto su svi vrhovi $v \in V \setminus \{u\}$ direktno povezani s u i vrijedi $d(v,w) = 2$ za sve vrhove $v, w \in V \setminus \{u\}$. ■

2.2.4 Vršna produktivnost

Teorem 2.31 *Za svaki graf G s n vrhova i $\lambda \in \langle 0,1\rangle$ vrijedi*

$$\min_{2\leqslant D\leqslant n-1} \frac{\lambda^D + \frac{1}{n-D}\left(\frac{\lambda^D-\lambda}{\lambda-1}\right)}{D\lambda^D + \frac{1}{n-D}\left[\frac{\lambda-D\lambda^D+(D-1)\lambda^{D+1}}{(\lambda-1)^2}\right]} \leqslant m\rho_\lambda^e(G) \leqslant 1.$$

Donja ograda se postiže za početni vrh metlice, a gornja ograda za bilo koji vrh u vršno tranzitivnom grafu.

Dokaz. Koristeći Teorem 2.15 i Lemu 2.3, vrijedi

$$min\left\{\frac{c_\lambda^e(u)}{t_\lambda^e(u)} : u \in V\right\} \leqslant \frac{\sum\limits_{u\in V} c_\lambda^e(u)}{\sum\limits_{u\in V} t_\lambda^e(u)} = \frac{\sum\limits_{k,l\in V} d(k,l)\cdot\lambda^{d(k,l)}}{\sum\limits_{k,l\in V} d(k,l)\cdot\lambda^{d(k,l)}} = 1.$$

Dokažimo donju ogradu. Neka je G graf za koji se postiže minimalna vrijednost $m\rho_\lambda^e(G)$ i neka je $u \in V(G)$ vrh u grafu za koji je $\rho_\lambda^e(u) = m\rho_\lambda^e(G)$. Vrijedi

$$\rho_\lambda^e(G) = \frac{c_\lambda^e(G)}{t_\lambda^e(G)} \geqslant \frac{\sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\lambda^{d(u,v)}}$$

jer vrh u sigurno leži na svakom najkraćem putu od sebe do svih ostalih vrhova v. Neka je v_D vrh koji je najdalji od u i neka je $S = uv_1v_2...v_D$ najkraći put od u do v_D. Nadalje, neka je $k = n - D - 1$, neka je $\{w_1, ..., w_k\} = V\setminus\{u, v_1, ..., v_D\}$ skup svih vrhova koji ne leže na putu S i neka je $W = \{w_1, w_2, ..., w_k, v_D\}$. Budući je $d(u, v_i) = i$ za svaki $i \in \{1, 2, ..., D\}$, imamo:

$$N_\lambda^e(u) \geqslant \frac{\sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\lambda^{d(u,v)}} = \frac{\sum\limits_{i=1}^{D}\lambda^i + \sum\limits_{i=1}^{k}\lambda^{d(u,w_i)}}{\sum\limits_{i=1}^{D} i\lambda^i + \sum\limits_{i=1}^{k} d(u,w_i)\lambda^{d(u,w_i)}}. \tag{2.9}$$

Zadnji izraz u (2.9) možemo napisati u obliku

$$\frac{\sum\limits_{v\in W}\left(\lambda^{d(u,v)}+\frac{1}{n-D}\sum\limits_{i=1}^{D-1}\lambda^i\right)}{\sum\limits_{v\in W}\left(d(u,v)\lambda^{d(u,v)}+\frac{1}{n-D}\sum\limits_{i=1}^{D-1}i\lambda^i\right)}. \tag{2.10}$$

Koristeći Lemu 2.3, minimum izraza (2.10) je

$$\frac{\lambda^x+\frac{1}{n-D}\sum\limits_{i=1}^{D-1}\lambda^i}{x\lambda^x+\frac{1}{n-D}\sum\limits_{i=1}^{D-1}i\lambda^i},$$

gdje je $x=d(u,q)$ za neki $q\in W$ za koji izraz (2.10) ima najmanju vrijednost. Minimum se postiže ako i samo ako je omjer $\frac{a_i}{b_i}$ konstantan za svaki $i\in\{1,2,...,n\}$. Jedan od načina za to postići je da je $d(u,v)$ konstanta za svaki $v\in W$, tj. da je $d(u,w_i)=d(u,v_D)=D$ za svaki $i\in\{1,2,...,k\}$. To je moguće ako su svi w_i direktno povezani s v_{D-1} za svaki $i\leqslant k$, što vrijedi kada je G metlica. ■

Teorem 2.32 *Za svaki graf G s n vrhova i $\lambda\in\langle 0,1\rangle$ vrijedi*

$$1\leqslant M\rho_\lambda^e(G)\leqslant\frac{1}{2}(n-2)\cdot\lambda+1.$$

Donja ograda se postiže za bilo koji vrh u vršno tranzitivnom grafu, a gornja ograda se postiže za centralni vrh zvijezde.

Dokaz. Dokažimo prvo donju ogradu. Koristeći Teorem 2.15, pošto je maksimum veći ili jednak prosjeku, vrijedi:

$$\max\left\{\frac{c_\lambda^e(u)}{t_\lambda^e(u)}:u\in V\right\}\geqslant\frac{\frac{1}{n}\sum\limits_{u\in V}c_\lambda^e(u)}{\frac{1}{n}\sum\limits_{u\in V}t_\lambda^e(u)}=\frac{\sum\limits_{k,l\in V}d(k,l)\cdot\lambda^{d(k,l)}}{\sum\limits_{k,l\in V}d(k,l)\cdot\lambda^{d(k,l)}}=1.$$

Za gornju ogradu, iz Leme 2.26 zaključujemo da je graf za koji se postiže maksimalna vrijednost $M\rho_\lambda^e(G)$ stablo. Naime, pošto je vršna produktivnost definirana

kao omjer međupoloženosti i transmisije, potrebno je da brojnik ima maksimalnu vrijednost, što vrijedi kada je graf G stablo. Možemo pretpostaviti da je graf koji se pojavljuje u nazivniku također stablo. Ako nije tako, možemo ponoviti konstrukciju grafa G' iz Leme 2.26 da bismo dobili stablo u kojem udaljenosti izmedju u i svih ostalih vrhova ostaju iste, a samim time, i transmisija ostaje ista.

Neka je u vrh za koji vršna produktivnost ima najveću vrijednost. Neka je P skup svih neuređenih parova vrhova $v, w \in V \setminus \{u\}$ takvih da najkraći put od v do w prolazi vrhom u. Vrijedi:

$$\begin{aligned}
\rho^e_\lambda(u) &= \frac{\sum\limits_{k,l\in V} \lambda^{d(k,l)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)}} \leqslant \frac{\sum\limits_{\{v,w\}\in P} \lambda^{[d(v,u)+d(u,w)]} + \sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)}} \\
&= \frac{\sum\limits_{\{v,w\}\in P} \lambda^{d(v,u)}\cdot\lambda^{d(u,w)} + \sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)}} \\
&\leqslant \frac{\frac{1}{2}(n-2)\cdot\lambda\cdot\sum\limits_{v\in V\setminus\{u\}} \lambda^{d(v,u)} + \sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)}} \\
&\leqslant \frac{\left[\frac{1}{2}(n-2)\cdot\lambda+1\right]\sum\limits_{v\in V\setminus\{u\}} \lambda^{d(u,v)}}{\sum\limits_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)}} \leqslant \frac{1}{2}(n-2)\cdot\lambda+1.
\end{aligned}$$

Jednostavan račun pokazuje da jednakost vrijedi za centralni vrh zvijezde. ■

2.2.5 Vršna profitabilnost

Teorem 2.33 *Za svaki graf G s n vrhova i $\lambda \in \langle 0,1\rangle$ vrijedi:*

$$\min_{1\leqslant D\leqslant n-1} \frac{\lambda\left[D\lambda^D - \lambda - (D-1)\lambda^{D+1}\right]}{(\lambda-1)^2} + (n-D-1)(\lambda^D - D\lambda^D) \leqslant m\nu^e_\lambda(G)$$

Donja ograda se postiže za početni vrh metlice. Također vrijedi

$$m\nu^e_\lambda(G) \leqslant 0.$$

Gornja ograda za bilo koji vrh u vršno tranzitivnom grafu.

Dokaz. Dokažimo prvo gornju ogradu. Koristeći Teorem 2.15 vrijedi:

$$\begin{aligned} m\nu_\lambda^e(u) &= \min\{c_\lambda^e(u) - t_\lambda^e(u) : u \in V\} \leqslant \frac{1}{n}\left(\sum_{u\in V}(c_\lambda^e(u) - t_\lambda^e(u))\right) \\ &= \frac{1}{n}\left(\sum_{u\in V} c_\lambda^e(u) - \sum_{u\in V} t_\lambda^e(u)\right) = 0. \end{aligned}$$

Prva nejednakost vrijedi jer je minimum manji ili jednak od prosjeka.

Za donju ogradu, pretpostavimo da je G graf za koji se postiže minimalna vrijednost $m\nu_\lambda^e(G)$ i neka je $u \in V(G)$ vrh u grafu u kojem se ta vrijednost postiže. Neka je v_D vrh koji je najudaljeniji od u i neka je $S = uv_1v_1...v_D$ najkraći put od u do v_D. Nadalje, neka jel $k = n - D - 1$ i neka je $W = \{w_1, w_2, ..., w_k\}$ skup svih vrhova koji ne leže na putu S. Pošto je $d(u, v_i) = i$ za svaki $i \in \{1, 2, ..., D\}$ vrijedi:

$$\begin{aligned} m\nu_\lambda^e(u) &= c_\lambda^e(u) - t_\lambda^e(u) \\ &\geqslant \sum_{i=1}^{D}\lambda^i + \sum_{w\in W}\lambda^{d(u,w)} - \sum_{i=1}^{D} i\cdot\lambda^i - \sum_{w\in W} d(u,w)\lambda^{d(u,w)} \\ &\geqslant \sum_{i=1}^{D}\left(\lambda^i - i\cdot\lambda^i\right) + \sum_{w\in W}\lambda^{d(u,w)}[1 - d(u,w)] \\ &\geqslant \sum_{i=1}^{D}\left(\lambda^i - i\cdot\lambda^i\right) + (n - D - 1)\lambda^x(1 - x). \end{aligned}$$

gdje je $x = d(u, q)$ za neki $q \in W$ za koji izraz $\lambda^{d(u,q)}[1 - d(u, q)]$ ima najmanju vrijednost. To znači da su svi vrhovi u W jednako udaljeni od u. Kao što je dokazano u Teoremu 2.24, u tom slučaju vrijedi $d(u, w) = D$ za svaki $w \in W$, tj. svi vrhovi u W su direktno povezani s v_{D-1}. Zaključujemo da je jedan od grafova za koje se postiže donja ograda metlica. ■

Teorem 2.34 *Za svaki graf G s n vrhova i $\lambda \in \langle 0, 1\rangle$ vrijedi*

$$0 \leqslant M\nu_\lambda^e(G) \leqslant \frac{1}{2}(n-1)(n-2)\cdot\lambda^2.$$

Donja ograda se postiže za bilo koji vrh u vršno tranzitivnom grafu, a gornja ograda se postiže za centralni vrh zvijezde.

Dokaz. Koristeći Teorem 2.15, za donju ogradu vrijedi

$$\begin{aligned} M\nu_\lambda^e(u) &= \max\{c_\lambda^e(u) - t_\lambda^e(u) : u \in V\} \geqslant \frac{1}{n}\left(\sum_{u\in V}(c_\lambda^e(u) - t_\lambda^e(u))\right) \\ &= \frac{1}{n}\left(\sum_{u\in V} c_\lambda^e(u) - \sum_{u\in V} t_\lambda^e(u)\right) = 0. \end{aligned}$$

Prva nejednakost vrijedi jer je maksimum veći ili jednak prosjeku.

Dokažimo sada gornju ogradu. Iz Leme 2.26 očito je da je traženi graf stablo. Neka je G graf za koji je vrijednost $M\nu_\lambda^e(G)$ maksimalna i neka je $u \in V$ vrh takav da je $\nu_\lambda^e(u) = M\nu_\lambda^e(G)$. Neka je P skup svih neuređenih parova vrhova $v, w \in V \setminus \{u\}$ takvih da najkraći put od v do w prolazi vrhom u. Vrijedi:

$$\begin{aligned} m\nu_\lambda^e(u) &= c_\lambda^e(u) - t_\lambda^e(u) \leqslant \sum_{k,l\in V} \lambda^{d(k,l)} - \sum_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)} \\ &\leqslant \sum_{\{v,w\}\in P} \lambda^{[d(v,u)+d(u,w)]} + \sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} - \sum_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)} \\ &\leqslant \sum_{\{v,w\}\in P} \lambda^2 + \sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} - \sum_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)} \\ &\leqslant \frac{1}{2}(n-1)(n-2)\cdot\lambda^2 + \sum_{v\in V\setminus\{u\}} \lambda^{d(u,v)} - \sum_{v\in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)} \\ &\leqslant \frac{1}{2}(n-1)(n-2)\cdot\lambda^2. \end{aligned}$$

■

U ovom poglavlju definirali smo i analizirali eksponencijalne mrežne deskriptore. Svaki od njih nam pruža uvid u važnost pojedinog vrha u mreži. Donje ograde za $Mt_\lambda^e(G)$ i $Mc_\lambda^e(G)$ ostaju kao otvoreni problemi.

Poglavlje 3

Kurikulne mreže

3.1 Uvod

Riječ "curriculum" izvedena je iz latinskog glagola *currere* što znači "trčati, pokrenuti" [17, 127]. Značenje je proširio Ciceron koji povezuje ovaj pojam s "curriculum vitae" koji danas označava životopis, odnosno "tijek nečijeg života". Tek se u 19. stoljeću termin počinje koristiti u području obrazovanja [146]. U hrvatskom jeziku koristi se pojam *"kurikul"*. Velik je broj istraživača i odgajatelja tijekom godina pružao nove uvide u značenje ovog pojma [17, 24, 66, 99, 105]. Iako su do sada napisani mnogi znanstveni radovi, održavane rasprave o filozofiji kurikul(um)a, njegovoj pedagoškoj izvedbi i sadržajnoj strukturi, još uvijek nema jedinstvenog određenja samog pojma. Na primjer, Kelly definira kurikul kao svo učenje koji je planirano i vođeno od strane škole, bilo da se odvija u grupama ili individualno, unutar ili izvan škole [85]. Braslavsky, s druge strane, navodi da je kurikul sporazum između društva, obrazovnih stručnjaka i države o onome što učenici trebaju usvojiti tijekom određenih razdoblja svog života [34]. Unatoč nesuglasicama među brojnim autorima koji su pisali o kurikulu, ipak možemo uočiti nekoliko zajedničkih točaka i komponenti prema kojima bi se moglo kazati da kurikul su-

vremenog odgoja, obrazovanja i škole podrazumijeva znanstveno zasnivanje cilja, zadataka, sadržaja, plana i programa, organizaciju i tehnologiju provođenja, te različite oblike evaluacije učinaka [128].

U prijevodima anglosaksonske literature često se kurikul uzima kao sinonim za nastavni plan i program, dok se u europskoj literaturi za nastavni plan i program koristi termin *syllabus* [138]. Iako se nastavni plan i program i kurikul odnose na iste didaktičke fenomene, među njima se danas prave razlike. One se odnose i na sam pristup izradi i na međusobni odnos [156]. Nastavni plan i program ipak treba shvatiti samo kao dio kurikula, dakle, pojam kurikul ima šire značenje i u sebi sadržava nastavni plan i program za pojedine predmete. Bitna razlika između tradicionalno shvaćenog nastavnog plana i programa i kurikula je ta što je kod nastavnog plana i programa naglasak na sadržaj kojeg učenici moraju naučiti, a kod kurikula je naglasak na cilj koji se treba postići. U ovom ćemo radu češće koristiti pojam kurikul, osim kada je riječ o specifičnom postupku sekvenciranja nastavnih materijala, tj. određivanja redoslijeda edukacijskih jedinica.

Može se reći da je kurikul osmišljen, sustavan i skladno uređen način reguliranja, planiranja, izvedbe i vrednovanja odgojno-obrazovnog procesa koji može biti određen na različitim razinama od cjelokupnog sustava odgoja i obrazovanja, preko određenih njegovih dijelova, odgojno-obrazovne ustanove do pojedinca. Prilikom sastavljanja, potrebno je odrediti opseg (širinu i dubinu sadržaja koji je potrebno odraditi) i redoslijed (poredak kojim se obrađuju nastavne jedinice) nastavnih materijala [106]. Preciznije, proces sastavljanja kurikula možemo podijeliti u četiri osnovne etape:

1. Formulacija ciljeva
2. Odabir sadržaja
3. Odabir metoda

4. Evaluacija

Detaljni pregled pretpostavki i ciljeva koje treba uzeti u obzir prilikom sastavljanja nastavnog plana i programa dan je u nizu radova "Sequencing of contents and learning objects " [173, 174, 175].

U ovom radu koncentrirali smo se na organizaciju nastavnog sadržaja, točnije, određivanje redoslijeda edukacijskih jedinica (sekvenciranje). Riječ je o važnom pedagoškom procesu čija je svrha upravljanje gradivom da bi učenici što lakše i efikasnije postigli zadane ciljeve. Jedan od razloga zašto je ovaj proces iznimno važan jest taj da se većina učenja zasniva na prethodno naučenim pojmovima i konceptima (npr. da bi se razumio koncept i pojam brzine potrebno je usvojiti i razumijeti pojam derivacije). Učenici i nastavnici očekuju da kurikul bude dizajniran tako da olakšava proces učenja i osigurava vremensku efikasnost. Pravilno sekvenciranje nastavnih materijala stvara učinkovitu strukturu za nastavnike i za učenike i omogućava svim sudionicima obrazovnog procesa da lakše dođu do spoznaje što je potrebno naučiti i koje korake moraju svladati da bi se došlo do željenog cilja. Da bi došlo do efektivnog učenja, sadržaj koji se uči mora biti u pravilnom redoslijedu. Brusilovsky razlikuje dvije vrste sekvenciranja: aktivno i pasivno [36]. Aktivno sekvenciranje podrazumijeva cilj učenja (skup pojmova ili tema koje se moraju usvojiti). Sustavi s aktivnim sekvenciranjem mogu izgraditi najbolji pojedinačni put za postizanje cilja . Pasivno sekvenciranje je reaktivan proces i ne zahtijeva aktivni cilj učenja. Ono počinje kada korisnik nije u mogućnosti riješiti problem ili odgovoriti ispravno na neko pitanje. Njegov cilj je ponuditi korisniku podskup dostupnih materijala za učenje kojima može popuniti prazninu u znanju i riješiti se zabluda (miskoncepcija)[36].

Pitanje kako bi materijali trebali biti poredani i organizirani je centralno pitanje edukacijskih debata [35, 50, 63, 148, 149, 152], no odgovor koji bi zadovoljavao sve uključene strane još nije pronađen. Postoji mnogo principa po kojima se moguće

voditi prilikom određivanja redoslijeda nastavih materijala, a većinu ih je moguće opisati sljedećim pitanjima [126]:

1. Koji su empirijski odnosi između edukacijskih jedinica i na koji način je moguće poredati materijale da su konzistentni sa svijetom koji nas okružuje?

2. Koja su konceptualna svojstva znanja koje je potrebno usvojiti i na koji način je moguće poredati materijale da su logički konzistentni?

3. Kako nastaju svojstva i koncepti i na koji način je moguće poredati materijale da su konzistentni s procesom propitivanja i istraživanja?

4. Na koji način učenici uče i na koji način je moguće poredati materijale da su konzistentni s procesom učenja?

5. Kako će učenik iskoristiti stečeno znanje na koji način je moguće poredati materijale da su konzistentni s procesom primjene naučenog?

Novi pristup u obrazovanju je adaptivno učenje koje je prilagođeno pozadini i sklonostima pojedinih učenika. To je u suprotnosti s tradicionalnim načinom sekvenciranja sadržaja koji obično propisuje samo jedan put učenja za skupinu pojedinaca. Sekvenciranje sadržaja je postalo važno polje istraživanja i za mrežne (eng. *web-based*) sustave učenja [145]. Mnogi istraživači usmjereni su na razvoj sustava e-učenja s personaliziranim mehanizmima kako bi se pomoglo učenje i osigurao najlakši i najbrži put za dolazak do određenog nastavnog cilja. Chen, Liu i Chang predstavili su prototip mrežnog sustava s uputama za učenje koji se temelji na personaliziranom sekvenciranju sadržaja vodeći računa o težini gradiva i učenikovim sposobnostima [41]. De Marcos je predložio razvoj sustava koji obavlja postupak sekvenciranja definiran u okvirima učenikovih sposobnosti i kompetencija. Definirao je objekte učenja i odnos između dva objekta ako se jedan objekt

nalazi ispred drugog u valjanom poretku. Prostor rješenja sadrži sve moguće poretke, a rješenje je poredak koji zadovoljava sve propisane odnose [103]. Budući je nastava u mrežnim kolegijima obično nelinearna, javila se potreba za dinamičkim sekvenciranjem dostupnih sadržaja. Jedan od modela za dinamičko sekvenciranje je MANIAC (Multimedia Asynchronous Networked Individualized Courseware), inteligentni tutorski sustav na kojem se temelje mnogi dostupni mrežni tečajevi na World Wide Webu. Nelinearna verzija MANIAC-a pohranjuje teme u obliku semantičke mreže u kojoj vrsta brida prikazuje odnos između teme i njezinih predznanja [143, 144]. Većina ovih pristupa uključuje neku vrstu testiranja studenata, tj. evaluaciju njihovog znanja. Zbog sve veće potrebe za cjeloživotnim obrazovanjem, formalnim i neformalnim, ljudi se sve više okreću samostalnom učenju, posebno mrežnim stranicama s izloženim predavanjima i ponuđenim materijalima, od kojih mnoge opće ne nude sustave za testitranje. Studentu je u mnogim slučajevima ponuđen samo skup edukacijskih jedinica bez odgovarajućeg redoslijeda. Naš pristup zaobilazi ovaj problem dajući mjere koje ne zahtijevaju nikakvu vrstu testiranja ni evaluacije znanja studenata.

Iako su tijekom godina razvijeni različiti pristupi sekvenciranju nastavnih materijala, malo je pristupa temeljeno na svojstvima kompleksnih mreža. U ovom radu osmišljen je okvir, odnosno teorijska podloga za sekvenciranje sadržaja pomoću teorije kompleksnih mreža i teorija grafova. Predstavljene se različite mjere za evaluaciju valjanosti sekvenciranja nastavnih sadržaja i materijala. Ove mjere imaju nekoliko važnih primjena:

1. razvoj programske podrške za automatsko sekvenciranje nastavnih sadržaja
2. potpora stručnjacima prilikom odlučivanja o redoslijedu nastavnih sadržaja
3. objektivne mjere za rješavanje sukoba oko valjanosti suprostavljenih prijedloga za sekvenciranje nastavnih sadržaja

U Republici Hrvatskoj trenutno je u tijeku *Cjelovita kurikularna reforma* koja za cilj ima "uspostavljanje usklađenog i učinkovitog sustava odgoja i obrazovanja kroz cjelovite sadržajne i strukturne promjene, kako bi se:

- Učenicima osiguralo korisnije i smislenije obrazovanje, usklađeno s njihovom razvojnom dobi i interesima te bliže svakodnevnom životu, obrazovanje koje će ih osposobiti za suvremeni život, svijet rada i nastavak obrazovanja,
- Roditeljima omogućilo veću uključenost u obrazovanje djece i život škole, jasno iskazana očekivanja, objektivnije ocjenjivanje i vrednovanje, smislenije i češće povratne informacije o postignućima njihove djece,
- Učiteljima, nastavnicima i ostalim djelatnicima odgojno-obrazovnih ustanova osiguralo osnaživanje uloge i jačanje profesionalnosti, veću autonomiju u radu, kreativniji rad, smanjenje administrativnih obveza, motiviranije učenike i smanjivanje vanjskih pritisaka" [122].

Budući najavljene promjene u prvoj fazi Reforme uključuju izradu kurikula za sve razine u sustavu odgoja i obrazovanja, smatramo da je proučavanje kurikulnih mreža i mjera za vrednovanje sekvenciranja nastavnih materijala i sadržaja od iznimne važnosti za obrazovni sustav i objektivno vrednovanje donesenih nastavnih planova i programa.

U ovom poglavlju analizirali smo standardni problem ekstremalnih grafova primjenjen na predložene mjere. Odredili smo najmanje i najviše složene nastavne planove u odnosu na svaku mjeru. Promatrali smo jednostavne usmjerene povezane grafove i tranzitivno zatvorene grafove. Odredili smo minimalne i maksimalne vrijednosti svake od definiranih mjera te ih analizirali u kontekstu tri različita pristupa. Detaljna matematička formulacija opisana je u sljedećem potpoglavlju.

3.2 Matematička formulacija problema

Definiramo ***kurikulni graf*** kao jednostavni usmjereni povezani graf G čiji vrhovi predstavljaju edukacijske jedinice, a usmjereni brid $uv \in E(G)$ označava da je razumijevanje jedinice u potrebno za učenje i razumijevanje jedinice v. Edukacijska jedinica može predstavljati određeni pojam, koncept, nastavnu temu, nastavnu cjelinu ili pak cijeli nastavni predmet ili kolegij. Na primjer, neka vrh $u \in V(G)$ predstavlja zbrajanje prirodnih brojeva i neka vrh $v \in V(G)$ predstavlja množenje prirodnih brojeva. Tada usmjereni brid uv predstavlja da je poznavanje definicije i svojstava zbrajanja prirodnih brojeva potrebno da bi se naučio i usvojio koncept množenja prirodnih brojeva. Primjetimo da kurikulni graf ne smije sadržavati usmjereni ciklus jer razumijevanje edukacijskih jedinica u tom ciklusu ne bi bilo moguće. Kurikulna ekspozicija je poredak p edukacijskih jedinica na takav način da za svaki usmjereni brid uv jedinica u prethodi jedinici v. Preciznije, neka je G jednostavni usmjereni graf bez usmjerenih ciklusa i neka je $P(G)$ skup svih bijekcija $p : V(G) \to \{1, ..., n\}$ takvih da vrijedi $p(u) < p(v)$ za svaki usmjereni brid $uv \in E(G)$.

Napomena 3.1 *Dokažimo da je skup P neprazan. Naime, pošto kurikulni graf G nema usmjerenih ciklusa, postoji barem jedan vrh u_1 čiji je instupanj jednak 0. Neka je $p(u_1) = 1$. Pošto digraf $G - u_1$ nema usmjerenih ciklusa, postoji barem jedan vrh u_2 s instupnjem 0 u $G - u_1$. Neka je $p(u_2) = 2$. Nadalje, pošto digraf $G-u_1-u_2$ nam usmjerenih ciklusa, sigurno postoji barem jedan vrh $u_3 \in G-u_1-u_2$ s instupnjem 0 u $G-u_1-u_2$. Neka je $p(u_3) = 3$. Nastavimo li ovaj proces moguće je konstruirati $p \in P$.*

Problem određivanja elemenata skupa P poznat je kao problem topološkog sortiranja. Za dani broj $n \in \mathbb{N}$ i skup parova $(i, j) \in \mathbb{N}^2$ takvih da je $1 \leq i, j \leq n$, problem topološkog sortiranja svodi se na pronalaženje permutacija skupa $\{1, 2, ..., n\}$

takvih da se i nalazi lijevo od j za sve dane parove (i, j). Obično se relacija između i i j obilježava s $i \prec j$ i kažemo da *"i prethodi j"*. Problem topološkog sortiranja ekvivalentan je problemu raspoređivanja vrhova usmjerenog grafa u liniju tako da svi usmjereni bridovi pokazuju s lijeva na desno. Dobro je poznato da je takav raspored moguć ako i samo ako u grafu nema usmjerenih ciklusa. U matematičkim terminima, problem se svodi na umetanje parcijalnog uređaja u linearni (potpuni) uređaj [89].

Jedan od poznatijih algoritama za topološko sortiranje predstavio je Kahn 1962. godine [84] i temelji se na ideji opisanoj u Napomeni 3.1. Algoritam uzastopno uklanja iz grafa vrhove instupnja 0 i bridove incidentne s njima i postavlja ih u odgovarajući poredak. Ovisno o redoslijedu kojim se vrhovi uklanjaju iz grafa, kreiraju se različita rješenja. Algoritam je brz i efikasan i ima računalnu složenost $O(\mid V \mid + \mid E \mid)$. Pseudokod je dan u Algoritmu 1.

Zanima nas koliko brzo i koliko blizu učenici moraju naučiti jedinice koje su snažno povezane. Pretpostavljamo da je učeniku teže naučiti neku novu lekciju ili usvojiti novi pojam ako je za njeno razumijevanje potrebno znanje jedinice koju je učenik savladao davno prije jer je moguće da je neke osnovne koncepte već zaboravio. Naš je zadatak definirati mjere složenosti ekspozicije edukacijskih jedinica. Označimo s $d_{in}(v)$ i $d_{out}(v)$ redom instupanj i outstupanj vrha $v \in V(G)$ u kurikulnom grafu G.

Algorithm 1 Kahnov algoritam za topološko sortiranje

Input: Lista S vrhova s instupnjem 0

Output: Lista L koja će sadržavati sortirane elemente

```
1: while S je neprazan do
2:    ukloni vrh u iz skupa S
3:    dodaj vrh u na kraj liste L
4:    for all vrh v takav da postoji brid e = uv do
5:       ukloni brid e iz grafa
6:       if vrh v ima instupanj 0 then
7:          dodaj v u skup S
8:       end if
9:    end for
10: end while
11: if graf ima još bridova then
12:    vrati grešku (graf sadrži bar jedan ciklus)
13: else
14:    vrati L
15: end if
```

Definiramo:

$$s_{p,G}(v) = \sum_{uv \in E(G)} [p(v) - p(u)]$$

$$a_{p,G}(v) = \begin{cases} \frac{s_{p,G}(v)}{d_{in}(v)} & \text{ako je } d_{in}(v) > 0, \\ 0 & \text{ako je } d_{in}(v) = 0 \end{cases}$$

$$m_{p,G}(v) = \max_{uv \in E(G)} \{p(v) - p(u)\}.$$

Svaka od ovih mjera nam daje svojevrstan uvid u složenost edukacijske jedinice $u \in V(G)$. Prva mjera je suma koliko prije su povezane jedinice naučene. Druga daje prosjek, a treća traži najstarije znanje potrebno za razumijevanje jedinice $u \in V(G)$ jer je moguće da je djelomično zaboravljeno. Koristeći ove mjere, želimo ocijeniti složenost cijelog nastavnog plana. Proučit ćemo više različitih pristupa. Možemo smatrati da je složenost nastavnog plana jednostavno suma složenosti pojedinih edukacijskih jedinica (ili do na linearnu konstantu prosječna složenost edukacijskih jedinica) [10]. U skladu s tim definiramo

$$s_p^1(G) = \frac{\sum\limits_{v \in V(G)} s_{p,G}(v)}{card\,(V(G))}$$

$$a_p^1(G) = \frac{\sum\limits_{v \in V(G)} a_{p,G}(v)}{card\,(V(G))}$$

$$m_p^1(G) = \frac{\sum\limits_{v \in V(G)} m_{p,G}(v)}{card\,(V(G))}$$

Nadalje, složenost nastavnog plana možemo definirati kao složenost najkompleksnije edukacijske jedinice koju sadrži [10] pa definiramo

$$s_p^\infty(G) = \max_{v \in V(G)} \{s_{p,G}(v)\}$$
$$a_p^\infty(G) = \max_{v \in V(G)} \{a_{p,G}(v)\}$$
$$m_p^\infty(G) = \max_{v \in V(G)} \{m_{p,G}(v)\}$$

Štoviše, možemo promatrati i α-sredinu za $\alpha > 1$ da bismo dobili mjere složenosti između ova dva ekstremna pristupa. U skladu s tim definiramo:

$$s_p^\alpha(G) = \left(\frac{\sum\limits_{v \in V(G)} s_{p,G}^\alpha(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

$$a_p^\alpha(G) = \left(\frac{\sum\limits_{v \in V(G)} a_{p,G}^\alpha(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

$$m_p^\alpha(G) = \left(\frac{\sum\limits_{v \in V(G)} m_{p,G}^\alpha(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

Primjetimo da slučaj $\alpha = 1$ odgovara prosječnoj složenosti edukacijske jedinice. Kada α teži u ∞, ove mjere teže u složenost najsloženije jedinice. Zaista, kada je $\alpha \in \langle 1, \infty \rangle$ uzete su u obzir sve mjere između dva navedena ekstremna pristupa. Cilj je osmisliti optimalnu kurikulnu ekspoziciju, tj. ekspoziciju edukacijskih jedinica takvu da je složenost nastavnog plana minimalna. Takve minimalne složenosti nazivamo **retencijske složenosti kurikula**. Za $\alpha \geq 1$ definiramo:

$$s^\alpha(G) = \min_{p \in P(G)} \left\{ s_p^\alpha(G) \right\}$$

$$a^\alpha(G) = \min_{p \in P(G)} \left\{ a_p^\alpha(G) \right\}$$

$$m^\alpha(G) = \min_{p \in P(G)} \left\{ m_p^\alpha(G) \right\}.$$

Promotrimo zanimljiv problem koji se pojavljuje prilikom sastavljanja nastavnog plana. Pretpostavimo da je jedinica A preduvjet za učenje jedinice B i neka je jedinica B preduvjet za učenje jedinice C. Možemo tvrditi da je očito da je onda jedinica A preduvjet za jedinicu C i da u kurikulnom grafu G mora postojati usmjereni brid AC. S druge strane, možemo tvrditi da je učenik prilikom učenja jedinice B ponovio jedinicu A i da nema potrebe staviti usmjereni brid između jedinica A i C. Kao ekstremni primjer, možemo tvrditi da bi trebao postojati brid između edukacijskih jedinica "zbrajanje prirodnih brojeva" i "rješavanje parcijalnih diferencijalnih jednadžbi" iako u većini slučajeva prođe otprilike 14 godina školovanja od jedne do druge jedinice i potpuno opravdano možemo tvrditi da nema potrebe da učenik ponovi zbrajanje prirodnih brojeva prije učenja parcijalnih diferencijalnih jednadžbi. Dakle, u različitim situacijama možemo opravdano

donijeti različite odluke. Iz navedenih razloga promatramo dvije vrste familija grafova:

1. Tip A - slabo povezani usmjereni grafovi koji ne sadrže usmjeren cikluse i nisu tranzitivno zatvoreni

2. Tip B - slabo povezani usmjereni grafovi koji ne sadrže usmjeren cikluse i tranzitivno su zatvoreni.

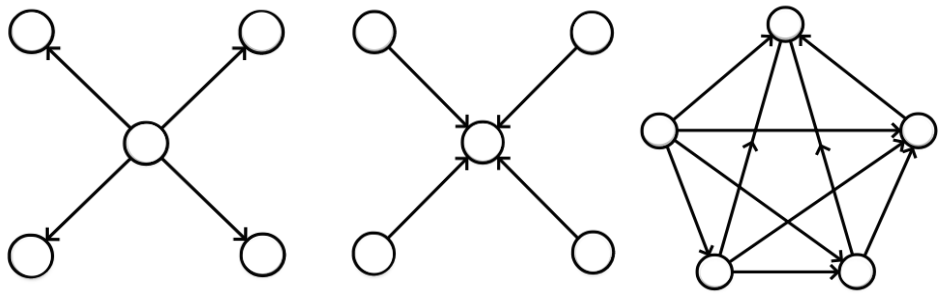
Nadalje, primjetimo da ako kurikulni graf nije (slabo) povezan, možemo ga podijeliti na njegove slabo povezane komponente.
U ovom poglavlju proučavat ćemo ekstremalne kurikulne grafove, tj. kurikulne grafove koji odgovaraju nastavnim planovima s najmanjom i najvećom složenosti u odnosu na sve definirane mjere i pristupe.

3.3 Ekstremalni rezultati

Definirajmo prvo neke kurikulne grafove koji se često pojavljuju prilikom rješavanja problema. Jednostavni primjeri prikazani su na Slikama 3.1 i 3.2.

Neka je S_n usmjereni graf čiji je pripadni graf zvijezda i svi bridovi su usmjereni od centra zvijezde prema listovima (Slika 3.1 a)). Neka je T_n usmjereni graf čiji je pripadni graf zvijezda i svi bridovi su usmjereni od listova zvijezde prema centru (Slika 3.1 b)). Neka je O_n usmjereni graf koji odgovara totalnom linearnom poretku n elemenata (ekvivalentno, neka je O_n jednostavni povezani graf bez usmjerenih ciklusa čiji je pripadni graf potpun ili ekvivlentno, neka je O_n turnir bez usmjerenih ciklusa). Primjer grafa O_n prikazan je na Slici 3.1 c). Neka je P_n usmjereni put, tj. digraf s n vrhova čiji je pripadni graf put i za svaki i vrijedi da je usmjereni brid $e_i = (v_{i-1}, v_i)$. Neka je DP_n usmjereni graf s n vrhova čiji je pripadni graf

put, ali ne postoje dva vrha $u, v \in V(DP_n)$ takvi da je $d(u, v) \geq 2$ (neki primjeri prikazani su na Slici 3.2).



(a) $G = S_n$ (b) $G = T_n$ (c) $G = O_n$

Slika 3.1: Jednostavni primjeri kurikulnih grafova s $n = 5$ vrhova

(a) Primjer grafa DP_n za $n = 5$ vrhova (b) Primjer grafa DP_n za $n = 5$ vrhova

Slika 3.2: Primjer dvije različite orijentacije na grafu DP_n za $n = 5$ vrhova

Razmotrit ćemo prvo grafove tipa A i odrediti gornje i donje ograde za svaku pojedinu mjeru te navesti po jedan primjer kurikulnog grafa u kojem se te vrijednosti postižu.

Teorem 3.2 *Neka je G kurikulni graf tipa A s $n \geq 3$ vrhova. Vrijedi:*

1. $1 \leq m^{\infty}(G) \leq n - 1$.
   Donja ograda se postiže za $G = P_n$, a gornja ograda za $G = O_n$.
2. $1 \leq s^{\infty}(G) \leq \sum_{i=1}^{n-1} i$.
   Donja ograda se postiže za $G = P_n$, a gornja ograda za $G = O_n$.

3. $1 \leq a^{\infty}(G) \leq n-1$.

Donja ograda se postiže za $G = P_n$*, a gornja ograda za* $G = S_n$*.*

Dokaz. Po definiciji, za kurikulni graf G i za svaki $uv \in E(G)$ vrijedi $p(u) < p(v)$ pa je $p(v) - p(u) > 0$. Nadalje, vrijedi $p(v) - p(u) < n$. Iz ovoga zaključujemo da je $1 \leq m^{\infty}(G) \leq n-1$. Lako se provjeri da se donja ograda postiže za P_n, a gornja za O_n. Time je dokazana tvrdnja 1. Ostale tvrdnje su trivijalne. ■

Teorem 3.3 *Neka je* G *kurikulni graf tipa* A *s* $n \geq 3$ *vrhova i neka je* $\alpha \geq 1$*. Vrijedi*

$$\left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}} \leq m^{\alpha}(G) \leq \left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n}\right)^{\frac{1}{\alpha}}.$$

Donja ograda se postiže za $G = P_n$*, a gornja ograda za* $G = O_n$*.*

Dokaz. Neka je G kurikulni graf s $n \geq 3$ vrhova i neka je p bijekcija koja minimizira $m^{\alpha}(G)$. Primjetimo da graf G ima barem $n-1$ usmjerenih bridova pošto je slabo povezan. Vrijedi

$$m_p^{\alpha}(G) = \left(\frac{\sum_{v \in V(G)} m_{p,G}^{\alpha}(v)}{n}\right)^{\frac{1}{\alpha}} \geq \left(\frac{\sum_{v \in V(G)} (d_G^+(v))^{\alpha}}{n}\right)^{\frac{1}{\alpha}}$$

Funkcija $f(x) = x^{\alpha}$ je konveksna funkcija za $\alpha > 1$ i postoji barem jedan vrh u G instupnja 0 pa je:

$$\left(\frac{\sum_{v \in V(G)} (d_G^+(v))^{\alpha}}{n}\right)^{\frac{1}{\alpha}} \geq \left(\frac{(n-1)\left(\frac{\sum_{v \in V(G)} d_G^+(v)}{n-1}\right)^{\alpha}}{n}\right)^{\frac{1}{\alpha}} \geq$$

$$\geq \left(\frac{(n-1)\left(\frac{e(G)}{n-1}\right)^{\alpha}}{n}\right)^{\frac{1}{\alpha}} \geq \left(\frac{(n-1)\left(\frac{n-1}{n-1}\right)^{\alpha}}{n}\right)^{\frac{1}{\alpha}} = \left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}}$$

Gornja ograda slijedi jer vrijedi $p(v) - p(u) \leq p(v) - 1$ za svaka dva vrha $u, v \in V(G)$, što je ispunjeno ako vrh u, takav da je $p(u) = 1$, pokazuje na sve ostale vrhove. ■

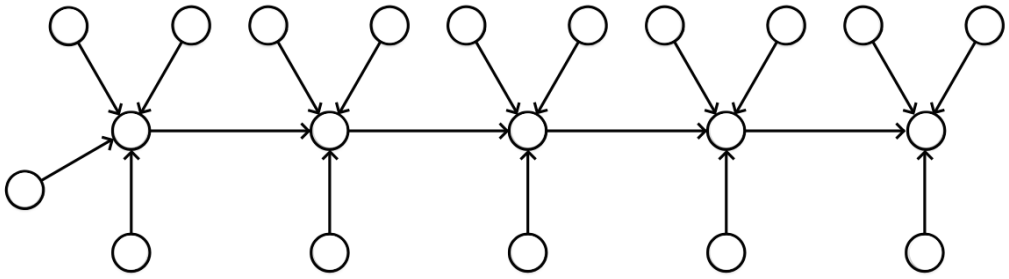
Teorem 3.4 *Neka je G kurikulni graf tipa A s $n \geq 3$ vrhova i neka je $\alpha \geq 1$. Vrijedi*

$$\left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}} \leq s^{\alpha}(G) \leq \left(\frac{\sum_{i=2}^{n}\left(\sum_{j=1}^{i-1} j\right)^{\alpha}}{n}\right)^{\frac{1}{\alpha}}.$$

Donja ograda se postiže za $G = P_n$, a gornja ograda se postiže za $G = O_n$.

Dokaz. Donja ograda slijedi iz Teorema 3.3. Gornja ograda je trivijalna pošto je vrijednost $s^{\alpha}_{p,G}(v)$ najveća ako svi mogući bridovi pokazuju na vrh $v \in V(G)$. ■

Označimo s $T_{n,k}$ graf dobiven iz grafa P_k dodavanjem preostalih $n - k$ vrhova na takav način da svi bridovi pokazuju na vrhove na putu P_k i svaki vrh na putu P_k ima jednak instupanj. Primjer takvog grafa prikazan je na Slici 3.3.



Slika 3.3: Primjer grafa $G = T_{n,k}$ za $k = 5$ i $n = 21$. Graf je dobijen tako da se na usmjereni put duljine 5 doda preostalih 17 vrhova tako da svaki vrh na usmjerenom putu ima instupanj jednak 4.

Teorem 3.5 *Neka je G kurikulni graf tipa A s $n \geq 3$ vrhova i neka je $\alpha \geq 1$. Vrijedi*

$$a^{\alpha}(G) \geq \begin{cases} \frac{n}{2n^{\frac{1}{\alpha}}} & \text{ako je } \alpha \leq \frac{1}{n-1} + 1 \\ \frac{\alpha}{2(\alpha-1)} \left[\frac{(n-1)(\alpha-1)}{n}\right]^{\frac{1}{\alpha}} & \text{ako je } \frac{1}{n-1} + 1 < \alpha < 2 \\ \left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}} & \text{ako je } \alpha \geq 2. \end{cases}$$

Donja ograda se postiže za $G = T_n$ ako je $\alpha \leq 1/(n-1)+1$, za $G = T_{n,k}$ gdje je $k = (n-1)(\alpha-1)$ i $1/(n-1)+1 < \alpha < 2$ ili za $G = P_n$ ako je $\alpha \geq 2$.

Dokaz. Neka je G kurikulni graf s $n \geq 3$ vrhova i neka je p bijekcija koja minimizira $a^{\alpha}(G)$. Graf G ima barem $n-1$ usmjerenih bridova pošto je slabo povezan. Vrijedi

$$a_p^{\alpha}(G) = \left(\frac{\sum\limits_{v \in V(G)} a_{p,G}^{\alpha}(v)}{n} \right)^{\frac{1}{\alpha}} \geq \left[\frac{\sum\limits_{\substack{v \in V(G) \\ d_G^+(v)=0}} 0 + \sum\limits_{\substack{v \in V(G) \\ d_G^+(v) \neq 0}} \left(\frac{\sum\limits_{i=1}^{d_G^+(v)} i}{d_G^+(v)} \right)^{\alpha}}{n} \right]^{\frac{1}{\alpha}} \geq \left[\frac{\sum\limits_{\substack{v \in V(G) \\ d_G^+(v) \neq 0}} \left(\frac{d_G^+(v)+1}{2} \right)^{\alpha}}{n} \right]^{\frac{1}{\alpha}}$$

Neka je $q \in \{1, ..., n-1\}$ broj vrhova $v \in V(G)$ takvih da je $d_G^+(v) \neq 0$. Vrijedi:

$$a_p^{\alpha}(G) \geq \left[\frac{q \left(\frac{e(G)+q}{2q} \right)^{\alpha}}{n} \right]^{\frac{1}{\alpha}} \geq \left[\frac{q \left(\frac{n-1+q}{q} \right)^{\alpha}}{2^{\alpha} n} \right]^{\frac{1}{\alpha}}$$

Potrebno je minimizirati funkciju

$$f(q) = q \left(\frac{n-1+q}{q} \right)^{\alpha}.$$

Prva derivacija nam daje minimum za $q = (n-1)(\alpha-1)$. Pošto vrijedi da je $q \in \{1, ..., n-1\}$, razlikujemo sljedeće slučajeve:

1. Ako je $(n-1)(\alpha-1) \leq 1$ tada će vrijednost $a^\alpha(G)$ biti minimalna za $q = 1$

2. Ako je $1 < (n-1)(\alpha-1) < n-1$ tada će vrijednost $a^\alpha(G)$ bit minimalna ako su q i $1/(\alpha-1)$ prirodni brojevi i $q \mid (n-1)$

3. Ako je $(n-1)(\alpha-1) \geq n-1$ tada će vrijednost $a^\alpha(G)$ biti minimalna za $q = n-1$.

Donje ograde se lako izračunaju iz ovog razmatranja. ■

Teorem 3.6 *Neka je G kurikulni graf tipa A s $n \geq 3$ vrhova i neka je $\alpha \geq 1$. Vrijedi*

$$a^\alpha(G) \leq \left(\frac{\sum_{i=1}^{n-1} i^\alpha}{n} \right)^{\frac{1}{\alpha}}.$$

Gornja ograda se postiže za $G = S_n$.

Dokaz. Trivijalan. ■

Promotrimo sada kurikulne grafove tipa B. Graf tipa B je slabo povezan usmjeren graf koji ne sadrži usmjereni ciklus i koji je tranzitivno zatvoren. Koristeći Teoreme 3.2 do 3.6 dolazimo do sljedećeg korolara:

Korolar 3.7 *Neka je G kurikulni graf tipa B s $n \geq 3$ vrhova i neka je $\alpha \geq 1$. Vrijedi:*

1. $m^\infty(G) \leq n-1$. *Gornja ograda se postiže za $G = O_n$.*

2. $s^\infty(G) \leq \sum_{i=1}^{n-1} i$. *Gornja ograda se postiže za $G = O_n$.*

3. $a^\infty(G) \leq n-1$. *Gornja ograda se postiže zar $G = S_n$.*

4. $m^{\alpha}(G) \leq \left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$. *Gornja ograda se postiže za* $G = O_n$.

5. $s^{\alpha}(G) \leq \left(\frac{\sum_{i=2}^{n} \left(\sum_{j=1}^{i-1} j \right)^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$. *Gornja ograda se postiže za* $G = O_n$.

6. $a^{\alpha}(G) \leq \left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$. *Gornja ograda se postiže za* $G = S_n$.

Definiramo graf $UT(G)$ kao graf dobiven sukcesivnom eliminacijom tranzitivnih bridova, tj. dobiven pomoću sljedećeg algoritma:

1. Neka je $G_1 = G$, $i = 1$, *proceed=true*
2. Dok vrijedi *proceed*
   - 2.1. Ako postoje vrhovi $u_1, u_2, ..., u_k \in V(G_1)$ takvi da je $u_1u_2, u_2u_3, ..., u_{k-1}u_k \in E(G_i)$ i $u_1u_k \in E(G_i)$ tada
     - 2.1.1. $G_{i+1} = G_i - u_1u_k$ i $i = i + 1$
   - 2.2. Inače
     - 2.2.1. *proceed=false*
3. $UT(G) = G_i$

Primjetimo da, ako je graf G slabo (jako) povezan, onda je i $UT(G)$ također slabo (jako) povezan.

Teorem 3.8 *Neka je G kurikulni graf tipa B s $n \geq 2$ vrhova. Vrijedi:*

$$\frac{2n-3}{n} \leq s^1(G)$$

Donja ograda se postiže za $G = DP_n$.

Dokaz. Dokazat ćemo tvrdnju indukcijom po n. Za $n = 2$ tvrdnja očito vrijedi. Pretpostavimo da smo tvrdnju dokazali za grafove s $n-1$ vrhova, $n \geq 3$, i neka graf G ima n vrhova. Neka je $u \in V(G)$ vrh koji nije rezni u grafu $UT(G)$ (tj. vrh takav ga je graf $UT(G) - u$ slabo povezan). Iz pretpostavke indukcije slijedi

$$s^1(G-u) \geq \frac{2n-5}{n-1}.$$

Ako je $d_G^+(u) + d_G^-(u) \geq 2$, tada postoje barem dva usmjerena brida više u G nego u $G-u$. Neka je q permutacija koja minimizira $s_p^1(G)$ i neka je r permutacija koja minimizira $s_p^1(G-u)$. Vrijedi:

$$\begin{aligned} s^1(G) = s_q^1(G) &\geq \frac{2 + (n-1)s^1_{q/(V(G)-u)}(G-u)}{n} \geq \frac{2 + (n-1)s_r^1(G-u)}{n} \\ &\geq \frac{2 + (2n-5)}{n} = \frac{2n-3}{n}. \end{aligned}$$

Ako vrijedi $d_G^+(u) + d_G^-(u) = 1$, razlikujemo dva slučaja: $uv \in E(G)$ i $vu \in E(G)$. Neka je $uv \in E(G)$. Slijedi da postoji vrh w takav da je $wv \in E(UT(G))$. Neka je q permutacija koja minimizira $s_q^1(G)$.
Ako je $q(v) - q(u) \geq 2$, tada je

$$\begin{aligned} s_q^1(G) &\geq \frac{[q(v) - q(u)] + (n-1) \cdot s^1_{q/G-u}(G)}{n} \geq 2 + (n-1) \cdot s^1(G) \\ &\geq \frac{2 + 2n - 5}{n} = \frac{2n-3}{n} \end{aligned}$$

Ako je $q(v) - q(u) = 1$, označimo s $r : V(G) \to \bullet$ funkciju definiranu s:

$$r(x) = \begin{cases} q(x), & \text{ako je } q(x) < q(u) \\ q(x) - 1, & \text{ako je } q(x) > q(u). \end{cases}$$

Primjetimo da za svaki brid $x_1x_2 \in E(G)$ vrijedi

$$r(x_2) - r(x_1) \leq q(x_2) - q(x_1);$$

$$r(v) - r(w) \leq q(v) - q(w) - 1.$$

Dakle, vrijedi

$$\begin{aligned} s_q^1(G) &\geq \frac{[q(v) - q(u)] + (n-1) \cdot s_{q/G-u}^1(G)}{n} \geq \frac{1 + (n-1) \cdot \left(s_r^1(G) + \frac{1}{n-1}\right)}{n} \\ &\geq \frac{1 + (n-1) \cdot \left(s^1(G) + \frac{1}{n-1}\right)}{n} \geq \frac{1 + (n-1) \cdot \left(\frac{2n-5}{n-1} + \frac{1}{n-1}\right)}{n} = \frac{2n-3}{n}. \end{aligned}$$

Ovim smo dokazali traženu nejednakost. Lako se provjeri da jednakost vrijedi za slučaj kada je $G = DP_n$.
Ako je $vu \in E(G)$, dokaz se provodi analogno kao u prethodnom slučaju. ■

Teorem 3.9 *Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova. Vrijedi:*

$$\frac{n-1}{n} \leq m^1(G)$$

Donja ograda se postiže za $G = T_n$.

Dokaz. Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova i neka je p bijekcija koja minimizira $m_p^1(G)$. Graf G ima barem $n-1$ usmjereni brid. Vrijedi:

$$m_p^1(G) = \frac{\sum\limits_{v \in V(G)} m_{p,G}(v)}{n} \geq \frac{\sum\limits_{v \in V(G)} d_G^+(v)}{n} \geq \frac{e(G)}{n} \geq \frac{n-1}{n}.$$

■

Teorem 3.10 *Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova. Vrijedi:*

$$a^1(G) \geq \frac{1}{2}$$

Donja ograda se postiže za $G = T_n$.

Dokaz. Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova i neka je p bijekcija koja minimizira $m_p^1(G)$. Graf G ima barem $n-1$ usmjereni brid. Vrijedi:

$$a_p^1(G) = \frac{\sum\limits_{v \in V(G)} a_{p,G}(v)}{n} \geq \frac{\sum\limits_{\substack{v \in V(G) \\ d_G^+(v)=0}} 0 + \sum\limits_{\substack{v \in V(G) \\ d_G^+(v) \neq 0}} \left(\frac{\sum\limits_{i=1}^{d_G^+(v)} i}{d_G^+(v)} \right)}{n} \geq \frac{\sum\limits_{\substack{v \in V(G) \\ d_G^+(v) \neq 0}} \left(\frac{d_G^+(v)+1}{2} \right)}{n}$$

Neka je $q \in \{1, ..., n-1\}$ broj vrhova v takvih da je $d_G^+(v) \neq 0$. Vrijedi:

$$\frac{\sum\limits_{\substack{v \in V(G) \\ d_G^+(v) \neq 0}} \left(\frac{d_G^+(v)+1}{2} \right)}{n} \geq \frac{n-1+q}{2n} \geq \frac{n-1+1}{2n} = \frac{1}{2},$$

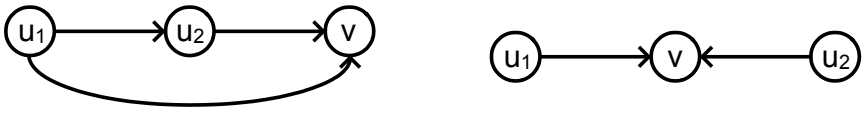
■

Teorem 3.11 *Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova. Vrijedi:*

$$s^\infty(G) \geq 4.$$

Donja ograda se postiže za $G = DP_n$.

Dokaz. Neka je kurikulni graf G tipa B s $n \geq 5$ vrhova takav da vrijedi $s^\infty(G) < 4$. Lako se vidi da je $s^\infty(G) > 2$ pa stoga pretpostavimo da je $s^\infty(G) = 3$. Razlikujemo tri slučaja. Prvo, razmotrimo dva slučaja prikazana na Slici 3.4.



(a) Slučaj 1 (b) Slučaj 2

Slika 3.4: Dva primjera poretka vrhova $u_1, u_2, v \in V(G)$ takvih da je $s_{p,G}(v) = 3$ u kurikulnom grafu G

Slučaj 1 *Pretpostavimo da u G postoji usmjereni put duljine 2.*

Označimo vrhove kao na Slici 3.4 a). Ako označimo $p(v) = a$ tada vrijedi $p(u_2) = a - 1$ i $p(u_1) = a - 2$. Pošto graf G ima $n \geq 5$ vrhova, postoje barem još tri vrha $u_3, u_4, u_5 \in V(G)$. Ne može postojati usmjereni brid koji počinje u u_3 i završava u u_1, u_2 ili v jer bi to povećalo vrijednost $s_{p,G}(v)$. Također, ne postoji usmjereni brid koji počinje u u_2 ili v i pokazuje na u_3. U tom bi slučaju vrijedilo $s_{p,G}(u_3) > s_{p,G}(v)$. Dakle, mora postojati usmjereni brid od u_1 do u_3 i $p(u_3) = a+1$. Promotrimo vrh u_4. Iz već navedenih razloga, može postojati jedino usmjereni brid između u_3 i u_4. Ako postoji usmjereni brid u_3u_4, zbog tranzitivnosti bi postojao i usmjereni brid u_1u_4 i vrijedilo bi $s_{p,G}(u_4) > s_{p,G}(v)$. Ako postoji usmjereni brid u_4u_3, tada je $s_{p,G}(u_3) > s_{p,G}(v)$ jer je $p(u_4) < a - 2$. U svakom slučaju, došli smo do kontradikcije.

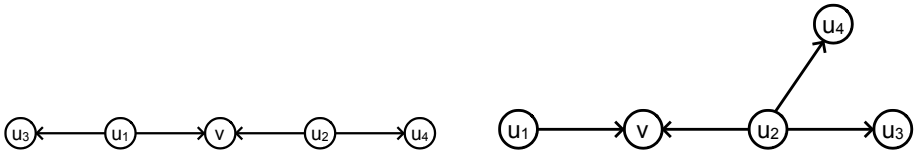
Slučaj 2 *Pretpostavimo da u G ne postoji usmjereni put duljine 2 i da postoji vrh instupnja 2*

Označimo vrhove kao na Slici 3.4 b). Bez gubitka općenitosti možemo pretpostaviti da vrijedi $p(u_1) = a - 2$ i $p(u_2) = a - 1$. Mora postojati vrh $u_3 \in V(G)$, ali ne postoji usmjereni brid koji počinje u u_3 i završava u u_1, u_2 ili v jer bi to povećalo vrijednost $s_{p,G}(v)$. Također, ne postoji usmjereni brid vu_3 jer bi tada postojao put duljine 2. Promotrimo ostale mogućnosti $u_1u_3 \in E(G)$ ili $u_2u_3 \in E(G)$.

1. Pretpostavimo da postoji brid u_1u_3. Mora postojati vrh $u_4 \in V(G)$. Ne postoji usmjereni brid u_4v jer bi to povećalo vrijednost $s_{p,G}(v)$, Također, ne postoji usmjereni brid u_4u_3 jer je $p(u_4) < a - 2$. Dakle, mora postojati usmjereni brid u_2u_4.

2. Pretpostavimo da postoji usmjereni brid u_2u_3. Analogno prethodnim razmatranjima, može postojati jedino usmjereni brid u_2u_4.

Oba slučaja prikazana su na Slici 3.5.



(a) Slučaj 1 (b) Slučaj 2

Slika 3.5: Mogući slučajevi prilikom konstrukcije kurikulnog grafa G s $n = 5$ vrhova takvog da je $s^{\infty}(G) = 3$ objašnjeni u prethodnom razmatranju

Ni u jednom od ovih slučaja ne možemo dodati još vrhova u $V(G)$ na način da $s^{\infty}(G) = s_{p,G}(v)$ zadrži vrijednost 3.

Slučaj 3 *Pretpostavimo da u G ne postoji usmjereni put duljine 2 i da ne postoji vrh u $V(G)$ s instupnjem većim od 1*

Označimo s $u, v \in V(G)$ vrhove takve da je $p(u) = p(v) - 3$ i $uv \in E(G)$. Označimo $p(v) = a$. Moraju postojati dva vrha $u_1, u_2 \in V(G)$ takvi da je $p(u_1) = a - 1$ i $p(u_2) = a - 2$ i bridovi $uu_1, uu_2 \in E(G)$. Pošto je $n \geq 5$ mora postojati vrh u_3 i usmjereni brid uu_3. Ali, $p(u_3) = a + 1$ pa je $s_{p,G}(u_3) > s_{p,G}(v)$ što je opet kontradikcija.

Zaključujemo da ni u kojem slučaju nije moguće konstruirati kurikulni graf G takav da je $s^{\infty} < 4$. ■

Teorem 3.12 *Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova. Vrijedi:*

$$2 \leq a^{\infty}(G).$$

Donja ograda se postiže za $G = DP_n$.

Dokaz. Donja ograda slijedi iz Teorema 3.11. ■

Teorem 3.13 *Neka je G kurikulni graf tipa B s $n \geq 5$ vrhova. Vrijedi:*

$$3 \leq m^{\infty}(G).$$

Donja ograda se postiže za $G = DP_n$.

Dokaz. Donja ograda slijedi iz Teorema 3.11. ■

U ovom poglavlju definirali smo kurikulne grafove kao jednostavne usmjerene grafove u kojima vrhovi predstavljaju edukacijske jedinice, a usmjereni bridovi povezanost dviju edukacijskih jedinica u smislu da je jedna predznanje za drugu. Edukacijska jedinica može predstavljati neki pojam, nastavnu temu, nastavnu cjelinu ili čak cijeli kolegij. Razmatrali smo složenosti pojedine edukacijske jedinice sa tri različita stajališta kao i složenost cjelokupnog nastavnog plana s naglaskom na sekvenciranje nastavnog sadržaja i nastavnih materijala. Budući u različitim situacijama možemo opravdano izabrati različite pristupe sastavljanju nastavnog plana i sekvenciranju nastavnog sadržaja, definirali smo različite mjere složenosti koje pokrivaju široki spektar pristupa i nude nastavnicima, profesorima i stručnjacima na izbor da sami odluče koje mjere smatraju korisnima u svom radu. Promotrili smo mjere složenosti s više različitih aspekata te smo se pozabavili problemom tranzitivno zatvorenih kurikulnih grafova. Odredili smo gornje i donje ograde za svaku pojedinu mjeru za grafove tipa A i B, za različite vrijednosti parametra α, te odredili neke primjere kurikulnih mreža (grafova) za koje se ekstremalne vrijednosti postižu.

3.4 Neka svojstva kurikulnih mreža

U ovom poglavlju proučit ćemo postoje li neka standardna svojstva koja se pojavljuju u kurikulnim mrežama. Analizirat ćemo stvarne kurikulne mreže iz područja matematike, fizike, informatike i biologije koje se mogu pronaći u Prilozima. Izuzeta je rasprava o podjeli na zajednice pošto će u sljedećem poglavlju ta tematika biti detaljno istražena.

Jedan od najvažnijih pojmova vezanih za grafove je matrica susjedstva $\mathbf{A}$. Za jednostavne grafove matrica susjedstva je kvadratna matrica koja ima svojstvo da je $a_{ij} = 1$ ako postoji brid između vrhova $i, j \in V(G)$, a inače je $a_{ij} = 0$. Kurikulna mreža je jednostavni usmjereni graf pa znamo da će elementi na glavnoj dijagonali biti jednaki 0. Budući za svaki usmjereni brid $uv \in E(G)$ vrijedi da je $p(u) < p(v)$, matrica susjedstva kurikulnog grafa je gornjetrokutasta kvadratna matrica. Suma jedinica u retku i matrice $\mathbf{A}$ nam daje $d_{out}(i)$, a suma jedinica u stupcu i jednaka je $d_{in}(i)$. Nadalje, analizirali smo standardne mjere centralnosti vrhova kao što su instupanj, outstupanj, međupoloženost, odredili smo usmjernike i autoritete, najmanje udaljenosti, prosječni koeficijent klasteriranja (*clustering coefficient*), dijametar i gustoću. Osnovne statistike svih navedenih mreža mogu se pronaći u Tablici 3.1. Dajemo kratak opis svake od promatranih mjera.

- *Međupoloženost vrha* $u \in V(G)$ računa se po formuli

$$c(u) = \sum_{v \in N(u)} \sum_{\{k,l\} \in V^2} \frac{s_{uv}^{kl}}{s^{kl}} \tag{3.1}$$

  gdje je s_{uv}^{kl} broj najkraćih puteva od k do l koji prolaze bridom uv, a s^{kl} je ukupan broj najkraćih puteva od k do l. Međupoloženost kao mjera centralnosti vrha nam govori koliko neki vrh sudjeluje u komunikaciji ostalih vrhova, tj. koliko najkraćih puteva među parovima povezanih vrhova u grafu prolazi preko njega.

- *Prosječni koeficijent klasteriranja* definiran je kao

$$C = \frac{1}{n} \sum_{i=1}^{n} C_i, \tag{3.2}$$

gdje su C_i lokalni koeficijenti dani sa

$$C_i = \frac{\text{broj parova susjeda od } i \text{ koji su povezani}}{\text{broj parova susjeda od } i} \tag{3.3}$$

Lokalni koeficijent klasteriranja nam kazuje u kojoj mjeri se susjedstvo nekog vrha razlikuje od klike. Prosječni koeficijent klasteriranja je prosjek lokalnih koeficijenata za sve vrhove u mreži koji se često naziva *Watts–Strogatz koeficijent* [162].

- U nekim je usmjerenim mrežama prigodno smatrati važnim onaj vrh koji pokazuje na druge centralne vrhove [116]. Na primjer, postoje brojne mrežne stranice koje sadrže poveznice na druge mrežne stranice koje su važne za neku temu ili područje. Takve stranice same po sebi ne sadrže važne podatke o temi, ali nas usmjeravaju na one stranice koje to sadrže. Iz tog razloga moguće je definirati dvije vrste vrhova: *autoriteti (eng. authority)* su vrhovi koji sadrže važne informacije ili su važni po nekim drugim mjerama centralnosti, a *usmjernici (eng. hubs)* su vrhovi koji nam pokazuju gdje možemo pronaći vrhove koji su autoriteti. Prvi je na ovu ideju došao Kleinberg [88]. U njegovom pristupu, svakom vrhu $i \in V(G)$ dodijeljuje se početna vrijednost za dva tipa centralnosti: *centralnost autoriteta (eng. authority centrality)* x_i i *centralnost usmjernika (eng. hub centrality)* y_i. Centralnost autoriteta za vrh $i \in V(G)$ proporcionalna je sumi centralnosti usmjernika svih vrhova koji na njega pokazuju:

$$x_i = \alpha \sum_j A_{ij} y_j \tag{3.4}$$

pri čemu je α konstanta. Slično, centralnost usmjernika za vrh $i \in V(G)$ proporcionalna je sumi centralnosti autoriteta na koje pokazuje:

$$y_i = \beta \sum_j A_{ji} x_j \tag{3.5}$$

gdje je β druga konstanta. Obično se za početnu vrijednost x_i i y_i uzimaju težine bridova, a ako je riječ o neuteženoj mreži, onda se početna vrijednost postavlja na 1.

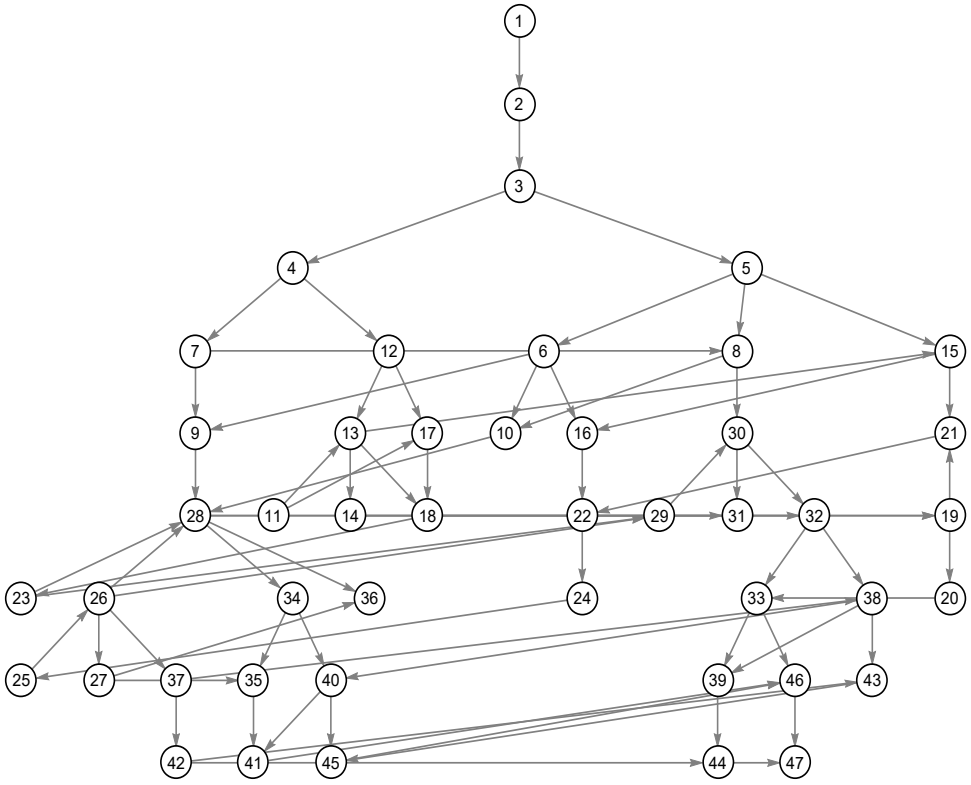
- *Gustoća (eng. density)* je postotak postojećih bridova u odnosu na maksimalni mogući broj bridova. To je podatak koji nam govori do koje mjere se graf "razlikuje" od potpunog grafa, a za jednostavni usmjereni graf s n vrhova i m bridova definirana je kao

$$D = \frac{m}{n(n-1)}.$$

Promotrimo sada neke kurikulne mreže iz različitih područja matematike, fizike, informatike i biologije. Mreže su nazvane po ključnom pojmu čije je usvajanje postavljeno kao cilj učenja za to područje. Sve su slike kurikulnih mreža napravljene koristeći programski alat *Wolfram Mathematica*, a radi preglednosti uklonjeni su tranzitivni bridovi. Nadalje, prilikom računanja centralnosti usmjernika i autoriteta postavljeno je $\beta = 1$, a $\alpha = 1/\lambda$, gdje je λ najveća svojstvena vrijednost matrice $\boldsymbol{AA^T}$, za matricu susjedstva $\mathbf{A}$.

"Skup racionalnih brojeva" (mreža prikazana na Slici 3.6) je kurikulna mreža pojmova s ukupno 47 vrhova pri čemu početni vrh s labelom 1 predstavlja pojam *prirodni broj*, a krajnji vrh s labelom 47 predstavlja pojam *skup racionalnih brojeva*. Vrh s labelom 24 koji predstavlja pojam *razlomak* ima najveću vrijednost međupoloženosti. Centralnost usmjernika je maksimalna (0.1219) za vrh 1 (*prirodni broj*), a slijede redom vrhovi 11 (*nula*), 12 (*negativni broj*) i 17 (*cijeli broj*). Početni vrh ima najveći outstupanj $d_{out}(1) = 26$, a krajnji vrh ima najveći

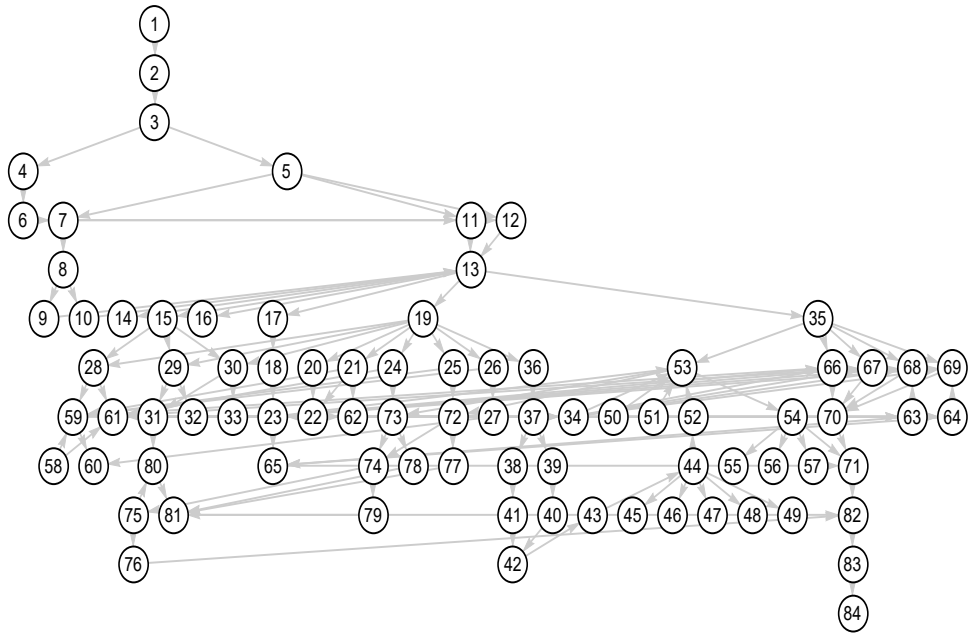
instupanj $d_{in}(47) = 17$. Prosječna najkraća udaljenost među povezanim parovima vrhova je približno jednaka 2 što ukazuje na efekt malog svijeta.



Slika 3.6: Kurikulna mreža pojmova potrebnih za učenje pojma skup racionalnih brojeva. Mreža ima 47 vrhova i 254 usmjerena brida. Vrh s labelom 1 je *prirodni broj*, a vrh s labelom 47 je *skup racionalnih brojeva*.

Mreža ***"Elementarne funkcije"*** ima 84 vrha i 502 usmjerena brida. Vrh s labelom 1 je pojam *skup*, vrh s labelom 82 predstavlja pojam *elementarne funkcije*, a krajnji vrh je pojam *transcedentne funkcije*. Ovo je jedina od promatranih kurikulnih mreža kojoj ključni pojam nije ujedno i pojam s najvećom labelom. Najveći outstupanj ima vrh s labelom 13 koji predstavlja pojam *funkcija*. Isti vrh ima i najveću vrijednost međupoloženosti, a i identificiran je kao usmjernik što

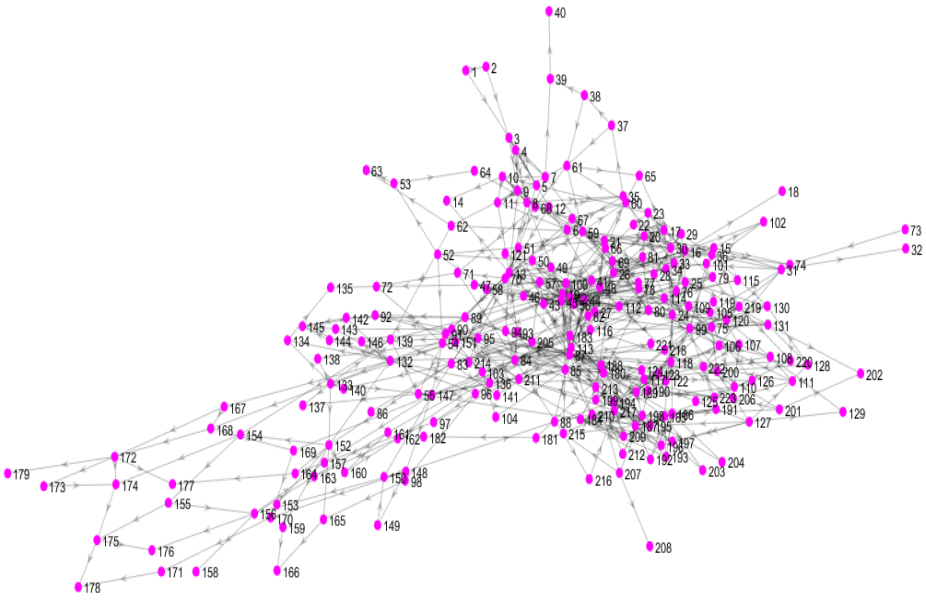
ukazuje na njegovu važnost u ovoj mreži. Pojam *elementarne funkcije* ima najveći instupanj. Prosječna najkraća udaljenost i u ovoj mreži iznosi otprilike 2. Mreža je prikazana na Slici 3.7



Slika 3.7: Kurikulna mreža pojmova potrebnih za učenje pojma elementarne funkcije. Mreža ima 84 vrha i 502 usmjerena brida. Vrh s labelom 1 je *skup*, a vrh s labelom 82 je *elementarne funkcije*.

Mreža ***"Integral"*** ima 223 vrha i 655 usmjerenih bridova (Slika 3.8). Riječ je o mreži pojmova koje je potrebno usvojiti da bi se uspješno položio uvodni kolegij iz matematike čije gradivo završava pojmom integrala. Mreža ima nešto veći dijametar nego prethodne dvije. Prosječna duljina najkraćeg puta je dovoljno mala pa možemo smatrati da je vrijedi efekt malog svijeta. Vrh koji ima najveći instupanj ima labelu 217 (*tok funkcije*), a najveći outstupanj ima vrh s labelom 87 (*realna funkcija realnog argumenta*) koji ujedno ima i najveću vrijednost međupoloženosti. Vrh identificiran kao glavni usmjernik je vrh s labelom 41 (*funkcija*). Ova mreža

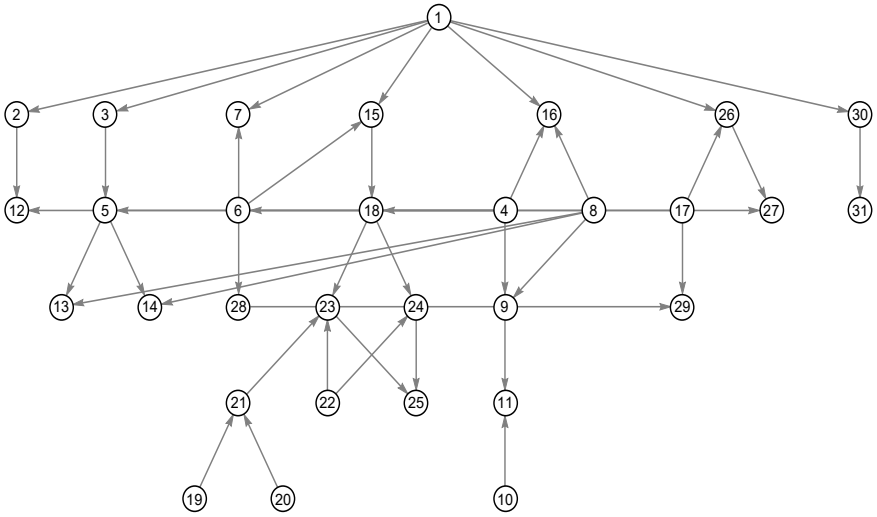
je najveća među promatranim kurikulnim mrežama i ima najmanju gustoću među mrežama čiji vrhovi predstavljaju pojmove.



Slika 3.8: Kurikulna mreža pojmova potrebnih za polaganje uvodnog kolegija iz matematke. Mreža ima 223 vrha i 655 usmjerenih bridova. Vrh s labelom 1 predstavlja pojam *skup*, a vrh s labelom 223 pojam *nepravi integral*. Budući je mreža nešto veća od prethodnih, slika je zbog preglednosti drugačija nego kod prethodnih mreža. Napravljena je u programskom alatu *MATLAB*.

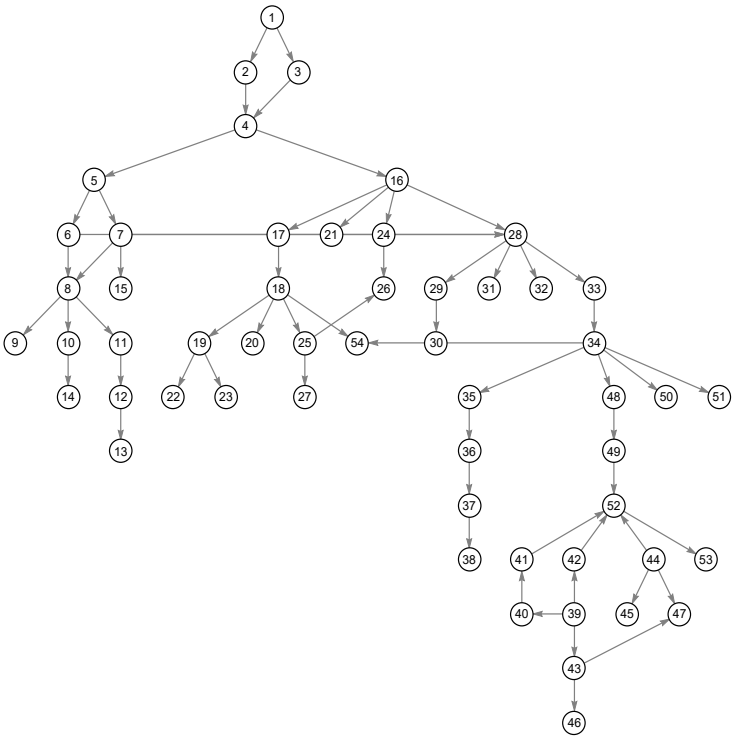
"Fizika" predstavlja kurikulnu mrežu nastavnih tema koje se obrađuju u 7. razredu osnovne škole u sklopu predmeta Fizika. Neke od ovih tema se obrađuju tijekom više nastavnih sati. Mreža ima 31 vrh i 49 usmjerenih bridova (prikazana na Slici 3.9). Vrh s labelom 1 predstavlja temu *duljina*, dok vrh s labelom 31 označava temu *jakost leće*. Budući je riječ o nastavnim temama, a ne konkretno o pojmovima, ova mreža je nešto rjeđa od prethodnih u kojima vrhovi predstavljaju pojmove. Dijametar mreže jednak je 4, dok je prosječna udaljenost dovoljna mala da bi se potvrdio efekt malog svijeta kojeg su pokazale i prethodne analizirane

kurikulne mreže. Maksimalni instupanj imaju vrhovi s labelama 23 (*električna struja*) i 27 (*akceleracija*). Najveći outstupanj ima vrh s labelom 1 (*duljina*) koji je ujedno i najvažniji usmjernik. Vrh s labelom 15 (*rad*) ima najveću vrijednost međupoloženosti.



Slika 3.9: Kurikulna mreža lekcija koje se obrađuju u 7. razredu osnovne škole iz predmeta Fizika. Mreža ima 31 vrh i 49 usmjerenih bridova. Vrh s labelom 1 predstavlja temu *duljina*, a vrh s labelom 31 temu *jakost leće*.

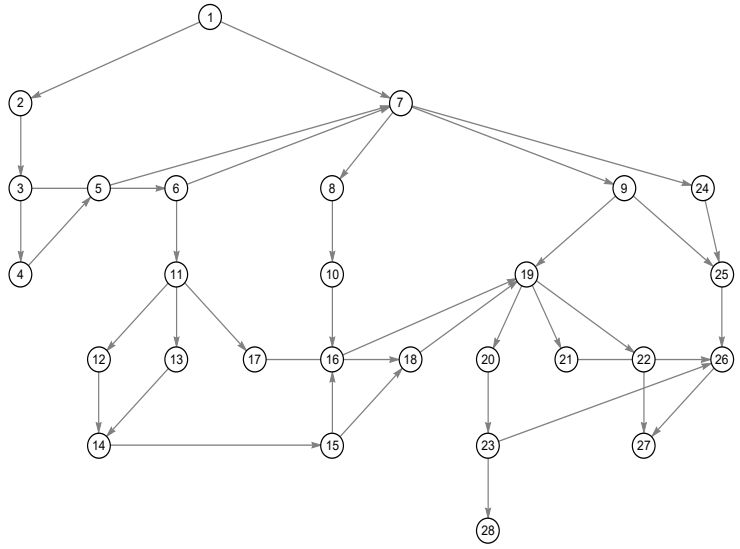
Sljedeća mreža koju analiziramo je kurikulna mreža pojmova koji se uče u sklopu kolegija Programiranje 1 na prvoj godini preddiplomskih studija. Nazvali smo je ***obrada podataka***. Mreža ima 54 vrha počevši od pojma *podatak* i završava pojmom *obrada podataka*. Kao i prethodne, mreža ima mali dijametar i prosječnu duljinu najkraćeg puta pa je još jednom potvrđen efekt malog svijeta. Vrh s najvećim instupnjem predstavlja pojam *obrada podataka*, a s najvećim outstupnjem pojam *podatak*. Maksimalnu vrijednost međupoloženosti ima vrh koji predstavlja *datoteka*. Mreža je prikazana na Slici 3.10.



Slika 3.10: Kurikulna mreža pojmova koji se obrađuju u sklopu kolegija Programiranje na preddiplomskom studiju. Mreža ima 54 vrha i 197 usmjerenih bridova. Vrh s labelom 1 predstavlja pojam *podatak*, a vrh s labelom 54 pojam *obrada podataka*.

Pojam primarne proizvodnje u ekologiji odnosi se na sintezu organskih spojeva iz atmosferskog i vodenog ugljičnog dioksida. Analiziramo kurikulnu mrežu ***"Model primarne proizvodnje"*** s 28 vrhova i 93 usmjerena brida u kojoj početni vrh predstavlja pojam *fotosinteza*, a krajnji vrh pojam *model primarne proizvodnje*, koja objašnjava kako se modelira proces primarne proizvodnje u oceanima. Duljina najkraćeg puta je očekivano mala, kao i ukupna gustoća grafa. Vrh s najvećim instupnjem je vrh s labelom 28 koji predstavlja pojam *primarna proizvodnja* koji je ujedno i glavni autoritet, a vrh s najvećim outstupnjem je vrh koji

predstavlja pojam *svjetlost* koji je ujedno i identificiran kao usmjernik. Najveću vrijednost međupoloženosti ima vrh s labelom 19 koji predstavlja pojam *integral*. Mreža je prikazana na Slici 3.11.



Slika 3.11: Kurikulna mreža pojmova potrebnih za određivanje modela primarne proizvodnje u oceanima. Vrh s labelom 1 predstavlja pojam *fotosinteza*, a vrh s labelom 28 pojam *primarna proizvodnja*.

Konačno, kod navedenih kurikulnih mreža možemo uočiti neka zajednička svojstva. U testiranim mrežama pojavljuju se dvije vrste struktura. Jedna je struktura ona u kojoj najveći instupanj ima vrh koji predstavlja ključni pojam, a najveći outstupanj vrh s labelom 1 koji predstavlja početni pojam, kao što je slučaj s mrežama "Skup racionalnih brojeva" i "Obrada podataka". Druga struktura koja se pojavljuje je ona u kojoj je moguće identificirati neki važan pojam s najvećim instupnjem ili outstupnjem, pomoću kojeg se grade ostali pojmovi, kao što je slučaj s pojmom *funkcija* u mreži "Elementarne funkcije". Obično taj vrh ima i najveću vrijednost ostalih mjera centralnosti. Sve testirane mreže imaju relativno mali

dijametar i prosječne najkraće udaljenosti što potvrđuje efekt malog svijeta. Sve su mreže relativno rijetke, što nas navodi na zaključak da osobe koje ih sastavljaju češće biraju grafove tipa A. U svim testiranim mrežama lako je identificirati ključni pojam odnosno vrh koji ima maksimalnu vrijednost međupoloženosti koja drastično odskače od vrijednosti za ostale vrhove u mreži, kao npr. *razlomak* u mreži "Skup racionalnih brojeva" ili *funkcija* u mreži "Elementarne funkcije". Obično se taj pojam nalazi negdje po sredini obrađenog gradiva i očito najbolje povezuje pojmove naučene prije i nakon njega. Sve promatrane mreže imaju relativno nizak prosječni koeficijent klasteriranja. Razlog tome je što ova mjera više značaja pridaje vrhovima s niskim stupnjem budući je za takve vrhove vrijednost nazivnika u jednadžbi 3.3 relativno mala. U kurikulnim mrežama je velik broj vrhova s relativno malim stupnjem što se odražava na konačnu vrijednost ove mjere.

Tablica 3.1: **Osnovne statistike za neke kurikulne mreže.** Prikazane su redom sljedeće opcije: broj vrhova n, broj bridova m, najveći instupanj d_{in}, najveći outstupanj d_{out}, prosječni stupanj vrha d_{avg}, prosječna duljina najkraćeg puta l za parove povezanih vrhova, koeficijent klasteriranja C, dijametar $diam$, gustoća grafa D, maksimalna vrijednost međupoloženosti c, maksimalna vrijednost centralnosti usmjernika h, maksimalna vrijednost centralnosti autoriteta a

Mreža	**n**	**m**	$\boldsymbol{d_{in}}$	$\boldsymbol{d_{out}}$	$\boldsymbol{d_{avg}}$	**l**	**C**	**diam**	**D**	**c**	**h**	**a**
Skup $\mathbb{Q}$	47	254	17	26	5.404	2.011	0.254	5	0.117	117.940	0.1219	0.0461
Elementarne funkcije	84	502	27	51	5.976	2.132	0.255	6	0.072	309.429	0.1125	0.0313
Integral	223	655	15	28	2.941	3.899	0.084	10	0.013	2041.796	0.0881	0.0489
Fizika	31	49	4	8	1.581	1.575	0.049	4	0.053	12.000	0.2402	0.1066
Model pr. proizvodnje	28	93	9	14	3.321	2.135	0.183	5	0.123	66.089	0.1764	0.0885
Obrada podataka	54	197	12	22	3.648	1.744	0.338	5	0.069	79.000	0.1463	0.0682

3.5 Optimalna ekspozicija jedinica

U ovom ćemo poglavlju izložiti algoritam koji daje odgovarajući poredak edukacijskih jedinica s minimalnom vrijednošću odabrane mjere složenosti. Obično se redoslijed edukacijskih jedinica, kao i cjelokupni plan i program, određuje od strane Ministarstva znanosti, obrazovanja i sporta, Agencije za odgoj i obrazovanje i ostalih mjerodavnih institucija, a uvelike se temelji na iskustvu i stručnosti profesora i ekspertne skupine. Predložen je algoritam koji vrednuje kvalitetu poretka edukacijskih jedinica pomoću jedne od mjera složenosti definiranih u prethodnom odlomku. Cilj je dobiti poredak edukacijskih jedinica koji ima najmanju složenost, tj. poredak za koji je vrijednost odabrane mjere složenosti minimalna.

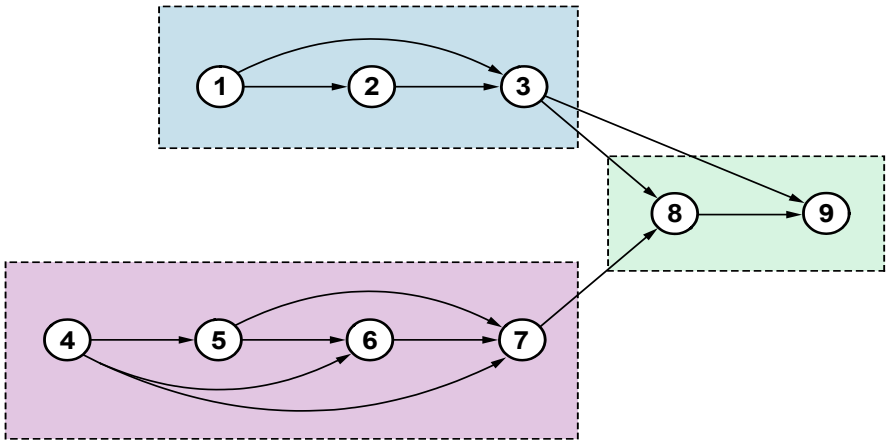
Kao što je već navedeno, problem određivanja optimalne kurikulne ekspozicije može se povezati s problemom topološkog sortiranja unutar područja računalnih znanosti. Za rješavanje tog problema poznata su neka rješenja u obliku Kahnovog algoritma (Algoritam 1) i tzv. *depth–first* algoritma. Međutim, nijedno od tih rješenja ne odgovara u potpunosti danom problemu. Kahnov algoritm kao rješenje daje jednu kurikulnu ekspoziciju. Da bi se rješio dani problem, trebalo bi izgenerirati sve moguće valjane poretke i usporediti vrijednosti izabrane mjere za svaki od njih, što je, ovisno o mreži, jako težak zadatak. Alternativno, mogli bismo nekako ograničiti broj generiranih poredaka pa među njima odrediti onaj s najmanjom vrijednosti odabrane mjere, ali ostaje pitanje na koji način ograničiti prostor mogućih rješenja da se što manje utječe na valjanost rezultata. Algoritam *depth–first* prolazi kroz svaki vrh $u \in V(G)$ u grafu G, u proizvoljnom redoslijedu, i razmatra sve vrhove na koje u pokazuje [45, 150]. U trenutku kada se vrh u doda u izlaznu listu, tj. u sortirani poredak, svi vrhovi koji su pokazivali na u već su razmotreni i uključeni u poredak. Pošto se svaki vrh i brid posjećuju jednom, algoritam radi u linearnom vremenu. Pseudokod je dan u Algoritmu 2.

Algorithm 2 *depth–first* algoritam za topološko sortiranje

Output: Lista L koja će sadržavati sortirane elemente

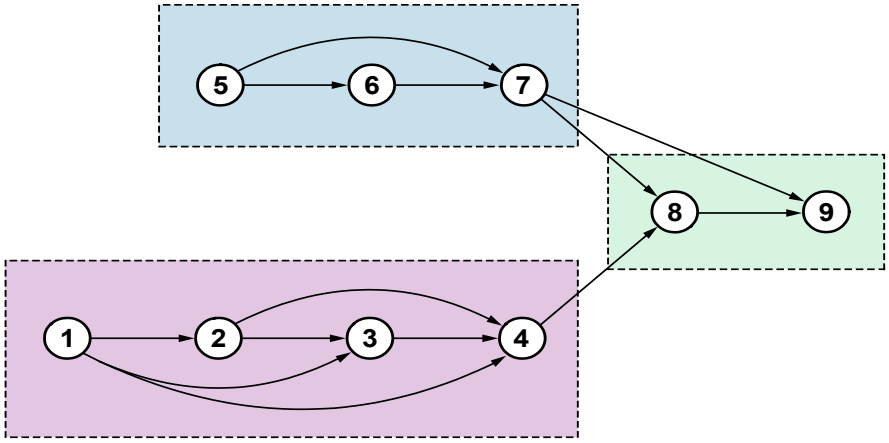
while postoje neoznačeni vrhovi **do**
  odaberi neoznačeni vrh u
  posjeti (u)
end while
Funkcija posjeti (vrh u)
if u ima privremenu oznaku **then**
  stop (postoji ciklus)
end if
if u još nije označen (posjećen) **then**
  privremeno označi u
  for all vrh v takav da postoji brid $e = uv$ **do**
    posjeti (v)
  end for
  vrhu u dodaj trajnu oznaku
  vrhu u ukloni privremenu oznaku
  dodaj vrh u na početak liste L
end if

Međutim, ovakav algoritam će imati problema s mrežom prikazanom na Slici 3.12. kakva se često pojavljuje tijekom sastavljanja nastavnog plana budući se edukacijske jedinice često grupiraju u veće cjeline. Mrežu možemo podijeliti u 3 bloka prikazana pravokutnicima. Prvi blok čine jedinice $1, 2$ i 3, drugi blok jedinice $4, 5, 6$ i 7, a treći jedinice 8 i 9.



Slika 3.12: Primjer kurikulne mreže za koju algoritam *depth–first* ne nalazi optimalnu ekspoziciju. Edukacijske jedinice možemo grupirati u skupine prikazane pravokutnicima. Vrijedi $s_1(G) = 3$.

Vrijednost mjere $s_1(G)$ bit će minimalna ako se izmjeni redoslijed blokova edukacijskih jedinica na način prikazan na Slici 3.13.



Slika 3.13: Optimalna kurikulna ekspozicija mreže s prethodne slike. Vrijednost mjere za ovakvu ekpoziciju je $s_1(G) = 2.444$.

Budući *depth–first* algoritam promatra vrhove pojedinačno, teško će biti u mogućnosti dati poredak koji rezultira najmanjom vrijednošću odabrane mjere za ovaj primjer. Stoga predlažemo heuristički algoritam temeljen na renumeraciji labela koji ne promatra nužno pojedinačne vrhove nego i blokove povezanih vrhova. Predloženi algoritam moguće je razložiti na nekoliko dijelova:

1. *svakom vrhu u mreži dodijeljuje se jedinstvena numerička labela na način da je labela početnog vrha manja od labele krajnjeg vrha za svaki usmjereni brid u mreži,*

2. *računa se inicijalna vrijednost odabrane mjere složenosti i postavlja kao privremeno rješenje,*

3. *radi se renumeracija labela vrhova na način opisan u daljnjem tekstu,*

4. *ako skup labela dobiven novom renumeracijom nije parcijalno uređen ili vrijednost zadane mjere takvog poretka nije manja od prethodne, poredak se odbacuje*

5. *ako je skup labela dobiven novom renumeracijom parcijalno uređen i vrijednost odabrane mjere je manja od privremene, takav se poredak edukacijskih jedinica sprema kao potencijalno rješenje*

6. *postupak se ponavlja sve dok nije moguće pronaći rješenje koje je bolje od rješenja iz prethodne iteracije*

Početnu numeraciju moguće je dobiti Kahnovim algoritmom opisanim u Algoritmu 1. Za primjer mjere složenosti odabrana je mjera $s_p^1(G)$ definirana:

$$s_p^1(G) = \frac{\sum\limits_{v \in V(G)} s_{p,G}(v)}{card\,(V(G))},$$

pri čemu je

$$s_{p,G}(v) = \sum_{uv \in E(G)} [p(v) - p(u)] .$$

Renumeracija labela odrađena je na sljedeći način. Labele vrhova poredane su u niz duljine $n = card(V(G))$. Zatim su rađene uzastopne izmjene blokova niza različitih duljina da bi se pokrilo što više različitih kombinacija. Svaka zamjena dvaju blokova rezultira novom numeracijom labela grafa. Nakon svake renumeracije vrši se provjera zadovoljava li nova renumeracija oba osnovna uvjeta:

1. je li poštovan uređaj, tj. je li za svaki usmjerni brid labela početnog vrha manja od labele krajnjeg vrha

2. je li vrijednost mjere $s_p^1(G')$ za novi graf G' dobiven renumeracijom labela manja ili jednaka od prethodne vrijednosti mjere $s_p^1(G)$

Ako jedan od uvjeta nije ispunjen, takav se poredak odbacuje. Ako su oba uvjeta ispunjena, graf G' s novom numeracijom labela se zapisuje kao privremeno najbolje rješenje i algoritam radi sljedeću izmjenu. Kad su promjenjeni svi blokovi, program se pokreće ponovo, ali sada kao početni graf, odnosno inicijalnu numeraciju, algoritam uzima najbolje rješenje iz prethodne iteracije. Postupak se ponavlja sve dok algoritam više nije u mogućnosti pronaći numeraciju labela, odnosno graf G za koji je vrijednost mjere $s_p^1(G)$ manja od privremene. Važno je napomenuti da se algoritmom ne mijenja odnos edukacijskih jedinica, samo redoslijed njihovog učenja. Pseudokod opisanog algoritma za optimalnu ekspoziciju edukacijskih jedinica moguće je pronaći u Algoritmu 3.

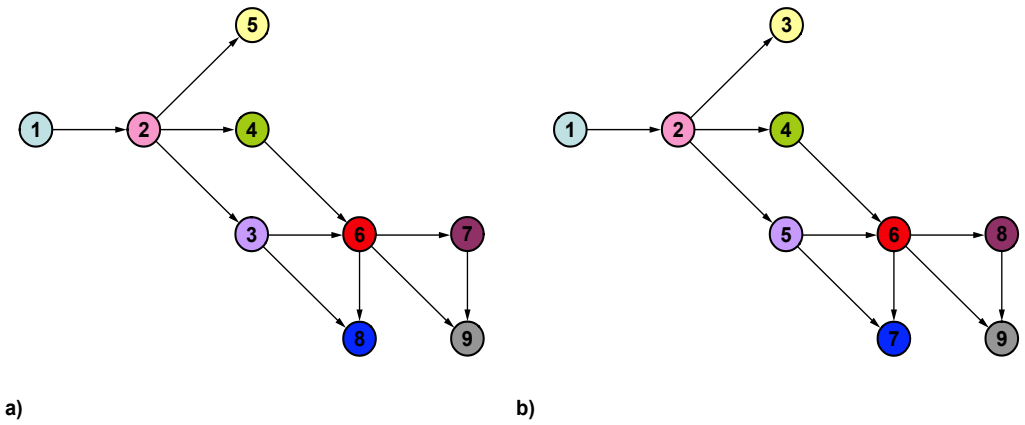
Algorithm 3 Algoritam za optimalnu ekspoziciju edukacijskih jedinica

Input: Lista bridova

svakom vrhu $i \in V(G)$ dodijeli jedinstvenu labelu $l_i = p(i)$
inicijalno pridruživanje labela proglasi privremenim rješenjem
repeat
    napravi renumeraciju labela
    if nova numeracija ne poštuje valjani uređaj labela **then**
        odbaci
    else
        izračunaj vrijednost odabrane mjere za novu numeraciju labela
        if nova vrijednost > privremene vrijednosti **then**
            postavi novu vrijednost za privremenu
        else
            odbaci poredak i idi na sljedeću numeraciju
        end if
    end if
until nije moguće pronaći numeraciju koja daje bolju vrijednost mjere od prethodne

Algoritam je implementiran u programskom alatu *Microsoft Visual Studio 2015* (moguće ga je pronaći u Prilozima). Prvo je testiran na manjim mrežama za koje pronalazi egzaktna rješenja. Na Slici 3.14 može se vidjeti jednostavan primjer kurikulne mreže s 9 vrhova i 11 usmjerenih bridova za koju algoritam pronalazi egzaktno rješenje, točnije kurikulnu mrežu s minimalnom vrijednošću mjere $s^1(G)$. Algoritam je zatim testiran na kurikulnim mrežama navedenim u prethodnom poglavlju. Za svaku od njih napravljen je tranzitivni zatvarač, tj. mrežu smo iz tipa A pretvorili u tip B (za svaka dva vrha $u, v \in V(G)$ takve da postoji usmjereni put od u do v dodan je usmjereni brid $uv \in E(G)$) , što je rezultiralo

promjenom broja usmjerenih bridova u odnosu na originalnu mrežu. Kao što je vidljivo iz Tablica 3.2 i 3.3, algoritam za svaku od navedenih kurikulnih mreža daje poboljšanje u odnosu na mjeru $s_1(G)$.



Slika 3.14: Kurikulna mreža s $n = 9$ vrhova i $m = 11$ usmjerenih bridova. Svaka edukacijska jedinica označena je jednom bojom. Nakon izvršavanja algoritma ne mijenja se odnos jedinica, nego samo njihov redoslijed. a) Početna mreža korištena kao ulazni podatak za koju je $s^1(G) = 2.77$ b) Algoritam daje egzaktno rješenje $s^1(G) = 2.11$

Što se tiče računalne složenosti predloženog algoritma, da bi se odradila renumeracija labela potrebno je $O(n^3)$. Nakon renumeracije potrebno je zamijeniti labelu svim vrhovima i ažurirati listu bridova za što je potrebno $O(n+m)$. Kako su sve testirane kurikulne mreže rijetke vrijedi da je $n \sim m$, pa je ukupna računalna složenost algoritma $O(n^3 m)$ za mrežu s n vrhova i m bridova. Kurikulne mreže imaju do nekoliko stotina vrhova i dosta su rijetke, pa je za naše potrebe vrijeme izvršavanja algoritma i više nego zadovoljavajuće.

Tablica 3.2: **Algoritam za optimalnu ekspoziciju edukacijskih jedinica testiran na kurikularnim mrežama tipa** A. *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, u stupcu "Stručnjak" nalaze se vrijednosti mjere $s_1(G)$ izračunate za ulaznu mrežu koju su sastavili stručnjaci, u stupcu "Algoritam" nalaze se vrijednosti mjere $s_1(G)$ nakon provedenog algoritma, tj. vrijednost koja se dobije za poredak edukacijskih jedinica dobiven pomoću opisanog algoritma

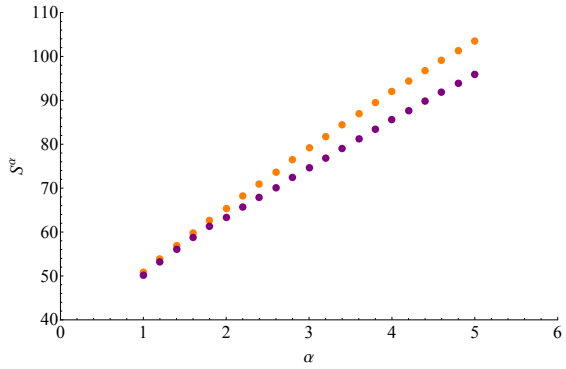
			Mjera složenosti $s_1(G)$	
Tip A	**n**	**m**	**Stručnjak**	**Algoritam**
Skup $\mathbb{Q}$	47	254	50.894	50.170
El. funkcije	84	502	143.167	112.833
Integral	223	655	93.399	72.413
Fizika	31	49	11.871	7.903
Model prim. proizvodnje	28	93	26.071	22.179
Obrada podataka	54	197	47.778	34.315

Promotrimo kurikulnu mrežu *Integral*. Pripadajući usmjereni graf ima 223 vrha i 656 usmjerenih bridova i gustoću 0.013 što je dosta rijedak (sparse) graf. Tranzitivni zatvarač, odnosno pripadni graf tipa B, ima 223 vrha i 10952 usmjerena brida. Algoritam daje znatno poboljšanje ulaznog grafa. Ovaj primjer je odličan pokazatelj koliko redoslijed i organizacija nastavnog procesa ovisi o iskustvu i pristupu gradivu. Grafovi tipa A obuhvaćaju pristup u kojem nije potrebno vraćati se na već usvojeno gradivo jer se u procesu učenja novog gradiva automatski osvježilo i prijašnje. Grafovi tipa B temelje se na stajalištu da je prilikom usvajanja nove jedinice potrebno ponoviti prethodno stečeno znanje. Naravno da nije moguće prilikom usvajanja svakog novog pojma ponavljati sve druge pojmove koji su pret-

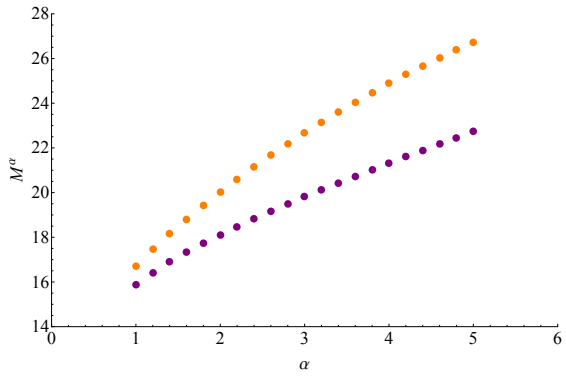
Tablica 3.3: **Algoritam za optimalnu ekspoziciju edukacijskih jedinica testiran na tranzitivnim zatvaračima kurikularnih mreža (na mrežama tipa B).** *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, u stupcu "Stručnjak" nalaze se vrijednosti mjere $s_1(G)$ izračunate za ulaznu mrežu koju su sastavili stručnjaci, u stupcu "Algoritam" nalaze se vrijednosti mjere $s_1(G)$ nakon provedenog algoritma, tj. vrijednost koja se dobije za poredak edukacijskih jedinica dobiven pomoću opisanog algoritma

			Mjera složenosti $s_1(G)$	
Tip B	**n**	**m**	**Stručnjak**	**Algoritam**
Skup $\mathbb{Q}$	47	815	316.915	312.340
El. funkcije	84	1941	803.417	705.988
Integral	223	10952	4753.785	4383.435
Fizika	31	87	27.484	22.129
Model prim. proizvodnje	28	281	107.821	104.821
Obrada podataka	54	383	134.30	112.167

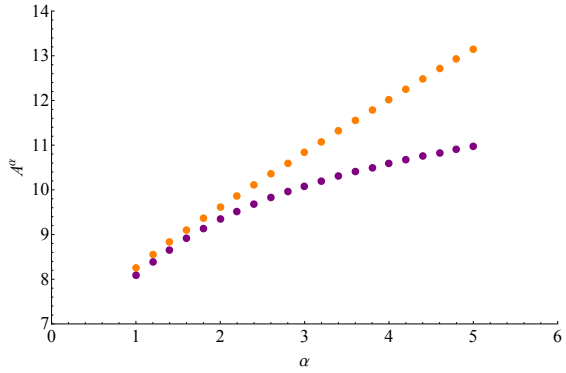
hodno naučeni, ali pošto je u mreži moguće identificirati važne vrhove možda bi bilo dobro povremeno ponoviti takve pojmove prije obrađivanja novog gradiva. Iz ovog primjera da se zaključiti kako tranzitivno zatvorene mreže ostavljaju prostora za daljnja poboljšanja u procesu sekvenciranja. Budući proces sastavljanja kurikula i određivanja ekspozicije edukacijskih jedinica ovisi o odabranoj mjeri, Algoritam za određivanje optimalne ekspozicije testirali smo na kurikulnim mrežama navedenim u Poglavlju 3.4 koristeći mjere $s^{\alpha}(G)$, $m^{\alpha}(G)$ i $a^{\alpha}(G)$ za različite vrijednosti parametra α. Za svaku od navedenih mjera algoritam daje poboljšanje u odnosu na ulaznu mrežu. Rezultate su prikazani na Slikama 3.15 do 3.19.



(a) Mjera složenosti $s^\alpha(G)$

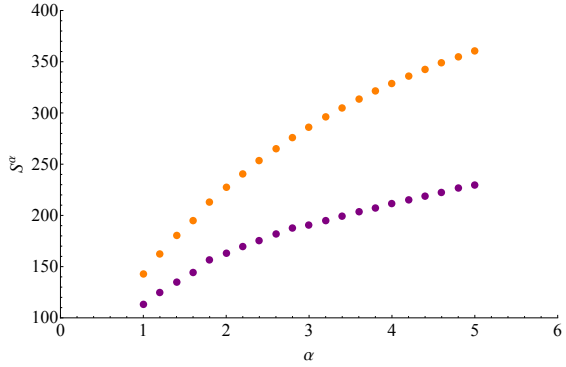


(b) Mjera složenosti $m^\alpha(G)$

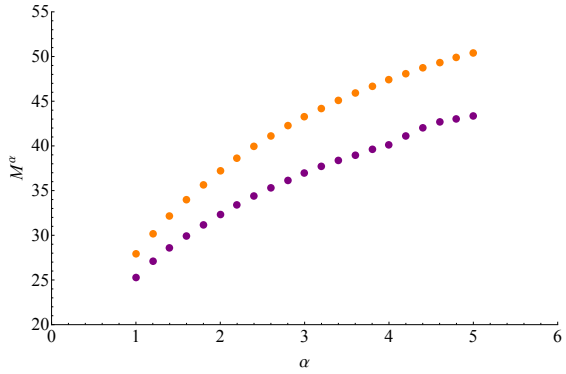


(c) Mjera složenosti $a^\alpha(G)$

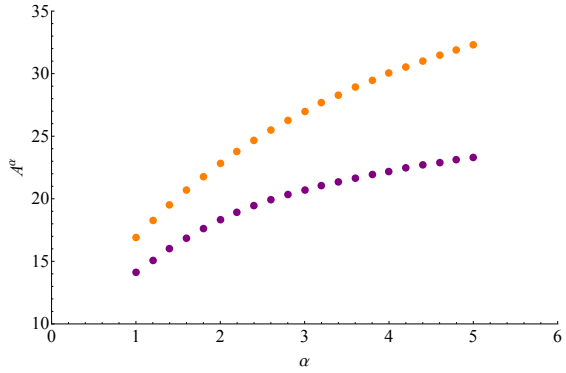
Slika 3.15: Usporedba rezultata različitih mjera složenosti dobivenih pomoću Algoritma za optimalnu ekspoziciju edukacijskih jedinica primijenjenog na kurikulnu mrežu *Skup* $\mathbb{Q}$. Narančastom bojom su prikazani rezultati odabrane mjere za mrežu sastavljenu od strane stručnjaka, ljubičastom bojom su prikazani rezultati dobiveni upotrebom algoritma. Svaka točka na grafu predstavlja vrijednost mjere dobivenu za različite vrijednosti parametra α.



(a) Mjera složenosti $s^\alpha(G)$

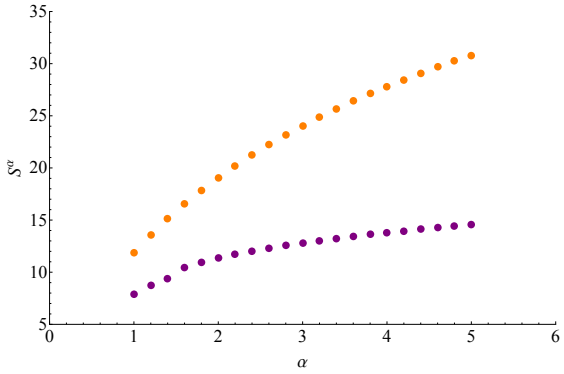


(b) Mjera složenosti $m^\alpha(G)$

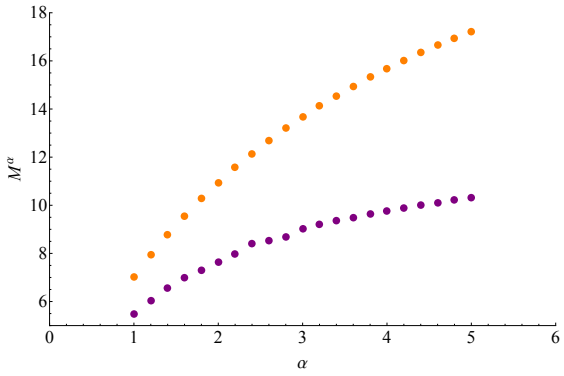


(c) Mjera složenosti $a^\alpha(G)$

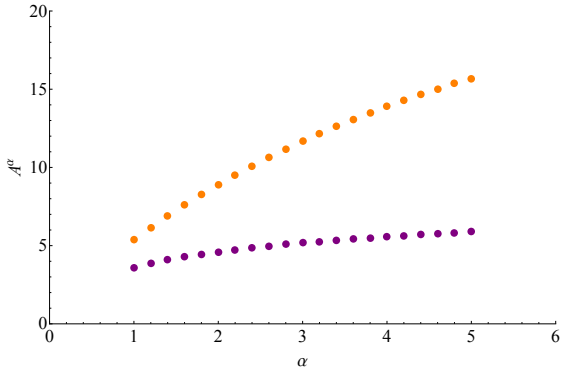
Slika 3.16: Usporedba rezultata različitih mjera složenosti dobivenih pomoću Algoritma za optimalnu ekspoziciju edukacijskih jedinica primijenjenog na kurikulnu mrežu *Elementarne funkcije*. Narančastom bojom su prikazani rezultati odabrane mjere za mrežu sastavljenu od strane stručnjaka, ljubičastom bojom su prikazani rezultati dobiveni upotrebom algoritma. Svaka točka na grafu predstavlja vrijednost mjere dobivenu za različite vrijednosti parametra α.



(a) Mjera složenosti $s^\alpha(G)$

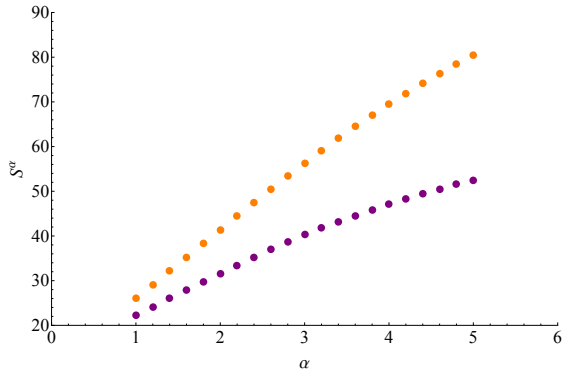


(b) Mjera složenosti $m^\alpha(G)$

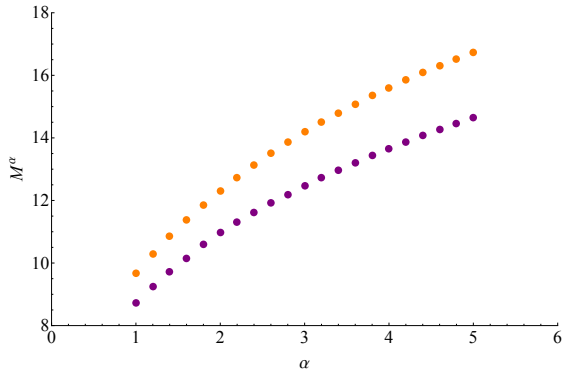


(c) Mjera složenosti $a^\alpha(G)$

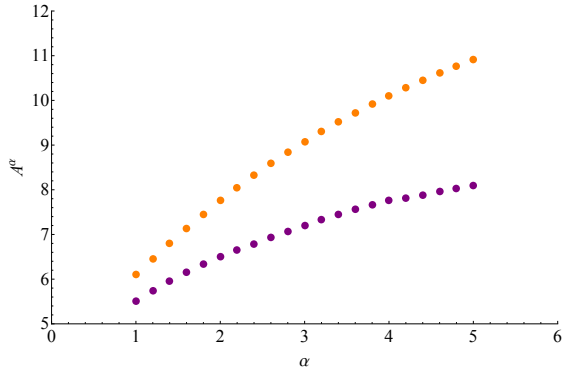
Slika 3.17: Usporedba rezultata različitih mjera složenosti dobivenih pomoću Algoritma za optimalnu ekspoziciju edukacijskih jedinica primijenjenog na kurikulnu mrežu *Fizika*. Narančastom bojom su prikazani rezultati odabrane mjere za mrežu sastavljenu od strane stručnjaka, ljubičastom bojom su prikazani rezultati dobiveni upotrebom algoritma. Svaka točka na grafu predstavlja vrijednost mjere dobivenu za različite vrijednosti parametra α.



(a) Mjera složenosti $s^\alpha(G)$

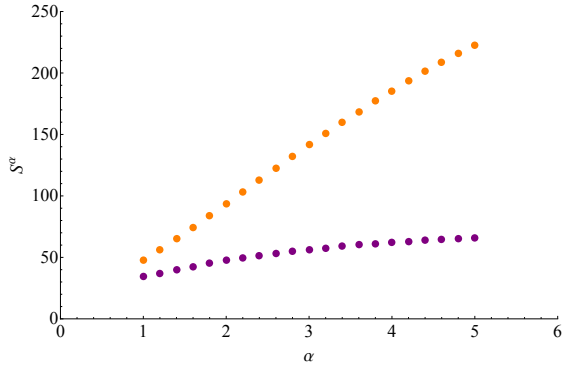


(b) Mjera složenosti $m^\alpha(G)$

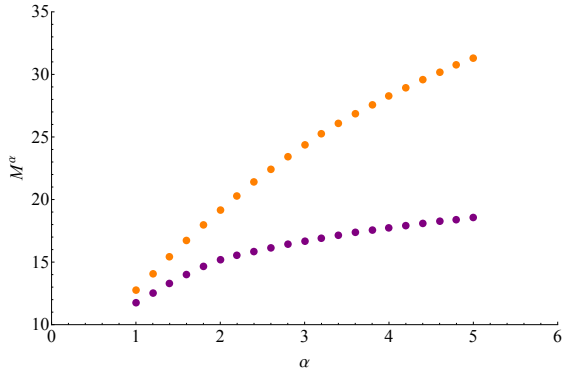


(c) Mjera složenosti $a^\alpha(G)$

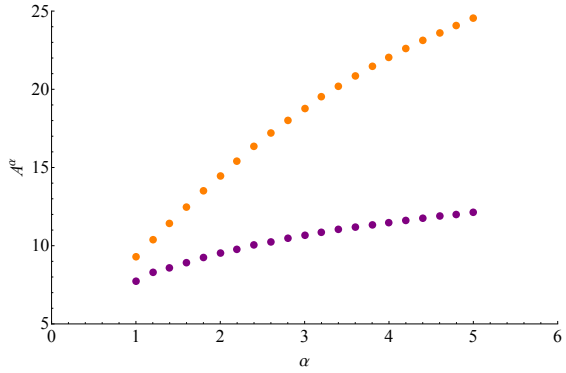
Slika 3.18: Usporedba rezultata različitih mjera složenosti dobivenih pomoću Algoritma za optimalnu ekspoziciju edukacijskih jedinica primijenjenog na kurikulnu mrežu *Model primarne proizvodnje*. Narančastom bojom su prikazani rezultati odabrane mjere za mrežu sastavljenu od strane stručnjaka, ljubičastom bojom su prikazani rezultati dobiveni upotrebom algoritma. Svaka točka na grafu predstavlja vrijednost mjere dobivenu za različite vrijednosti parametra α.



(a) Mjera složenosti $s^\alpha(G)$



(b) Mjera složenosti $m^\alpha(G)$



(c) Mjera složenosti $a^\alpha(G)$

Slika 3.19: Usporedba rezultata različitih mjera složenosti dobivenih pomoću Algoritma za optimalnu ekspoziciju edukacijskih jedinica primijenjenog na kurikulnu mrežu *Obrada podataka*. Narančastom bojom su prikazani rezultati odabrane mjere za mrežu sastavljenu od strane stručnjaka, ljubičastom bojom su prikazani rezultati dobiveni upotrebom algoritma. Svaka točka na grafu predstavlja vrijednost mjere dobivenu za različite vrijednosti parametra α.

Poglavlje 4

Detektiranje zajednica u kurikulnim mrežama

Ponekad je podjelom nekog kompleksnog sustava u manje dijelove moguće steći bolji uvid u organizaciju sustava i njegovo funkcioniranje. Stoga je detektiranje zajednica u mrežama jedan od fundamentalnih problema u teoriji kompleksnih mreža [53, 91, 117].

Problem detektiranja zajednica odnosi se pronalaženje prirodne podjele mreže na grupe vrhova takve da postoji mnogo bridova unutar zajednice, a nekoliko (manje) bridova među grupama [116]. Intuitivno, zajednica je kohezijska skupina vrhova koji su povezani "gušće" jedni s drugima nego s vrhovima u drugim zajednicama. Jedan od aspekata koji čini problem detektiranja zajednica zahtjevnim je to što točna formulacija pojma "zajednica" često ovisi o domeni iz koje se promatra. Na primjer, u društvenim mrežama zajednica može odgovarati grupi prijatelja koji pohađaju istu školu ili koji studiraju na istom sveučilištu [108]; u mreži proteina zajednice mogu predstavljati funkcionalne module proteina koji su u međusobnoj interakciji [3]; u mrežama suradnje i koautorstva, istoj zajednici mogu pripadati autori koji pripadaju istoj znanstvenoj disciplini [65] i slično.

Zajednice se često definiraju u smislu podjele vrhova na način da svaki vrh pripada u točno jednu zajednicu. To je korisno pojednostavljenje strukture mreže i većina metoda za detektiranje zajednica pronaći će upravo ovu vrstu podjele. Međutim, u nekim slučajevima bolji prikaz strukture mreže može biti onaj u kojem vrhovi pripadaju u više od jedne zajednice. To se često događa u društvenim mrežama, gdje vrhovi predstavljaju osobe, a zajednice predstavljaju različite vrste pripadnosti: jedna zajednica za obitelj, druga zajednica za suradnike, treća za prijatelje i tako dalje [124]. Identificiranje zajednica omogućuje i svojevrsnu klasifikaciju vrhova ovisno o njihovoj poziciji unutar zajednice i općenito u grafu. Vrhovi koji su na centralnim pozicijama u zajednici, tj. imaju puno susjeda unutar iste zajednice, zasigurno imaju važnu ulogu u kontroli i stabilnosti zajednice. Vrhovi koji se nalaze na putu između dviju ili više zajednica imaju važnu ulogu u posredovanju i kontroliraju komunikaciju među zajednicama što je posebno istaknuto u društvenim i metaboličkim mrežama.

Budući različite grane znanosti imaju različite potrebe, razvijen je širok spektar algoritama za detekciju zajednica koji udovoljavaju tim potrebama. Razvijene su brojne metode koje se temelje na alatima i tehnikama iz disciplina kao što su fizika, biologija, primjenjena matematika, računalne i društvene znanosti. Uspješnost i efikasnost ovakvih algoritma uvelike ovisi o tome kako se definira zajednica. Obzirom da je detektiranje zajednica u mrežama interdisciplinarni i zadnjih desetljeća aktivno istraživani problem, kao rezultat brojnih istraživanja razvijen je skup raznih realnih i računalno generiranih mreža koje se koriste za testiranje razvijenih algoritama [92].

Otkrivanje zajednica u usmjerenim mrežama je mnogo složeniji problem nego je to slučaj u neusmjerenim mrežama. U kurikulnim mrežama problem je dodatno otežan zahtjevom da dobivene zajednice moraju zadovoljavati određene uvjete. Organizacija nastavnih sadržaja i materijala po područjima, kolegijima, programima ili obrazovnim razinama često nije linearan proces i ne podrazumijeva samo podjelu određenog broja materijala u više nastavnih sati. Riječ je o procesu koji podliježe čitavom nizu kriterija. Jedan od mogućih kriterija je zahtjev da organizacija nastavnih materijala poštuje načelo odgovarajućeg sekvenciranja, na primjer, linearna progresija u kojoj se prvo obrađuju prirodni brojevi, zatim cijeli, racionalni i konačno realni brojevi. Moguće je raditi podjelu materijala i sadržaja na temelju njihove prirode, na primjer, podjelu po područjima gdje se prvo obrađuje algebra, zatim geometrija i slično. Može se očekivati da podjela materijala prati i neka druga načela, primjerice da se materijali grupiraju u odnosu na određeni cilj učenja koji treba postići i slično [174].

Problem detektiranja zajednica u kurikulnim mrežama možemo definirati kao problem grupiranja edukacijskih jedinica u veće cjeline koje se mogu proučavati uzastopno, na primjer, grupiranje nastavnih tema u nastavne cjeline koje se obrađuju u nizu, jedna za drugom. Drugim riječima, ako je dan parcijalni uređaj edukacijskih jedinica $x_1 \prec x_2 \prec ... \prec x_n$ (znak "$\prec$" koristimo u smislu da se edukacijska jedinica x_i obrađuje i uči prije edukacijske jedinice x_j) želimo podijeliti kurikulnu mrežu na zajednice $A_1, A_2, ..., A_k$ koje možemo urediti tako da vrijedi:

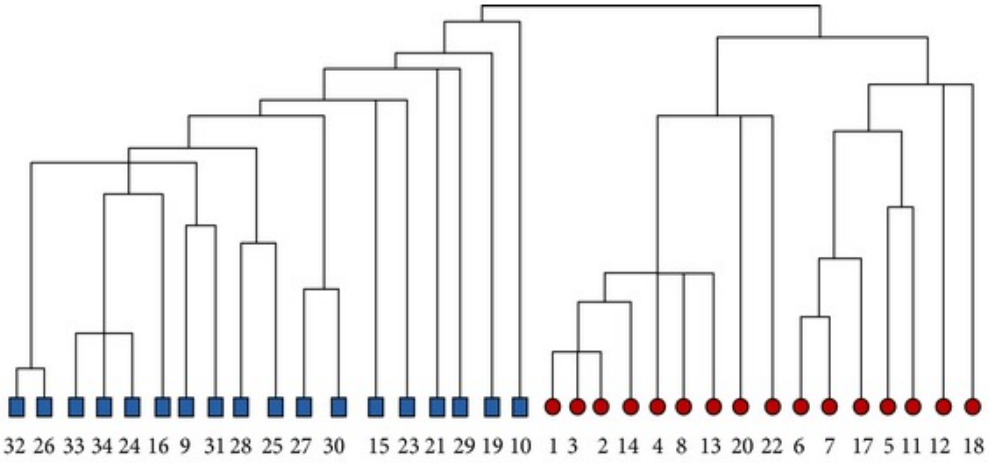
ako je $x_i \prec x_j$, $x_i \in A_p$ i $x_j \in A_q$ onda je $A_p \prec A_q$ ili $A_p = A_q$.

U ovom radu koncentrirali smo se detektiranje nepreklapajućih zajednica u kurikulnim mrežama po načelu sekvenciranja.

4.1 Detektiranje zajednica u neusmjerenim mrežama

U cilju otkrivanja svojstava i strukture kompleksnih mreža razvijeni su brojni algoritmi za detektiranje zajednica, no kvaliteta pojedinog algoritma, u smislu točnosti i računalne složenosti, ovisi o mnogo faktora [170]. Pošto se većina ideja i metoda na neki način prenosi i modificira za usmjerene mreže, u ovom poglavlju navodimo neke algoritme za detektiranje zajednica u neusmjerenim mrežama. Teško je kategorizirati sve postojeće algoritme, ali okvirno ih je moguće podijeliti na hijerarhijske, optimizirajuće i ostale [61]. Rani radovi temelje se na hijerarhijskom pristupu spajanja ili razdvajanja zajednica odabranih u skladu s nekom mjerom sličnosti ili funkcijom udaljenosti. Aglomerativni pristup započinje s jednakim brojem zajednica i vrhova, tako da svaki vrh pripada svojoj zajednici, i iterativno spaja zajednice sve dok ne ostane samo jedna [16]. Suprotno tome, divizivni pristup započinje s jednom zajednicom koja sadrži sve vrhove i iterativno je dijeli na manje zajednice sve dok svaki vrh ne postane svoja vlastita zajednica [132]. Algoritami u ovoj familiji se međusobno razlikuju po vrsti odabrane mjere ili funkcije. Kao rezultat spajanja ili razdvajanja zajednica dobije se stablo zajednica koje nazivamo dendrogram [117] (primjer prikazan na Slici 4.1) no i dalje je potrebno odrediti gdje zaustaviti proces da bi se dobila stvarna particija, što, naravno, ovisi o potrebama istraživača. Metode koje se temelje na optimizaciji koriste mjeru pomoću koje procijenjuju kvalitetu podjele mreže. Mjera koja se najčešće koristi je Newmanova modularnost [120]. Algoritmi se obično sastoje od procesuiranja nekoliko particija mreže (odabranih slučajno ili u skladu s nekom funkcijom), a zatim zadržavanja one particije koja je najbolja u odnosu na odabranu mjeru (na primjer, podjela koja ima najveću vrijednost modularnosti). U posljednju familiju možemo ubrojiti sve ostale pristupe. Neki koriste principe poput podjele temeljene na gustoći (eng. *density-based clustering*) [58], neki su temeljeni na agentima [98], neki omogućuju pronalaženje preklapajućih zajednica

(jedan vrh može biti dio nekoliko zajednica odjednom) [48], neki koriste pristup latentnog prostora (eng. *latent space approach*) za određivanje vjerojatnosti da vrh pripada određenoj zajednici [73] i slično.



Slika 4.1: Dendrogram zajednica otkriven u mreži Zachary Karate Club. Riječ je o društvenoj mreži od 34 člana karate kluba koju je u svom radu [171] opisao Wayne W. Zachary (preuzeto sa www.researchgate.net)

Detektiranje zajednica u proizvoljnoj mreži često je računalno zahtjevan zadatak. Broj zajednica u mreži, ako je opće podjela na zajednice moguća, obično je nepoznat i zajednice su nejednakih veličina i gustoća. Unatoč ovim poteškoćama, razvijene su i implementirane razne metode s različitim razinama uspjeha. U ovom poglavlju dajemo kratki pregled postojećih metoda i algoritama.

Metoda najmanjeg reza

Jedan od najstarijih algoritama za podjelu mreže u zajednice temelji se na metodi najmanjeg reza i njenim varijantama. U ovom pristupu mreža se dijeli u unaprijed

određeni broj dijelova, približno iste veličine, odabranih tako da je broj bridova između grupa minimalan. Metodu je moguće djelomično poboljšati tako da se dopusti da veličina zajednica varira. Umjesto da se minimizira samo broj bridova među zajednicama (u oznaci R), moguće je minimizirati omjer $\frac{R}{n_1 n_2}$, gdje su n_1, n_2 veličine zajednica [117]. Metoda je uspješna za one mreže za koje je originalno razvijena, ali općenito nije idealna jer će pronaći zajednice neovisno o tome jesu li one prirodno u strukturi mreže i pronaći će samo unaprijed zadani broj zajednica.

Girvan – Newman algoritam

Jedan od najčešće korištenih algoritama za detektiranje zajednica je Girvan–Newman algoritam [65]. Temelji se na uzastopnom uklanjanju bridova iz originalne mreže, a zajednicama se smatraju povezane komponente koje ostanu nakon uklanjanja bridova. U Poglavlju 1 definirali smo bridnu međupoloženost brida $uv \in E(G)$ kao postotak najkraćih putova među parovima vrhova u grafu koji prolaze tim bridom. Ukoliko postoji više od jednog takvog puta, svakom od njih dodijeljuje se jednaka težina. Ako mreža sadrži zajednice koje su međusobno povezane malim brojem bridova, onda bi svi putevi između različitih zajednica trebali prolaziti upravo tim bridovima. Dakle, takvi bridovi bi trebali imati visoke vrijednosti bridne međupoloženosti (točnije, barem jedan od njih bi trebao imati visoku vrijednost bridne međupoloženosti). Njihovim uklanjanjem zapravo se odvajaju zajednice i otkriva pripadna sturktura mreže. Algoritam možemo opisati u nekoliko koraka:

1. izračunava se vrijednost bridne međupoloženosti za sve bridove
2. uklanja se brid s najvećom vrijednošću bridne međupoloženosti
3. ponovo se izračunava vrijednost bridne međupoloženosti za sve bridove na koje je utjecalo uklanjanje prethodnog brida

4. ponavljaju se koraci 2 i 3 sve dok se ne uklone svi bridovi

Krajnji rezultat je dendrogram koji se konstruira od vrha prema dnu, čiji listovi predstavljaju pojedine vrhove.

Girvan–Newman algoritam daje zadovoljavajuće rezultate i implementiran je u većini standardnih programskih alata. Međutim, dosta je spor, računalne složenosti $O(m^2n)$ za mrežu s n vrhova i m bridova, zbog čega je nepraktičan za mreže s više od nekoliko tisuća vrhova [118].

Hijerarhijsko klasteriranje

U ovoj metodi definira se mjera sličnosti koja kvantificira neku (obično topološku) vrstu sličnosti među vrhovima, a zatim se vrhovi dijele u zajednice ovisno o odabranoj mjeri. Postoji više različitih pristupa grupiranju vrhova. Dva najjednostavnija su *single linkage*, u kojem se smatra da grupe vrhova predstavljaju zajednice ako i samo ako svi parovi vrhova u različitim grupama imaju odabranu mjeru sličnosti manju od nekog zadanog praga, i *complete linkage*, u kojem svi vrhovi unutar zajednice imaju odabranu mjeru sličnosti veću od zadane vrijednosti [117].

Metode temeljene na klikama

Klika je podgraf u kojem je svaki vrh povezan sa svim ostalim vrhovima unutar klike. Budući je to oblik maksimalne povezanosti, mnogi pristupi temelje se na otkrivanju klika u grafu i analiziranja na koji način se klike preklapaju [125]. Pošto vrh može istovremeno pripadati dvjema ili više klika također istovremeno može pripadati u dvije ili više zajednica [124]. Jedna od metoda temelji se na pronalaženju maksimalnih klika, odnosno određivanju klika koje nisu podgrafovi nijedne druge klike.

Maksimizacija modularnosti

Posljednih nekoliko desetljeća najpopularnije metode za detektiranje zajednica temelje se na maksimiziranju modularnosti [95, 120, 121]. Modularnost mjeri razliku između stvarnog postotka bridova unutar zajednica i očekivanog postotka u slučajnom grafu s istim brojem vrhova i distribucijom stupnjeva te se često koristi kao mjera kvalitete podjele mreže na zajednice. Predložene su brojne modifikacije ove mjere [62, 114, 178] kao i niz algoritamskih pristupa koji uključuju "greedy" algoritme [42, 118], spektralne metode [121], ekstremalnu optimizaciju [55], "simulated annealing " [69, 107] i matematičko programiranje [2].

Potraga za optimalnom (najvećem) vrijednosti modularnosti je NP–težak problem zahvaljujući činjenici da prostor mogućih particija mreže prebrzo raste [33]. Iz tog razloga potrebno je razviti heurističku strategiju pretraživanja da bi se ograničio prostor za pretraživanje, ali i sačuvao cilj optimizacije. Različiti pristupi ovoj problematici daju različite omjere između brzine i točnosti [47, 69].

Budući se algoritam koje predlažemo za detektiranje zajednica u kurikulnim mrežama temelji na propagaciji labela, u sljedećem poglavlju dajemo kratak pregled algoritama koji funkcioniraju po istom principu. LPA (Label Propagation Algorithm) algoritmi imaju neka korisna svojstva i daju zadovoljavajuće rezultate, u dostupnoj literaturi moguće je pronaći njihove modifikacije i proširenja [169, 176]. Neki od poznatijih primjera su Speaker–listener LPA koji otkriva preklapajuće zajednice i razvijen je primarno za društvene mreže [67, 168] i različite verzije Weighted LPA razvijene za utežene mreže [76, 100].

4.2 Algoritmi za propagaciju labela

4.2.1 LPA

Algoritam temeljen na propagaciji labela (*Label Propagation Algorithm (LPA)*) koristi samo strukturu mreže kao ulazni podatak i relativno je brz pa je jedan od često korištenih algoritama za detektiranje zajednica. U praksi se pokazalo da algoritam radi dobro i da je vrlo efikasan. Međutim, ima jedan veliki nedostatak. Budući uključuje nasumične procese, u različitim izvršavanjima algoritam daje različite particije mreže [96]. Takva nestabilnost je krajnje nepoželjna u praksi.

Osnovna ideja algoritma je sljedeća. Pretpostavimo da vrh x ima susjede $x_1, x_2, ..., x_k$ i da svaki od susjeda ima labelu koja označava zajednicu kojoj taj vrh pripada. Tada x odabire zajednicu na temelju labela svojih susjeda. Pretpostavimo da svaki vrh odabire biti u onoj zajednici kojoj pripada najveći broj njegovih susjeda. U početku postavljamo da svaki vrh ima jedinstvenu labelu i započinje proces propagacije koji se širi kroz mrežu. Kako se proces ponavlja, gusto povezane grupe vrhova brzo dolaze do koncenzusa koju labelu odabrati. Kada se stvori više takvih gusto povezanih skupina, nastavljaju se širiti dalje po mreži. Na kraju procesa propagacije, oni vrhovi koji imaju iste labele grupiraju se zajedno i tvore zajednicu.

Proces propagacije ponavlja se iterativno, pri čemu, u svakom koraku, svaki vrh ažurira svoju labelu na temelju labela svojih susjeda. Proces ažuriranja može biti sinkroni ili asinkroni. Prilikom sinkronog ažuriranja, vrh x u iteraciji t ažurira svoju labelu na temelju labela svojih susjeda u iteraciji $t-1$. Ako označimo s $C_x(t)$ labelu vrha x u iteraciji t onda je

$$C_x(t) = f(C_{x_1}(t-1), C_{x_2}(t-1), ..., C_{x_k}(t-1)).$$

Problem je što grafovi koji su bipartitni ili skoro bipartitni imaju velike oscilacije

u podjeli labela [131]. Stoga se češće koristi asinkrono ažuriranje gdje je

$$C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1)..., C_{x_{ik}}(t-1)),$$

pri čemu su $x_{i1}, .., x_{im}$ susjedi od x koji su već napravili ažuriranje u trenutnoj iteraciji, a $x_{i(m+1)}, .., x_{ik}$ oni koji to još nisu napravili u trenutnoj iteraciji. Formalno, pravilo ažuriranja labele za vrh x je

$$l_x^{new} = argmax \left(\sum_{u=1}^{n} A_{ux} \delta(l_u, l) \right) \tag{4.1}$$

gdje l_x^{new} označava novu labelu vrha x. Ako postoji više od jedne labele koja se najčešće pojavljuje među susjednima od x, onda se nova labela bira nasumičnim uniformnim odabirom. Redoslijed kojim se svih n labela vrhova u mreži ažurira u svakoj iteraciji također se odabire nasumično.

Algoritam možemo opisati u nekoliko koraka.

1. Postavimo početne labele za svaki vrh. Za dani vrh x neka je $C_x(0) = x$.

2. Postavimo t=1

3. Postavimo vrhove u nasumični redoslijed i označimo ga s X

4. Za svaki $x \in X$ odabran po tom specifičnom redoslijedu neka je

   $$C_x(t) = f(C_{x_{i1}}(t), ..., C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1)..., C_{x_{ik}}(t-1)).$$

   Funkcija f vraća labelu koja se najčešće pojavljuje među susjedima od x. Ako se dogodi da ima dvije ili više takvih labela, bira se jedna uniformno nasumično.

5. Ako svaki vrh ima labelu koju ima najveći broj njegovih susjeda, algoritam staje. Ako ne, postavlja se se $t = t + 1$ i vraća na korak 3.

Najznačajnije obilježje LPA je njegova računalna složenost (blizu linearnog vremena $O(m)$) [131]. Problem je što LPA nije stabilan: algoritam je osjetljiv na poredak kojim vrhovi ažuriraju labele u svakoj iteraciji, zbog čega rješenja mogu biti različita u različitim pokretanjima algoritma [96]. Ponekad LPA može završiti i trivijalnim rješenjem - svi vrhovi su identificirani u istoj zajednici [15].

4.2.2 LPAm

Barber i Clark proširili su LPA tako da su modificirali pravilo ažuriranja i predložili su novi algoritam kojeg su nazvali LPAm [15]. Umjesto da bira labelu koja se najčešće pojavljuje među njegovim susjedima, vrh x odabire labelu koja će rezultirati maksimalnim povećanjem modularnosti. Modularnost smo već spominjali u prijašnjim poglavljima, a sada dajemo preciznu matematičku definiciju. Promatramo neusmjerenu i neuteženu mrežu s n vrhova i m bridova koja je reprezentirana matricom susjedstva $\mathbf{A}$. Element A_{uv} je jednak 1 ako postoji brid između vrhova u i v, a 0 inače. Stupanj vrha u označit ćemo s d_u. Pretpostavimo da je mreža podijeljena u N_c zajednica, tako da vrh u pripada zajednici l_u. Modularnost zapravo mjeri stvarni omjer bridova unutar zajednice umanjen za očekivanu vrijednost u nul–modelu, gdje je podjela na zajednice ista, ali se bridovi između vrhova postavljaju nasumično [97]. Formalno, modularnost se definira kao

$$Q = \frac{1}{2m} \sum_{u,v=1}^{n} (A_{uv} - P_{uv}) \delta(l_u, l_v) \tag{4.2}$$

gdje je $P_{uv} = d_u d_v / 2m$ vjerojatnost da postoji brid između u i v u nul–modelu, a $\delta(l_u, l_v)$ je Kroneckerova delta. Nadalje, ako definiramo matricu modularnosti $\mathbf{B}$ s elementima $B_{uv} = A_{uv} - P_{uv}$, modularnost je moguće izraziti i kao

$$Q = \frac{1}{2m} \sum_{u,v=1}^{n} B_{uv} \delta(l_u, l_v) \tag{4.3}$$

Jednadžbu (4.3) možemo zapisati tako da odvojimo elemente koji se odnose na labelu vrha x od ostalih

$$Q = \frac{1}{2m}\left(\sum_{u\neq x}\sum_{v\neq x} B_{uv}\delta(l_u, l_v) - B_{xx}\right) + \frac{1}{m}\left(\sum_{u=1}^{n} B_{ux}\delta(l_u, l)\right) \tag{4.4}$$

Prilikom ažuriranja labele vrha x, ako odaberemo labelu koja maksimizira drugi pribrojnik u (4.4), zapravo maksimiziramo modularnost Q. Dakle, pravilo za ažuriranje labele vrha x je

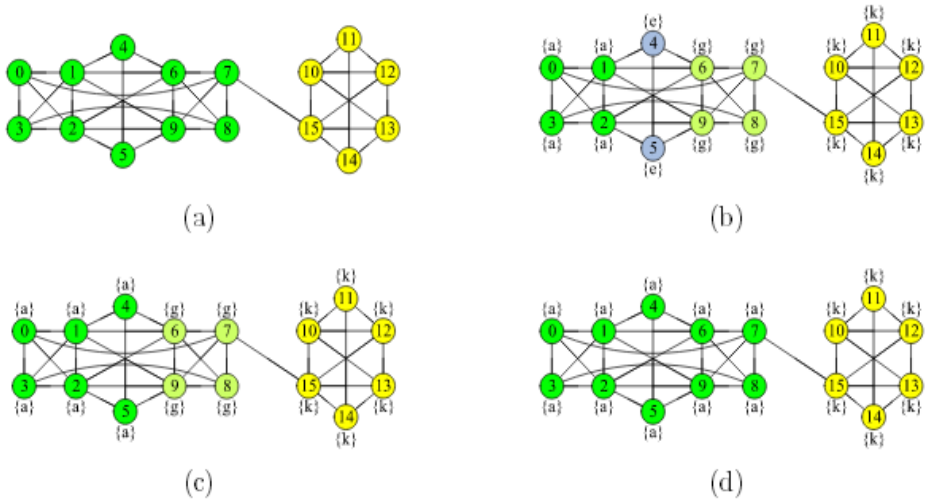
$$l_x^{new} = argmax\left(\sum_{u=1}^{n} B_{ux}\delta(l_u, l)\right) \tag{4.5}$$

Implementacija LPAm donosi monotono povećanje modularnosti i zaobilazi mogućnost formiranja trivijalnog rješenja. Uz to, LPAm ima jednaku efikasnu brzinu kao i LPA. Međutim, sklon je zapeti u lošem lokalnom maksimumu što se tiče modularnosti [97].

4.2.3 LPAm+

Liu i Murata zaobilaze ovaj problem spajanjem zajednica koje maksimizira modularnost. Po pravilu ažuriranja labela (4.5), LPAm preferira podjelu na zajednice koje su sličnog ukupnog stupnja. Da bi se riješili toga ograničenja, kada LPAm zapne u lokalnom maksimumu (daljnjom propagacijom labela nije moguće dobiti veću modularnost), računa se promjena u modularnosti koju uzrokuje spajanje parova zajednica i spoje se one zajednice koje rezultiraju najvećim povećanjem modularnosti. Na taj način se izbjegne zapinjanje u lokalnom maksimumu modularnosti. Nakon spajanja potrebno je opet izvršiti LPAm. Međutim, nije sigurno da će novi lokalni maksimum do kojeg dođemo biti dovoljno dobar (iako je bolji od prethodnog). Iz tog razloga potrebno je ponavljati ovaj proces sve dok više nije moguće dobiti povećanje modularnosti. Predloženi algoritam nazvan je LPAm+. Na Slici 4.2 prikazano je na koji način funkcionira.

Što se tiče složenosti, jedan korak propagacije labela u LPAm ima složenost $O(m)$, pa je ukupna složenost LPAm jednaka $O(rm)$ gdje je r broj koraka potreban za doseći lokalni maksimum modularnosti. Nadalje, za jedan postuoak spajanja zajednica potrebno je $O(mlogn)$. Ako označimo s h broj koliko puta će doći do spajanja zajednica, ukupna složenost LPAm+ je $O(rm) + h(O(mlogn) + O(rm)$ [97]. Točna procjena parametra h nije moguća, budući ovisi o kvaliteti rješenja dobivenog pomoću LPAm algoritma.



Slika 4.2: **Mreža igračaka.** **(a)** Intuitivna podjela na 2 zajednice. **(b)** LPAm zapne u lokalnom maksimumu (podjela na 4 zajednice i modularnost 0.399). **(c)** Spajamo zajednice s labelama a i e (što uzrokuje porast modularnosti za 0.008. **(d)** Nakon još jednog izvršavanja LPAm dolazimo do globalnog maksimuma ($Q = 0.413$). (preuzeto od [97])

4.3 Detektiranje zajednica u usmjerenim mrežama

Mreže (ili grafovi) pojavljuju se kao dominantne strukture u različitim domenama, uključujući sociologiju, biologiju, neuroznanosti i informatiku. U većini spomenutih slučajeva grafovi su usmjereni što mijenja semantiku bridova koji više nisu simetrični, u smislu da početni vrh prenosi neko svojstvo ili vrijednost na krajnji vrh, ali ne i obrnuto. Otkrivanje strukture zajednica u usmjerenim kompleksnim mrežama je interdisciplinarna tema s mnoštvom relevantnih područja primjene. Problem detekcije zajednica u usmjerenim mrežama smatra se izazovnijim zadatakom u odnosu na neusmjerene mreže. Naglašavajući pozadinu problema, u svom nedavnom radu Santo Fortunato je izjavio: *"Razvoj metoda za detekciju zajednica u usmjerenim mrežama je težak zadatak. Na primjer, usmjereni graf karakteriziraju asimetrične matrice (matrica susjedstva, Laplacian, itd), pa je spektralna analiza mnogo složenija. Samo nekoliko metoda može se lako prilagoditi iz neusmjerenih mreža. Inače, problem se mora formulirati od nule"* [61].

Precizna i općeprihvaćena definicija problema detekcije zajednica u usmjerenim mrežama još uvijek ne postoji. Ideja o tome da postoji više bridova unutar zajednice nego među zajednicama ne može se jednostavno proširiti na usmjereni slučaj zbog nepostojanja simetrije brida. Jasno je da ignoriranje usmjerenosti bridova i promatranje mreže kao da je neusmjerena nije dovoljno dobar način podjele usmjerene mreže budući se tako ne uzima u obzir asimetričan odnos koji se implicira usmjerenošću brida, iako je ponekad i na taj način moguće doći do nekih spoznaja o strukturi mreže. Dakle, glavni izazov je pronalaženje smislenih načina kako da se usmjerenost brida uklopi u proces detekcije zajednica [102]. Postoji više različitih pristupa ovom problemu, ovisno o načinu na koji se tretiraju usmjereni bridovi. U ovom poglavlju dajemo kratki pregled osnovnih metoda i principa.

Naivni pristup transformacije grafa

U ovu klasu spadaju svi algoritmi koji ignoriraju usmjerenost bridova i tretiraju mrežu kao neusmjerenu. Svi algoritmi predloženi za detekciju zajednica u neusmjerenim mrežama mogu se na ovaj način primijeniti za otkrivanje temeljne strukture zajednica u usmjerenoj mreži. Međutim, zbog naivne transformacije grafa ne uzimaju se u obzir korisne informacije pri otkrivanju zajednica. Na primjer, razmotrimo mrežu citata, gdje znanstveni radovi predstavlju vrhove, a bridovi citate. Pretpostavimo da rad i citira rad j, ali ne i obrnuto. Korištenjem naivne transformacije grafa gubi se informacija koji rad se temelji na kojem i činjenica da je rad i objavljen nakon rada j.

Transformacije koje zadržavaju usmjernost

Metode i principi u ovoj klasi zasnivaju se na pretvaranju usmjerene mreže u neusmjerenu, ali se usmjerenost brida značajno odražava u novonastaloj mreži. Na primjer, u nekim pristupima usmjerena mreža konvertira se u neusmjerenu i uteženu, pri čemu su informacije o usmjerenosti brida uključene putem težine bridova [135]. U drugim pristupima, usmjerena mreža može se pretvoriti u bipartitnu [177] i slično.

Proširivanje funkcija cilja i metodologije

Ovdje ubrajamo sve pristupe koji na neki način proširuju metodologije iz neusmjerenih mreža. Problem detekcije zajednica se obično izražava kao problem optimizacije, pri čemu se objektivni kriterij ili neka mjera, koji opisuju željena svojstva zajednica, optimiziraju uzastopnim dodavanjima vrhova u različite zajednice. Algoritmi se obično iterativno ponavljaju dok se ne nađe lokalni minimum ili maksimum tražene vrijednosti. Prirodni način prenošenja ove taktike u usmjerene

mreže je modifikacija i proširivanje definicije odabrane mjere tako da se usmjerenost bridova smatra unutarnjim svojstvom mreže. Neki istaknuti predstavnici ove kategorije su usmjerene verzije modularnosti [11, 86, 95] i objektivne funkcije metode minimalnih rezova u usmjerenim mrežama[109].

Alternativni pristupi

Ova kategorija uključuje pristupe koji slijede različite metodologije, koje se uglavnom razlikuju od onih opisanih u prethodne tri kategorije. Identificiramo tri glavne vrste metoda i to (i) informacijsko–teorijske metode, (ii) metode koje se temelje na probabilističkim modelima i statističkom zaključivanju te (iii) stohastičke metode. Iako su posljednje dvije metodologije usko povezane i obje se oslanjaju na vjerojatnosne modela, možemo ih pregledati samostalno jer se oslanjaju na različite koncepte statističkog zaključivanja [102].

4.4 Detektiranje zajednica u kurikulnim mrežama

Za detektiranje zajednica u kurikulnim mrežama, osim usmjerenosti brida, potrebno je uzeti u obzir i zahtjev za parcijalnom uređenošću skupa edukacijskih jedinica. Nedavno je predložena mjera modularnosti za usmjerene acikličke mreže Q^{dag} razvijena definiranjem odgovarajućeg nul–modela i poštujući redoslijed vrhova. Provedena je spektralna metoda kako bi se maksimizirala predložena mjera modularnosti koja je testirana na mrežama citata i ostalim acikličkim usmjerenim mrežama. Utvrđeno je da su dobivene vrijednosti modularnosti Q^{dag} slične za particije dobivene maksimiziranjem predložene modularnosti Q^{dag}, modularnosti za neusmjerene mreže Q i modularnosti za usmjerene mreže Q_d (definirane u formuli (4.6)). Drugim riječima, ako se zanemari redoslijed vrhova i maksimizira konvencionalna mjera modularnosti, dobivena particija je blizu optimalne u

smislu modularnosti Q^{dag} [140]. Iako je redoslijed vrhova uzet u obzir prilikom definiranja mjere modularnosti Q^{dag}, nema zahtjeva da dobivene zajednice budu u određenom redosllijedu. Koliko nam je poznato, nijedan dostupni algoritam za detektiranje zajednica u usmjerenim mrežama ne udovoljava zahtjevu da dobivene zajednice poštuju uvjet naveden na početku poglavlja. Stoga je potrebno razviti nove metode i algoritme za detektiranje zajednica u kurikulnim mrežama.

U ovom poglavlju dajemo opis algoritma koji predlažemo za detektiranje zajednica u kurikulnim mrežama. Algoritam se temelji na modifikacijama opisanih LPA algoritama i proširenju definicije modularnosti na usmjerene mreže. Neka je dana kurikulna mreža s n vrhova i m bridova i neka je dana njena matrica susjedstva $\mathbf{A}$. Već smo komentirali da je matrica susjedstva kurikulne mreže gornjetrokutasta s nulama na glavnom dijagonali. Neka su $d^{in}(i)$ i $d^{out}(i)$ redom instupanj i outstupanj vrha $i \in V(G)$. Neka vrh i pripada zajednici l_i. Modularnost za usmjerene mreže definira se kao [95]

$$Q_d = \frac{1}{m} \sum_{1 \leq i,j \leq n} \left[A_{ij} - \frac{d^{in}(j) d^{out}(i)}{m} \right] \delta(l_i, l_j) \tag{4.6}$$

pri čemu je $\delta(l_i, l_j)$ Kroneckerova delta. Ako definiramo matricu $\mathbf{B}$ s elementima

$$B_{ij} = A_{ij} - \frac{d^{in}(j) d^{out}(i)}{m},$$

definiciju modularnosti u usmjerenim mrežama možemo zapisati kao

$$Q_d = \frac{1}{m} \sum_{1 \leq i,j \leq n} B_{ij} \delta(l_i, l_j) \tag{4.7}$$

Budući je $\delta(l_i, l_j) = \delta(l_j, l_i)$, formulu 4.6 moguće je napisati kao [86]

$$Q_d = \frac{1}{m} \sum_{1 \leq i,j \leq n} \left(A_{ij} + A_{ji} - \frac{d^{in}(j) d^{out}(i)}{m} - \frac{d^{out}(j) d^{in}(i)}{m} \right) \delta(l_i, l_j) \tag{4.8}$$

Odnos između modularnosti Q_d u usmjerenoj mreži i modularnosti Q u pripadnoj neusmjerenoj mreži može se izraziti jednadžbom

$$Q_d = Q + \frac{1}{4M^2} \sum_{i,j} \Delta_i \Delta_j \delta(l_i, l_j), \tag{4.9}$$

gdje je Q modularnost neusmjerene mreže koja se dobije ignoriranjem usmjerenosti bridova u originalnoj mreži i $\Delta_i = d^{out}(i) - d^{in}(i)$. Lako se vidi iz jednadžbe (4.9) da drugi element ima pozitivan doprinos vrijednosti Q_d ako i samo ako su Δ_i i Δ_j oba pozitivni ili oba negativni [86].

Naš cilj je dobiti podjelu kurikulne mreže na zajednice tako da je modularnost Q_d maksimalna, ali uz uvjet da za sve povezane edukacijske jedinice $x_i, x_j \in V(G)$ takve da je $x_i \in A_i$ i $x_j \in A_j$ vrijedi $A_i \prec A_j$ ili $A_i = A_j$.

4.4.1 OLPAm+ algoritam

Algoritam koji predlažemo za detektiranje zajednica u kurikulnim mrežama je heuristički algoritam koji se temelji na propagaciji labela i maksimizaciji modularnosti Q_d definirane u (4.6). Osnovna ideja slična je ideji LPA algoritama opisanih u prošlom poglavlju. Vrhovi uzimaju jednu od labela svojih susjeda koja maksimalno povećava modularnost Q_d, ali da se pritom ne poremeti zahtjev za parcijanom uređenošću skupa dobivenih zajednica. U svakom koraku odabire se optimalno rješenje pronađeno na temelju trenutno dostupnih informacija u nadi da će se konačno rješenje približiti globalnom optimumu. Ovakva algoritamska paradigma poznata je kao "greedy algoritm" [71] i često se primjenjuje u optimizacijskim problemima.

Algoritam možemo opisati kako slijedi. Svakom vrhu $i \in V(G)$ dodijeljuje se jedinstvena numerička labela l_i. Vrhovi se stavljaju u slučajni poredak i započinje proces propagacije labela. Za svaki vrh u tom poretku računa se promjena modularnosti uzrokovana promjenom labele. Iz jednadžbe (4.8) slijedi da povećanje

modularnosti mogu izazvati oni parovi vrhova $i, j \in V(G)$ za koje je $A_{ij} \neq 0$ ili $A_{ji} \neq 0$ pa vrh može ažurirati svoju labelu na jednu od labela svojih susjeda (ili insusjeda ili outsusjeda). Kada vrh i promijeni labelu, tj. promijeni zajednicu kojoj pripada, izaziva promjenu modularnosti koja se može izračunati na sljedeći način. Neka vrh i mijenja postojeću labelu l_i u novu labelu l_j. Promjena modularnosti koju izaziva ta promjena slijedi iz jednadžbe (4.8) i računa se kao

$$\Delta Q_d(ij) = \frac{d_i^j}{m} - \left[\frac{d^{out}(i)S_{in}(j) + d^{in}(i)S_{out}(j)}{m^2}\right] \tag{4.10}$$

pri čemu je:

- d_i^j ukupan broj insusjeda i outsusjeda od i koji imaju labelu l_j
- $S_{in}(j)$ je ukupan instupanj vrhova s labelom l_j (suma svih $d^{in}(u)$ takvih da je $l_u = j$)
- $S_{out}(j)$ je ukupan outstupanj vrhova s labelom j (suma svih $d^{out}(u)$ takvih da je $l_u = j$)

Proces ažuriranja labela je asinkron. Ako vrh i mijenja labelu u trenutku t, neki njegovi susjedi $j \in V(G)$ su već promijenili labelu u trenutnoj iteraciji i imaju labelu $l_j(t)$, dok neki susjedi $k \in V(G)$ još uvijek imaju labelu iz prethodne iteracije $l_k(t-1)$. Vrh i donosi odluku na temelju stanja koje je zatekao u trenutku t, odnosno odabire onu labelu koja uzrokuje maksimalno povećanje modularnosti bez da se poremeti poredak zajednica. Ukoliko takva labela ne postoji, vrh zadržava svoju labelu. Da bismo osigurali da vrhovi (i dobivene zajednice) budu u valjanom poretku, vrh i može odabrati najveću među labelama svojih insusjeda ili najmanju među labelama svojih outsusjeda. U suprotnom bi došlo do premećaja poretka. Naime, pretpostavimo da vrh x_i ima insusjede x_j, x_k, x_m s labelama l_j, l_k, l_m redom, pri čemu vrijedi da je $l_j < l_k < l_m$. Ukoliko vrh x_i uzme, na primjer, labelu

l_k, dolazi do poremećaja u poretku vrhova jer vrh x_i ima labelu l_k koja je manja od labele l_m njegovog insusjeda x_m. Analogno vrijedi za outsusjede. Kada je točno jednom razmotren svaki vrh u poretku, stavljamo vrhove u slučajni poredak i ponavljamo postupak. Algoritam se zaustavlja kada mijenjanjem labela više nije moguće dobiti pozitivnu promjenu modularnosti. Na kraju algoritma identificiramo zajednice kao skupinu vrhova koji imaju istu labelu. Opisani algoritam nazvat ćemo *Orientation Respecting LPAm (OLPAm)*. Pseudokod je dostupan u Algoritmu 4.

Kao što je opisano u prošlom poglavlju, LPAm algoritam je osjetljiv na poredak vrhova koji je slučajan u svakoj iteraciji i sklon je zapeti u lošem lokalnom maksimumu modularnosti. Isto vrijedi iza OLPAm. Kada se to dogodi, računamo promjenu u modularnosti koja nastaje spajanjem parova zajednica i spajamo onaj par zajednica koji rezultira najvećim povećanjem modularnosti i ne remeti poredak zajednica. Ako spojimo zajednice l_i i l_j, promjena modularnosti uzrokovana spajanjem može se izračunati kao

$$\Delta Q_d(l_i l_j) = \frac{E_{ij}}{m} - \left[\frac{S_{out}(i)S_{in}(j) + S_{in}(i)S_{out}(j)}{m^2}\right] \tag{4.11}$$

pri čemu je E_{ij} broj bridova između zajednica l_i i l_j. Jednadžba (4.11) dobije se iz (4.10) zbrajanjem po svim vrhovima u zajednici l_i. Iz jednadžbe (4.11) jasno je da je dovoljno izračunati $\Delta Q_d(l_i l_j)$ za parove zajednica koje su povezane jer samo povezane zajednice mogu dati pozitivnu promjenu modularnosti. Iako spajanjem zajednica dolazi do povećanja modularnosti, nije sigurno da je dobiveni maksimum ujedno i globalni, pa opet pozivamo OLPAm i ponavljamo postupak dokle god je moguće spajanjem zajednica dobiti povećanje modularnosti bez da se poremeti dobiveni poredak zajednica. Da bi se sačuvao valjani poredak zajednica, razmotrimo sljedeće. Ako postoji barem jedan vrh x_i u zajednici A_i i barem jedan vrh x_j u zajednici A_j takav da postoji brid $x_i x_j \in E(G)$ kažemo da zajednica A_i

pokazuje na zajednicu A_j. Analogno kao s vrhovima, zajednicu A možemo spojiti s onom zajednicom A_i koja ima najveću labelu l_{max} među labelama zajednica koje pokazuju na A ili sa zajednicom A_j koja ima najmanju labelu l_{min} među labelama zajednica na koje A pokazuje. U suprotnom neće biti zadovoljen uvjet s početka poglavlja. Cjelokupni algoritam (OLPAm sa spajanjem zajednica) nazvali smo *Orientation Respecting LPAm+ (OLPAm+).* Pseudokod se može pronaći u Algoritmu 5.

Što se tiče računalne složenosti OLPAm+, jedan korak propagacije labela u OLPAm ima složenost $O(n)$ budući za svaki vrh računamo dvije vrijednosti, pa je ukupna složenost OLPAm algoritma jednaka $O(rn)$ gdje je r maksimalni broj koraka propagacije labela potreban za pronalaženje maksimalne vrijednosti modularnosti. Nadalje, jedan postupak spajanja zajednica u OLPAm+ algoritmu ima složenost $O(n)$. Naime, n je maksimalni broj dobivenih zajednica pa je za računanje svih vrijednosti $\Delta Q_d(l_i l_j)$ potrebno $O(n)$ vremena. Određivanje maksimalne vrijednosti niza od n elemenata ima složenost $O(n)$ što daje ukupnu složenost jednog spajanja zajednica $O(n + n) = O(n)$. Neka je h broj puta koliko dolazi do spajanja zajednica. Ukupna računalna složenost algoritma je $O(rn) + h(O(n) + O(rn)) = O(n)$. Vrijednost parmetra h nije moguće precizno procijeniti jer ovisi o kvaliteti rješenja dobivenog u OLPAm algoritmu. Čak ni vrijednost parametra r nije moguće u potpunosti predvidjeti. U [131] autori navode da je u LPA algoritmu broj koraka propagacije labela potreban za konvergenciju algoritma neovisan o broju vrhova. U Tablici 4.1 prikazane su prosječne vrijednosti parametara dobivene pokretanjem OLPAm+ algoritma 100 puta.

Algorithm 4 Orientation Respecting LPAm **(OLPAm)**

Input: Lista bridova

Output: Podjela na zajednice, modularnost

svakom vrhu i dodijeli jedinstvenu labelu $l_i(0) = p(i)$
postavi $t = 1$
repeat
  poredaj vrhove u slučajni poredak X
  for svaki vrh $i \in X$ **do**
    među insusjedima $x_{i_1}, x_{i_2}, ..., , x_{i_k}$ vrha i s labelama $l_{i_1}, l_{i_2}, ..., l_{i_n}$ odredi najveću labelu l_{max}
    među outsusjedima $x_{i_{k+1}}, x_{i_{k+2}}, ..., , x_{i_n}$ vrha i s labelama $l_{i_{k+1}}, l_{i_{k+2}}, ..., l_{i_n}$ odredi najmanju labelu l_{min}
    izračunaj $\Delta Q_d(i, max)$ i $\Delta Q_d(i, min)$
    if $\Delta Q_d(i, max) > \Delta Q_d(i, min)$ i $\Delta Q_d(i, max) > 0$ **then**
      postavi $l_i(t) = l_{max}$
    else if $\Delta Q_d(i, min) > \Delta Q_d(i, max)$ i $\Delta Q_d(i, min) > 0$ **then**
      postavi $l_i(t) = l_{min}$
    else if $\Delta Q_d(i, min) = \Delta Q_d(i, max) > 0$ **then**
      uniformno slučajno odaberi jednu od labela l_{max} ili l_{min} i postavi je za labelu vrha i
    end if
    postavi $t = t + 1$
  end for
  if nijedan vrh $i \in X$ ne mijenja labelu **then**
    završi algoritam
  else
    postavi $t = t + 1$
  end if
until niijedan vrh u iteraciji ne promijeni labelu

Algorithm 5 Orientation Respecting LPAm+ (**OLPAm+**)

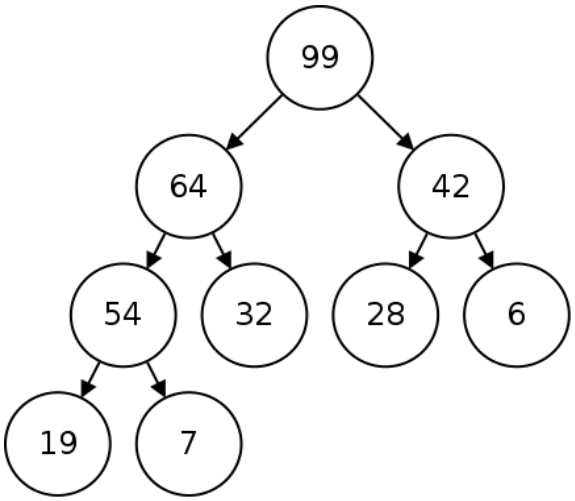
svakom vrhu dodijeli jedinstvenu numeričku labelu
koristeći OLPAm algoritam maksimiziraj modularnost Q_d
while postoje zajednice A_i i A_j takve da je $\Delta Q_d(l_i l_j) > 0$ **do**
  for svaku zajednicu A_i **do**
    izračunaj $\Delta Q_d(l_i l_{max})$ i $\Delta Q_d(l_i l_{min})$
  end for
  odredi maksimum svih dobivenih vrijednosti $\Delta Q_d(l_i l_j) > 0$
  spoji zajednice A_i i A_j za koje je $\Delta Q_d(l_i l_j) > 0$ maksimalan
  maksimiziraj modularnos Q_d pomoću OLPAm algoritma
end while

Tablica 4.1: **Procjene vrijednosti parametara r i h u OLPAm+ algoritmu.** Prikazan je prosječan broj koraka r potreban da algoritam OLPAm konvergira i prosječan broj puta h spajanja zajednica u OLPAm+ algoritmu primijenjenom na kurikulne mreže.

Mreža	n	m	r	h
Skup $\mathbb{Q}$	47	254	7.31	3.25
Elementarne funkcije	84	502	6.09	2.64
Integral	223	656	11.02	5.51
Obrada podataka	54	197	6.09	2.64
Model primarne proizvodnje	28	93	7.11	1.23
Fizika	31	49	4.77	1.99

Za određivanje maksimalne vrijednosti $\Delta Q(l_i l_j)$ korišteno je *max–heap* binarno stablo. Stablo predstavlja važnu strukturu podataka, pogodnu za modeliranje objekata koji oslikavaju hijerarhijsku organizaciju [165]. U binarnom stablu svaki čvor ima najviše dva djeteta. *Max–heap* je binarno stablo koje ima sljedeća dva svojstva [12] (primjer prikazan na Slici 4.3):

- svi listovi u stablu se nalaze na najviše dvije susjedne razine
- vrijednost spremljena u pojedinom čvoru je veća ili jednaka vrijednosti pohranjenoj u njegovoj djeci.



Slika 4.3: ***Max–heap* stablo**. (preuzeto sa http://cs.umw.edu/ finlayson)

Max–heap stablo moguće je kreirati iz niza ulaznih elemenata uzastopnim umetanjem svakog elementa [165]. Umetanje elementa u stablo sastoji se od sljedećih koraka:

1. umetanje elementa na najnižu razinu stabla
2. usporedba dodanog elementa s njegovim roditeljem. Ako su u točnom poretku, stani.
3. ako nisu u točnom poretku, zamijeni ih i vrati se na prethodni korak.

Broj potrebnih operacija ovisi o broju razina koje element mora proći da bi zadovoljio svojstva *max–heap* stabla, pa je računalna složenost u najgorem slučaju $O(logn)$ [70]. Ukupna složenost kreiranja stabla je $O(nlogn)$: umeće se n elemenata i za svaki je potrebno $O(logn)$ operacija [45]. Pogledamo li detaljniju analizu složenosti, moguće je postići složenost $O(n)$. Na svakoj razini h nalazi se najviše $\lceil n/2^{h+1} \rceil$ vrhova i potrebno je najviše $O(h)$ operacija da se vrh s razine h smjesti na valjanu poziciju u stablu. Ako je $\lfloor logn \rfloor$ visina stabla, vrijedi:

$$\sum_{h=0}^{\lfloor logn \rfloor} \lceil n/2^{h+1} \rceil O(h) = O\left(n \sum_{h=0}^{\lfloor logn \rfloor} \frac{h}{2^h} \right).$$

Nadalje:

$$\sum_{h=0}^{\infty} x^h = \frac{1}{1-x}.$$

Deriviramo obje strane, pomnožimo s x i dobijemo

$$\sum_{h=0}^{\infty} hx^h = \frac{x}{(1-x)^2}.$$

Uvrstimo $x = \frac{1}{2}$ i dobijemo

$$\sum_{h=0}^{\infty} hx^h = \frac{\frac{1}{2}}{(1-\frac{1}{2})^2} = \frac{\frac{1}{2}}{\frac{1}{4}} = 2.$$

Budući je

$$\sum_{h=0}^{\lfloor logn \rfloor} hx^h \leq \sum_{h=0}^{\infty} hx^h,$$

vrijedi

$$O\left(n\sum_{h=0}^{\lfloor logn\rfloor}\frac{h}{2^h}\right)=O(n).$$

Dakle, konstrukcija *max–heap* stabla ima složenost $O(n)$.

Brisanje korijenskog čvora koji ima maksimalnu vrijednost odvija se na sljedeći način:

1. zamijeni se korijenski čvor sa zadnjim elementom na zadnjoj razini
2. usporedi se dodani element s njegovom djecom. Ako su u točnom poretku, stani.
3. ako nisu u točnom poretku zamijeni se korijenski čvor s jednim od njegove djece i vrati se na prethodni korak.

U najgorem slučaju, novi korijenski čvor mora biti zamijenjen sa svojim djetetom na svakoj razini dok ne dođe opet do posljednje razine, pa brisanje elementa ima složenost $O(logn)$.

Prilikom spajanja zajednica u OLPAm+ algortimu, spojili smo samo one dvije zajednice čije spajanje rezultira najvećim povećanjem modularnosti bez remećenja poretka. Po uzoru na [97], razmotrit ćemo i modifikaciju OLPAm+ algoritma. Kada OLPAm zapne u lokalnom maksimumu (daljnjom propagacijom labela nije moguće dobiti povećanje modularnosti), računamo promjene modularnosti uzrokovane spajanjem zajednica i spajamo one parove zajednica koji najviše povećavaju modularnost. Pseudokod ovako modificiranog OLPAm+ algoritma dan je u Algoritmu 6.

Algorithm 6 Modificirani OLPAm+ s višestrukim spajanjem zajednica

1: svakom vrhu dodijeli jedinstvenu numeričku labelu
2: koristeći OLPAm algoritam maksimiziraj modularnost Q_d
3: **while** $\exists$ par zajednica (A_i, A_j) takav da je $\Delta Q(l_i, l_j) > 0$ **do**
4: **for** svaki par povezanih zajednica (A_i, A_j) gdje je $\Delta Q(l_i, l_j) > 0$ **do**
5: **if** ne postoji zajednica A s labelom l takva da je $\Delta Q(l, l_i) > \Delta Q(l_i, l_j)$ i $\Delta Q_{l,l_j} > \Delta Q(l_i, l_j)$ **then**
6: spoji zajednice A_i i A_j
7: **end if**
8: **end for**
9: maksimiziraj modularnost Q_d pomoću OLPAm algoritma
10: **end while**

Ako se sve izračunate vrijednosti $\Delta Q(l_i, l_j)$ stave u *max–heap* stablo, nakon spajanja onog para zajednica koji daje najveće povećanje modularnosti, potrebno je izbrisati korijenski čvor iz stabla i poredati ostale čvorove u valjanu strukturu, pa je računalna složenost jednog spajanja parova zajednica koje odgovara While petlji u Algoritmu 6 jednaka $O(nlogn)$. Ukupna složenost ovako modificiranog algoritma $O(rn) + h(O(nlogn) + O(rn))$.

4.4.2 Eksperimenti i rezultati

Predloženi algoritam implementiran je u programskom alatu *Microsoft Visual Studio 2015*. Budući je algoritam osjetljiv na slučajni poredak kojim se vrši ažuriranje labela, rezultati mogu varirati u svakom izvršavanju algoritma. Iz tog razloga, po uzoru na [97], pokrenuli smo algoritam 100 puta za svaku od kurikulnih mreža opisanih u Poglavlju 3 (osnovne statistike mreža mogu se pronaći u Tablici 3.1). Vrijednosti modularnosti Q_d i broja zajednica dobivenih primjenom algoritma mogu se pronaći u Tablici 4.2.

Budući nam nije poznat nijedan algoritam koji zahtjeva uređenost skupa dobivenih zajednica, usporedili smo vrijednosti dobivene predloženim OLPAm+ algoritmom s vrijednostima dobivenim LPAm+ algoritmom [97] opisanim u Poglavlju 4.2.3 i s Girvan–Newman algortimom [65] opisanim u Poglavlju 4.1 na sljedeći način. LPAm+ algoritam pokrenut je 100 puta za svaku od kurikulnih mreža i promatrana je maksimalna vrijednost dobivene modularnosti Q. Za tako dobijenu podjelu na zajednice, izračunali smo vrijednosti modularnosti za usmjerene mreže Q_d dane u formuli (4.6). Isto je napravljeno i za podjelu dobivenu Girvan–Newmanovim algoritmom. Iako je na ovaj način zanemaren zahtjev za poretkom zajednica, možemo usporediti koliko spomenuti zahtjev utječe na vrijednost modularnosti i kvalitetu podjele mreže na zajednice. Usporedba rezultata prikazana je u Tablici 4.4. Iz nje se vidi da OLPAm+ daje dosta dobre rezultate u usporedbi s druga dva navedena algoritma iako je djelomično ograničen zahtjevom za valjanim poretkom dobivenih zajednica. Za mreže *Skup* $\mathbb{Q}$ i *Fizika* dalje bolje rezultate, dok za mreže *Elementarne funkcije*, *Obrada podataka* i *Model primarne proizvodnje* daje neznatno slabije rezultate. Za mrežu *Integral* zahtjev za poretkom zajednica rezultira manjom vrijednošću modularnosti u odnosu na LPAm+ i Girvan–Newman algoritam.

Tablica 4.2: **OLPAm+ algoritam za detektiranje zajednica u kurikulnim mrežama.** Prikazani su osnovni rezultati dobiveni pokretanjem OLPAm+ algoritama (OLPAm+ s jednim spajanjem zajednica i OLPAm+ s višesrukim spajanjem zajednica) 100 puta za svaku od kurikulnih mreža opisanih u Poglavlju 3. *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, Q_{max} je maksimalna vrijednost modularnosti dobijena iz 100 puta pokretanja algoritma, N_c je broj zajednica koji daje maksimalnu vrijednost modularnosti Q_{max}, $OLPAm+(vs)$ je verzija algoritma s višestrukim spajanjem zajednica.

			$OLPAm+$		$OLPAm+(vs)$	
	n	m	Q_{max}	N_c	Q_{max}	N_c
Skup $\mathbb{Q}$	47	254	0.377	4	0.377	4
Elementarne funkcije	84	502	0.354	4	0.337	5
Integral	223	656	0.468	7	0.470	10
Obrada pod.	54	197	0.426	5	0.426	5
Model prim. proizvodnje	28	93	0.293	3	0.293	3
Fizika	31	49	0.476	6	0.467	5

Usporedili smo rezultate dobivene OLPAm+ algoritmom s podjelom na zajednice koju su predložili autori svake od kurikulnih mreža. Za predloženu podjelu za svaku mrežu izračunali smo vrijednosti modularnosti Q_d. Rezultati se nalaze u Tablici 4.3. Za svaku kurikulnu mrežu algoritam daje veće vrijednosti modularnosti, odnosno bolju podjelu na zajednice.

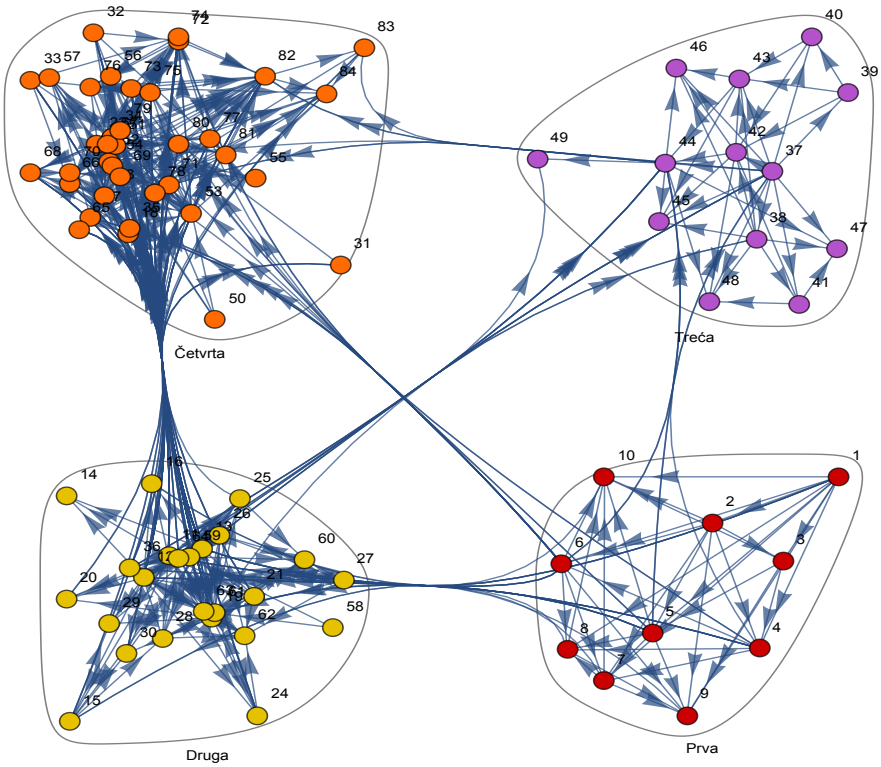
Tablica 4.3: **Usporedba rezultata OLPAm+ algoritma s podjelom na zajednice predloženom od strane stručnjaka.** Prikazani su osnovni rezultati dobiveni pokretanjem OLPAm+ algoritma 100 puta za svaku od kurikulnih mreža kao i rezultati dobiveni računanjem vrijednosti modularnosti Q_d za podjelu koju su predložili autori pojedine kurikulne mreže. *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, Q_{max} je maksimalna vrijednost modularnosti dobijena iz 100 puta pokretanja algoritma, N_c je broj zajednica koji daje maksimalnu vrijednost modularnosti Q_{max}, Q_d je vrijednost modularnosti dobivena za predloženu podjelu na zajednice.

			OLPAm+		Stručnjak	
	n	m	Q_{max}	N_c	Q_d	N_c
Skup $\mathbb{Q}$	47	254	0.377	4	0.311	5
Elementarne funkcije	84	502	0.354	4	0.239	6
Integral	223	655	0.468	7	0.455	10
Obrada pod.	54	197	0.426	5	0.389	6
Model primarne proizvodnje	28	93	0.293	3	0.237	3
Fizika	31	49	0.476	6	0.238	6

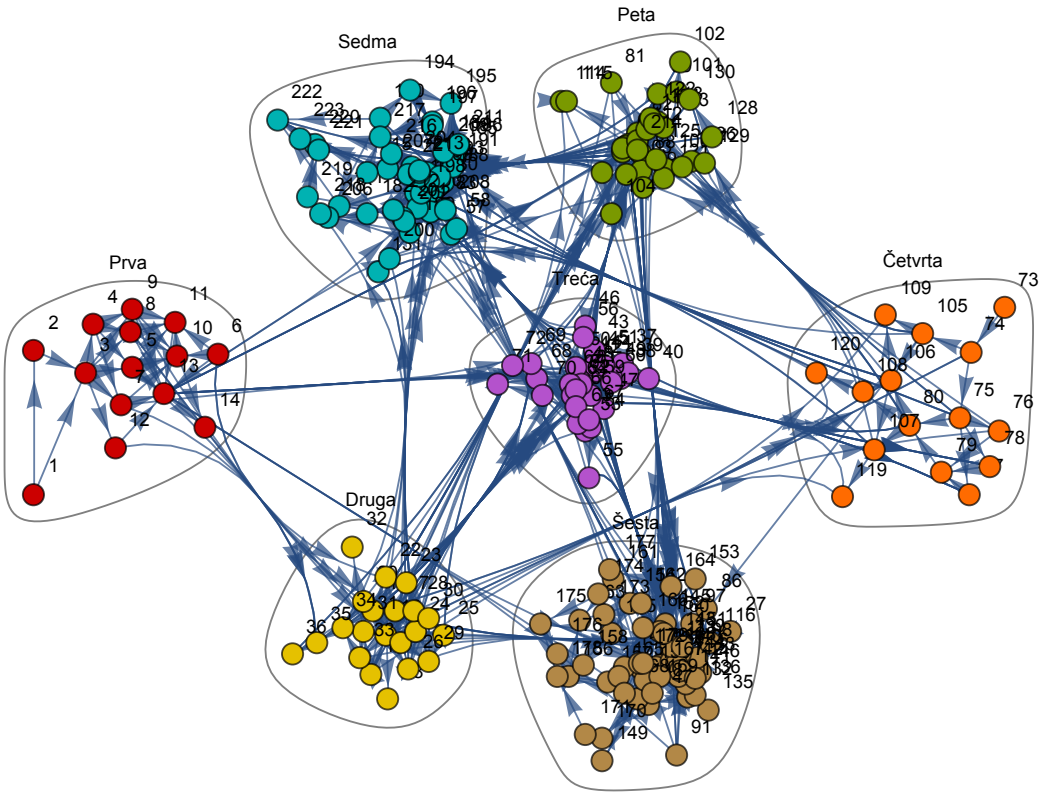
Tablica 4.4: **Usporedba rezultata algoritama za detektiranje zajednica primijenjih na kurikulne mreže.** Prikazani su osnovni rezultati dobiveni pokretanjem LPAm+ i OLPAm+ algoritama 100 puta te Girvan–Newman algoritma za svaku od kurikulnih mreža. *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, Q_d je vrijednost modularnosti dane formulom (4.6) izračunata za podjelu dobivenu danim algoritmom, N_c je broj zajednica koji daje maksimalnu vrijednost modularnosti Q .

			$LPAm+$		GN		$OLPAm+$		$OLPAm+(vs)$	
	n	m	Q_d	N_c	Q_d	N_c	Q_d	N_c	Q_d	N_c
Skup $\mathbb{Q}$	47	254	0.376	4	0.367	4	0.377	4	0.377	4
Elementarne funkcije	84	502	0.361	4	0.223	4	0.354	4	0.337	5
Integral	223	655	0.567	7	0.542	12	0.468	7	0.470	10
Obrada podataka	54	197	0.477	4	0.438	4	0.426	5	0.426	5
Model prim. proizvodnje	28	93	0.297	3	0.099	5	0.293	3	0.293	3
Fizika	31	49	0.457	7	0.377	8	0.476	6	0.467	5

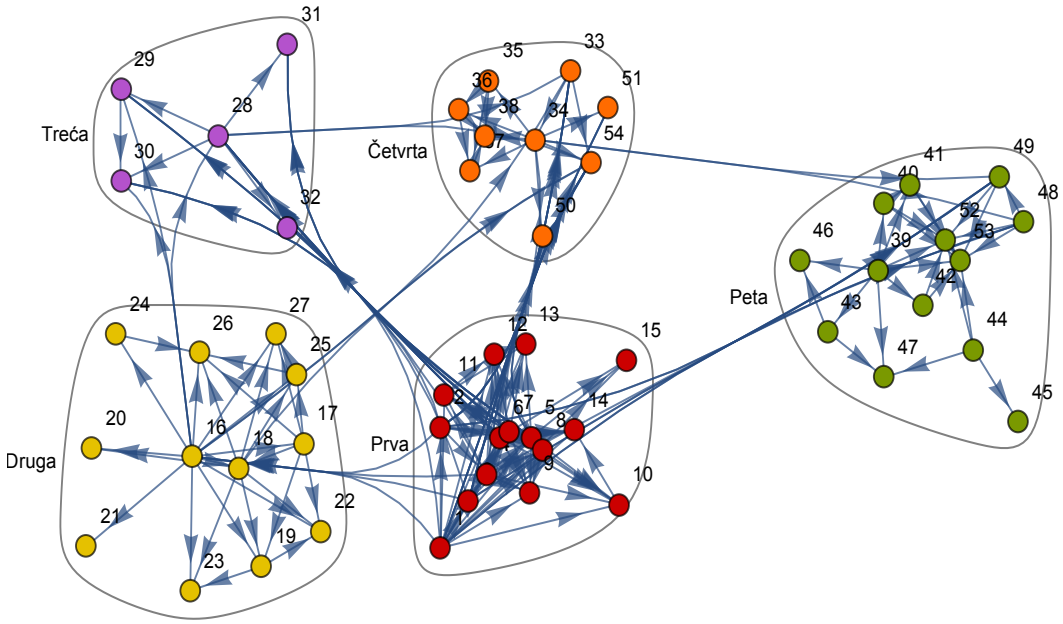
Na sljedećim slikama prikazane su podjele na zajednice dobivene primjenom OLPAm+ algoritma za detektiranje zajednica kurikulne mreže. Sve slike napravljene su u programu *Wolfram Mathematica 2017.*



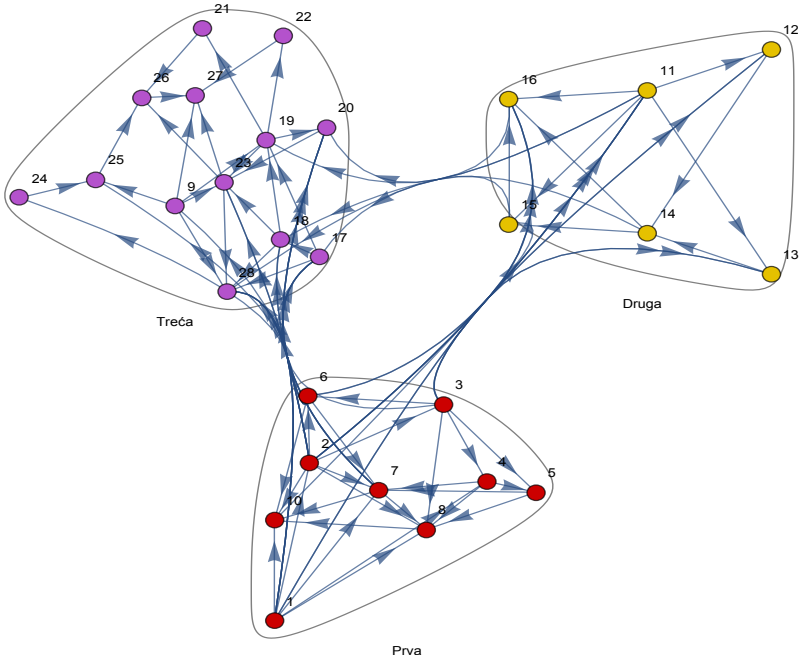
Slika 4.4: Podjela kurikulne mreže *Elementarne funkcije* na zajednice dobivene Greedy OLPAm+ algoritmom. Svaka zajednica označena je labelom. Dobivena podjela zadovoljava uvjet o poretku zajednica, odnosno usmjereni bridovi iz zajednice s manjom labelom mogu pokazivati samo na vrhove u zajednici s većom labelom.



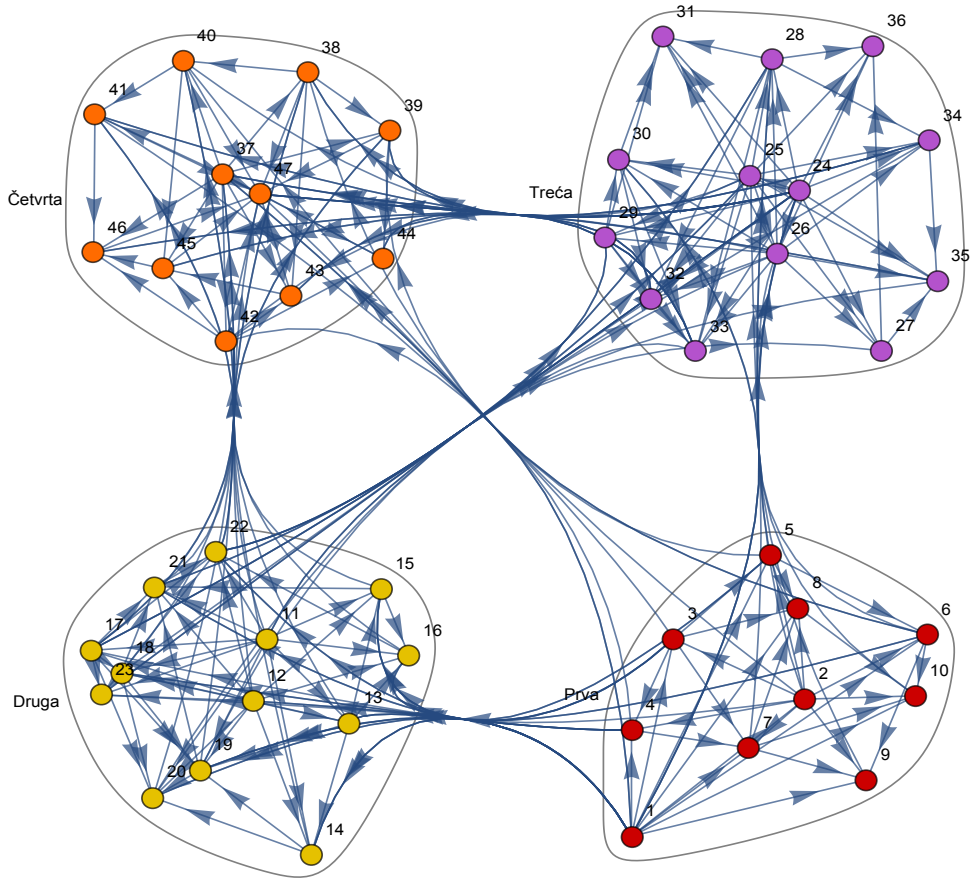
Slika 4.5: Podjela kurikulne mreže *Integral* na zajednice dobivene Greedy OLPAm+ algoritmom. Svaka zajednica označena je labelom. Dobivena podjela zadovoljava uvjet o poretku zajednica, odnosno usmjereni bridovi iz zajednice s manjom labelom mogu pokazivati samo na vrhove u zajednici s većom labelom.



Slika 4.6: Podjela kurikulne mreže *Obrada podataka* na zajednice dobivene Greedy OLPAm+ algoritmom. Svaka zajednica označena je labelom. Dobivena podjela zadovoljava uvjet o poretku zajednica, odnosno usmjereni bridovi iz zajednice s manjom labelom mogu pokazivati samo na vrhove u zajednici s većom labelom.



Slika 4.7: Podjela kurikulne mreže *Model primarne proizvodnje* na zajednice dobivene Greedy OLPAm+ algoritmom. Svaka zajednica označena je labelom. Dobivena podjela zadovoljava uvjet o poretku zajednica, odnosno usmjereni bridovi iz zajednice s manjom labelom mogu pokazivati samo na vrhove u zajednici s većom labelom.



Slika 4.8: Podjela kurikulne mreže *Skup* $\mathbb{Q}$ na zajednice dobivene Greedy OLPAm+ algoritmom. Svaka zajednica označena je labelom. Dobivena podjela zadovoljava uvjet o poretku zajednica, odnosno usmjereni bridovi iz zajednice s manjom labelom mogu pokazivati samo na vrhove u zajednici s većom labelom.

Predloženi OLPAm+ algoritam je prvi algoritam za detektiranje zajednica u kurikulnim mrežama. Temelji se na propagaciji labela i maksimizaciji modularnosti Q_d definirane za usmjerene mreže. Algoritam uzima u obzir poredak vrhova i zahtijeva valjani poredak dobijenih zajednica. Kada proces propagacije labela kroz mrežu zapne u lokalnom maksimumu modularnosti, spajaju se zajednice čije spajanje rezultira maksimalnim povećanjem modularnosti. Predložene su dvije verzije: verzija gdje se spaja samo jedan par zajednica i verzija s višestrukim spajanjem parova zajednica. Obje verzije su vremenski i računalno vrlo efikasne i daju slične rezultate u smislu modularnosti. U usporedbi s podjelom na zajednice koju su predložili autori pojedinih mreža, vrijednosti modularnosti za podjelu dobivenu pomoću algoritma su veće nego vrijednosti za podjele predložene od strane autora mreža. Budući nije moguće napraviti usporedbu s drugim sličnim algoritmima, usporedba rezultata s dostupnim algoritmima za detektiranje zajednica u usmjerenim mrežama navodi na zaključak da algoritam dobro radi i daje kvalitetne podjele na zajednice. Uz kurikulne mreže, algoritam je moguće primijeniti i na sve usmjerene mreže u kojima je važan poredak vrhova.

4.4.3 Optimalni redoslijed zajednica

Konačno, predstavljamo algoritam u kojem se rekurzivnim smještanjem vrhova u odgovarajuće zajednice dobiva optimalna podjela mreže na zajednice, odnosno edukacijske jedinice se dijele u veće cjeline koje se mogu predavati uzastopno. Pretpostavimo da kurikulna mreža ima n vrhova i m usmjernih bridova. Algoritam funkcionira na sljedeći način. U početku smatramo da svaki vrh pripada u zasebnu zajednicu. Počevši od posljednjeg vrha, u svakom koraku razmatramo smještanje vrha u odnosu na zajednice koje su dobivene kao najbolja rješenja u prethodnim koracima. Uvodimo sljedeće oznake: neka je r_k najbolje rješenje dobiveno u k–tom koraku algoritma (prilikom smještanja vrha k) i neka je z_{ij} zajednica koja se sastoji

od vrhova $i, j \in V(G)$. Algoritam kreće od posljednjeg vrha (onog s najvećom labelom n). Vrijedi $r_n = z_n$. Zatim promatramo sljedeći vrh s labelom $n-1$. Za njega razmatramo hoće li se modularnost povećati više ako ga spojimo s najboljim rješenjem iz prethodnog slučaja ili ako ostanu razdvojeni u zasebne zajednice. Preciznije, razmatramo slučajeve $z_{(n-1)n}$ i $z_{(n-1)} + r_n$ (znak "+" označava da se radi o dvije zasebne zajednice). U sljedećem koraku razmatramo vrh $n-2$ i povećanje modularnosti za slučajeve $z_{(n-2)(n-1)n}$, $z_{(n-2)(n-1)} + r_n$ i $z_{(n-2)} + r_{(n-1)}$, gdje je $r_{(n-1)}$ najbolje rješenje iz prethodnog koraka. Općenito, za vrh k razmatramo slučajeve:

- $z_{k(k+1)\ldots n}$
- $z_{k(k+1)\ldots(n-1)} + r_n$
- $z_{k(k+1)\ldots(n-2)} + r_{(n-1)}$

  $\vdots$
- $z_k + r_{(k+1)}$

U svakom koraku odabiremo najbolje rješenje koje daje najveće povećanje modularnosti. Algoritam završava kada su svi vrhovi smješteni u odgovarajuće zajednice. Preciznije, svakom vrhu $i \in V(G)$ dodijeljuje se jedinstvena numerička labela l_i tako da je $l_i = p(i)$. Labela označava zajednicu kojoj pojedini vrh pripada, pa u početku, pošto su sve labele različite, svaki vrh pripada u zasebnu zajednicu. Za posljednji vrh (onaj s najvećom labelom) vrijedi da je $r_n = z_n$ i vrijednost modularnosti se ne mijenja jer je po pretpostavci vrh n zasebna zajednica. Vrijedi $dQ(nn) = 0$. Za sljedeći vrh $n-1$ razmatraju se mogućnosti $z_{(n-1)} + r_n$ (vrhovi n i $n-1$ su u zasebnim zajednicama) i $z_{n(n-1)}$ (vrhovi n i $n-1$ su u istoj zajednici). U prvom slučaju vrh $n-1$ zadržava svoju labelu i $dQ(n(n-1)) = 0$ jer smo u početku pretpostavili da je svaki vrh zasebna zajednica pa nije došlo do promjene. U drugom slučaju, pošto vrhovi n i $n-1$ moraju pripadati istoj zajednici, vrh

$n-1$ mijenja svoju postojeću labelu $l_{(n-1)}$ za labelu vrha n, odnosno $l_{(n-1)} = l_n$. Promjena modularnosti izazvana smještanjem vrhova u istu zajednicu, tj. promjenom labele vrha $n-1$ računa se po formuli (4.10). Općenito, dodavanje vrha k u neku zajednicu odvija se promjenom labele vrha k na labelu koju imaju vrhovi u toj zajednici. Ako vrh k ostaje u zasebnoj zajednici, ne dolazi do promjene labele. Ako ne dođe do promjene labele onda je $dQ = 0$, a ako vrh mijenja svoju labelu (tj. dodaje se u neku zajednicu) promjenu modularnosti dQ lako je izračunati pomoću formule (4.10). Način na koji smještamo vrhove u zajednice osigurava da je zadovoljen uvjet valjanog poretka zajednica i da je dobiveno rješenje optimalno budući u svakom koraku promatramo najbolja rješenja iz prethodnih slučajeva. Pseudokod je prikazan u Algoritmu 7.

Algorithm 7 Algoritam za optimalni redoslijed zajednica **(AORZ)**

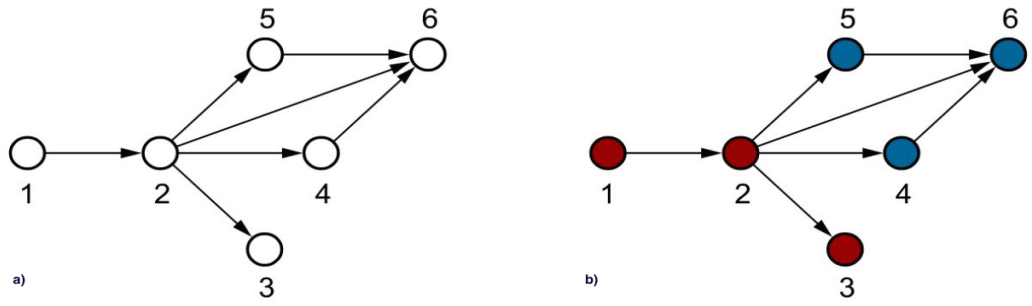
1: svakom vrhu dodijeli jedinstvenu numeričku labelu $l_i \in \{1, 2, ..., n\}$
2: poredaj vrhove u uzlazni poredak
3: **while** postoje vrhovi koji još nisu razmotreni **do**
4: **for** svaki vrh k u uzlaznom poreku počevši od zadnjeg **do**
5: izračunaj promjenu modularnosti za svaki od slučaja $z_{k(k+1)...n}$, $z_{k(k+1)...(n-1)} + r_n$, $z_{k(k+1)...(n-2)} + r_{(n-1)}$, ... , $z_k + r_{(k+1)}$
6: odredi optimalno rješenje r_k
7: smjesti vrh k u odgovarajuću zajednicu u skladu s dobivenim rješenjem
8: **end for**
9: **end while**

Što se tiče računale složenosti algoritma, za smjestiti vrh k u odgovarajuću zajednicu potrebno je razmotriti $n - k + 1$ slučajeva. Za svaki od tih slučajeva potrebno je izračunati promjenu modularnosti koja se računa pomoću formule (4.10) u kojoj je potrebno proći sve susjede onog vrha za koji računamo promjenu. Označimo sa $d_k = d^{in}(k) + d^{out}(k)$. Ukupno je potrebno $(n-k+1)d_k$ operacija za pravilno smještanje vrha k. Sumiranjem po svim vrhovima dobijemo:

$$\sum_{k=1}^{n}(n-k+1)d_k = nd_1 + (n-1)d_2 + (n-2)d_3 + ... + 2d_{n-1} + d_n$$
$$= (d_1 + d_2 + d_3 + ... + d_n) + (d_1 + d_2 + d_3 + ... + d_{(n-1)}) + ... + (d_1 + d_2) + d_1$$
$$\leq 2m + 2m + ... + 2m \leq 2mn$$

Iz prethodnog razmatranja slijedi da je ukupna složenost algoritma jednaka $O(nm)$.

Na Slici 4.9 prikazan je jednostavni primjer kurikulne mreže i podjele na zajednice dobivene pomoću ovog algoritma. Promotrimo na primjeru sa slike kako funkcionira algoritam.



Slika 4.9: **Djelovanje Algoritma za optimalni redoslijed zajednica na primjeru jedne kurikulne mreže. a)** Jednostavni primjer kurikulne mreže s $n = 6$ vrhova i $m = 7$ usmjerenih bridova. **b)** Podjela na 2 uzastopne zajednice dobivena primjenom Algoritma za optimalni redoslijed zajednica.

$i = 6$	Z_6	$[6] \rightarrow r_6$
$i = 5$	Z_{56}	$[5\ 6] \rightarrow r_5$
	$Z_5 + r_5$	$[5]\ [6]$
$i = 4$	Z_{456}	$[4\ 5\ 6] \rightarrow r_4$
	$Z_{45} + r_5$	$[4\ 5][6]$
	$Z_4 + r_5$	$[4]\ [5\ 6]$
$i = 3$	Z_{3456}	$[3\ 4\ 5\ 6]$
	$Z_{345} + r_6$	$[3\ 4\ 5]\ [6]$
	$Z_{34} + r_5$	$[3\ 4]\ [5\ 6]$
	$Z_3 + r_4$	$[3]\ [4\ 5\ 6] \rightarrow r_3$
$i = 2$	Z_{23456}	$[2\ 3\ 4\ 5\ 6]$
	$Z_{2345} + r_6$	$[2\ 3\ 4\ 5]\ [6]$
	$Z_{234} + r_5$	$[2\ 3\ 4]\ [5\ 6]$
	$Z_{23} + r_4$	$[2\ 3]\ [4\ 5\ 6] \rightarrow r_2$
	$Z_2 + r_2$	$[2]\ [3]\ [4\ 5\ 6]$
$i = 1$	Z_{123456}	$[1\ 2\ 3\ 4\ 5\ 6]$
	$Z_{12345} + r_6$	$[1\ 2\ 3\ 4\ 5]\ [6]$
	$Z_{1234} + r_5$	$[1\ 2\ 3\ 4]\ [5\ 6]$
	$Z_{123} + r_4$	$[1\ 2\ 3]\ [4\ 5\ 6] \rightarrow r_1$
	$Z_{12} + r_3$	$[1\ 2]\ [3]\ [4\ 5\ 6]$
	$Z_1 + r_2$	$[1]\ [2\ 3]\ [4\ 5\ 6]$

Algoritam je testiran na kurikulnim mrežama iz Poglavlja 3, a dobiveni rezultati uspoređeni su s rezultatima dobivenim analiziranjem podjele na zajednice koju su odredili autori pojedinih kurikulnih mreža. Usporedba rezultata prikazana je u Tablici 4.5.

Tablica 4.5: **Usporedba rezultata dobivenih pomoću Algoritma za optimalni redoslijed zajednica s rezultatima predloženim od strane stručnjaka koji su sastavljali kurikulne mreže.** Prikazani su rezultati dobiveni primjenom Algoritma za optimalni redoslijed zajednica (AORZ) na kurikulne mreže iz Poglavlja 3 i rezultati dobiveni za podjele koje su predložili kreatori kurikulnih mreža. *Oznake*: n je broj vrhova, m je broj usmjerenih bridova, N_c je broj zajednica, Q_d je vrijednost modularnosti dobivena za predloženu podjelu na zajednice.

			Stručnjak		AORZ	
	n	m	Q_d	N_c	Q_d	N_c
Skup $\mathbb{Q}$	47	254	0.311	5	0.377	4
Elementarne funkcije	84	502	0.239	6	0.286	8
Integral	223	655	0.455	10	0.484	10
Obrada pod.	54	197	0.389	6	0.430	6
Model primarne proizvodnje	28	93	0.237	3	0.259	3
Fizika	31	49	0.238	6	0.375	4

Iako su vrijednosti modularnosti dobivene pomoću Algoritma za optimalni redoslijed zajednica veće nego za podjele na zajednice dane od strane stručnjaka, u nastavku smo za svaku kurikulnu mrežu zasebno usporedili dobivene podjele. Algoritam daje podjele dobivene na temelju mjera kao što su stupanj vrha i vrijednost modularnosti, no ipak je potrebno da je dobivena podjela valjana u pedagoškom i metodičkom smislu.

Kurikulna mreža ***Skup*** $\mathbb{Q}$ od strane stručnjaka podijeljena je na 5 zajednica kako slijedi: vrhovi $1-16$ su u zajednici 1, vrhovi $17-23$ su u zajednici 2, vrhovi $24-36$ su u zajednici 3, vrhovi $37-41$ su u zajednici 4, vrhovi $42-47$ su u zajednici 5. Podjela dobivena pomoću Algoritma je na 4 zajednice: vrhovi $1-10$ su u zajednici 1, vrhovi $11-23$ su u zajednici 2, vrhovi $24-36$ su u zajednici 3, vrhovi $37-47$ su u zajednici 4. Odmah se vidi da je Algoritam spojio zajednice 4 i 5 u jednu zajednicu. Zajednica 3 je ista, a zajednice 1 i 2 su drugačije podijeljenje. U podjeli stručnjaka pojmovi vezani za prirodne brojeve, nula i negativni brojevi u zajednici 1 dok su cijeli brojevi u zajednici 2. U podjeli dobivenoj pomoću Algoritma u zajednici 1 su pojmovi vezani za prirodne brojeve, dok su nula, negativni i cijeli brojevi stavljeni u zajednicu 2. Autor mreže (profesor matematike u srednjoj školi) smatra da je podjela dobivena algoritmom i više nego valjana te da na pravilan način dijeli cjelinu 1.

Kurikulna mreža ***Elementarne funkcije*** od strane stručnjaka podijeljena je na 6 zajednica kako slijedi: vrhovi $1-10$ su u zajednici 1, vrhovi $11-35$ su u zajednici 2, vrhovi $36-49$ su u zajednici 3, vrhovi $50-57$ su u zajednici 4, vrhovi $58-81$ su u zajednici 5, vrhovi $82-84$ su u zajednici 6. Podjela dobivena pomoću Algoritma je na 8 zajednica: vrhovi $1-10$ su u zajednici 1, vrhovi $11-33$ su u zajednici 2, vrh 34 je u zajednici 3, vrh 35 je u zajednici 4, vrhovi $36-49$ su u zajednici 5, vrhovi $50-57$ su u zajednici 6, vrhovi $58-71$ su u zajednici 7, vrhovi $72-84$ su u zajednici 8. Podjele se poklapaju u 3 zajednice, a autro mreža

(nastavnik matematike i informatike u osnovnoj i srednjoj školi) podjela dobivena pomoću algoritma je zajednice 5 i 6 podijelila na prikladan i jako smislen način. Podjela zajednica 7 i 8 na vrhu 71 (*osnovne elementarne funkcije*) je odlična granica što se tiče logičke i metodičke podjele.

Kurikulna mreža ***Fizika*** od strane stručnjaka podijeljena je na 6 zajednica kako slijedi: vrhovi $1-5$ su u zajednici 1, vrhovi $6-14$ su u zajednici 2, vrhovi $15-22$ su u zajednici 3, vrhovi $23-25$ su u zajednici 4, vrhovi $26-29$ su u zajednici 5, vrhovi $30-31$ su u zajednici 6. Podjela dobivena pomoću Algoritma je na 3 zajednice: vrhovi $1-16$ su u zajednici 1, vrhovi $17-27$ su u zajednici 2, vrhovi $128-29$ su u zajednici 3, vrhovi $30-31$ su u zajednici 4. Autor mreže (nastavnik fizike u osnovnoj školi) smatra da je dobivena podjela u redu te da ovisi o preferiranoj veličini zajednica.

Kurikulna mreža ***Model primarne proizvodnje*** od strane stručnjaka podijeljena je na 3 zajednice kako slijedi: vrhovi $1-10$ su u zajednici 1, vrhovi $11-18$ su u zajednici 2, vrhovi $19-28$ su u zajednici 3. Podjela dobivena pomoću Algoritma je na 3 zajednice: vrhovi $1-10$ su u zajednici 1, vrhovi $11-16$ su u zajednici 2, vrhovi $17-28$ su u zajednici 3. Vrhovi 17 i 18 su pomaknuti u susjednu zajednicu. Autor mreže (dr.sc. oceanologije zaposlen na jednom institutu) smatra da je bolje da vrhovi 11 i 17 budu u istoj zajednici. Moguće je sugerirati profesoru da bi se ovako zadano gradivo moglo kvalitetno podijeliti na, recimo, 3 parcijalna ispita s naglaskom da se za posljedni ispit ponove lekcije 17 i 18 za koje nije potpuno jasno u kojoj zajednici bi se trebale nalaziti.

Kurikulna mreža ***Obrada podataka*** od strane stručnjaka podijeljena je na 6 zajednica kako slijedi: vrhovi $1-4$ su u zajednici 1, vrhovi $5-15$ su u zajednici 2, vrhovi $16-27$ su u zajednici 3, vrhovi $28-33$ su u zajednici 4, vrhovi $34-38$ su u zajednici 5, vrhovi $39-54$ su u zajednici 6. Podjela dobivena pomoću Algoritma je na 6 zajednica: vrhovi $1-15$ su u zajednici 1, vrhovi $16-27$ su u zajednici 2,

vrhovi $28-33$ su u zajednici 3, vrhovi $34-38$ su u zajednici 4, vrhovi $39-53$ su u zajednici 5, a vrh 54 je u zajednici 6. Zajednice 1 i 2 su spojenje u jednu zajednicu. Zajednice 3, 4 i 5 su iste, dok je vrh 54 algoritmom izdvojen u posebnu zajednicu. Autor mreže (profesor informatike na fakultetu) smatra da je nepotrebno da je jedan vrh sam u cjelini i da je bolje vrh 54 spojiti u zajednicu 5. Izdvajanje ovog vrha može se shvatiti kao napomena da je ova lekcija učenicima zahtjevna te da bi joj profesor mogao posvetiti više pažnje.

Kurikulna mreža ***Integral*** od strane stručnjaka (profesor matematike na fakultetu) podijeljena je na 10 zajednica kako slijedi: vrhovi $1-14$ su u zajednici 1, vrhovi $15-40$ su u zajednici 2, vrhovi $41-58$ su u zajednici 3, vrhovi $59-86$ su u zajednici 4, vrhovi $87-104$ su u zajednici 5, vrhovi $105-131$ su u zajednici 6, vrhovi $132-179$ su u zajednici 7, vrhovi $180-197$ su u zajednici 8, vrhovi $198-218$ su u zajednici 9, vrhovi $219-223$ su u zajednici 10. Podjela dobivena pomoću Algoritma je na 10 zajednica: vrhovi $1-14$ su u zajednici 1, vrhovi $15-34$ su u zajednici 2, vrhovi $35-40$ su u zajednici 3, vrhovi $41-58$ su u zajednici 4, vrhovi $59-81$ su u zajednici 5, vrhovi $82-98$ su u zajednici 6, vrhovi $99-131$ su u zajednici 7, vrhovi $132-179$ su u zajednici 8, vrhovi $180-218$ su u zajednici 9 i vrhovi $219-223$ su u zajednici 10. Čak 4 zajednice se poklapaju u oba slučaja. Zajednica 2 kod stručnjaka Algoritmom je podijeljena na dvije. Zajednice 8 i 9 kod stručnjaka Algoritmom su spojene u jednu zajednicu. Napomena koja bi se mogla shvatiti iz rezultata Algoritma je da bi se lekcije 99 do 104 koji se tiču jednadžbi mogle izdvojiti u posebnu cjelinu.

Analizom obiju podjela dolazi se do zaključka da Algoritam daje smislene podjele na zajednice. U većini slučajeva postoje zajednice koje su iste ili se spajanjem zajednica iz prvog slučaja dolazi do zajednica iz drugog slučaja. Podjele dobivena Algoritmom većinom imaju i logičko tumačenje te se iz rezultata mogu izvući vrijedni savjeti koji bi mogli pomoći sastavljačima da kvalitetnije sastave i podijele

gradivo. Stručnjaci koji su sastavljali mreže smatraju da su, iako se pojedini vrhovi možda mogu smjestiti na drugačiji način, dobivene podjele smislene i valjane te da treba uzeti u obzir je li cilj podjela na veće ili manje cjeline te po kojem principu se gradivo dijeli. U predloženom algoritmu nema zahtjeva za veličinom ni brojem zajednica iako je tendencija sastavljača da gradivo dijele na približno jednake manje zajednice. Osvrćući se na brojne metodičke i pedagoške zahtjeve koje treba uzeti u obzir prilikom sastavljanja gradiva i podjele edukacijskih jedinica u veće cjeline, smatramo da je najbolje koristiti kombinaciju znanja stručnjaka s rezultatima dobivenim pomoću algoritama.

Poglavlje 5

Zaključak

U ovoj disertaciji izloženi su rezultati istraživanja iz područja mrežnih deskriptora i kurikulnih mreža, te je predložen računalni algoritam za detekciju zajednica u kurikulnim mrežama.

U Poglavlju 2 predstavljene su poopćene definicije mrežnih deskriptora transmisije, međupoloženosti, vršne produktivnosti i vršne profitabilnosti. Prilikom toga, uzeta je u obzir pretpostavka da vrhovi u mreži koji se nalaze na manjim udaljenostima komuniciraju više nego vrhovi na većim udaljenostima. U prvom slučaju količina komunkacije utežena je s $d(u,v)^{\lambda}$ gdje je $\lambda < 0$, a u drugom slučaju s $\lambda^{d(u,v)}$, pri čemu je $d(u,v)$ udaljenost dva vrha u i v, a $\lambda \in \langle 0,1\rangle$. Izložen je niz teorema o gornjim i donjim ogradama za vrijednosti poopćenih deskriptora te o grafovima u kojima se te vrijednosti postižu. U poglavlju 2.1. dan je pregled definicija i rezultata za generalizirane mrežne descriptore (utežene s $d(u,v)^{\lambda}$). Osnovne definicije deskriptora dane su sa

$$t_\lambda(u) = \sum_{v \in V \setminus \{u\}} d(u,v) \cdot d(u,v)^\lambda = \sum_{v \in V \setminus \{u\}} d(u,v)^{1+\lambda},$$

$$c_\lambda(u) = \sum_{v \in [u]} \sum_{\{k,l\} \in \binom{V}{2}} \frac{s_{uv}^{kl}}{s^{kl}} \cdot d(k,l)^\lambda,$$

$$N_\lambda(u) = \frac{c_\lambda(u)}{t_\lambda(u)},$$

$$\nu_\lambda(u) = c_\lambda(u) - t_\lambda(u).$$

Proučene su njihove minimalne i maksimalne vrijednosti, te su istražene gornje i donje ograde tih vrijednosti.

$$mN_\lambda(G) = \min\{N_\lambda(u) : u \in V\},$$

$$MN_\lambda(G) = \max\{N_\lambda(u) : u \in V\},$$

$$m\nu_\lambda(G) = \min\{\nu_\lambda(u) : u \in V\},$$

$$M\nu_\lambda(G) = \max\{\nu_\lambda(u) : u \in V\}.$$

Rezultati se mogu pronaći u Tablicama 5.1 i 5.2

Tablica 5.1: Rezultati za $d(u,v)^\lambda$, $\lambda \in \langle -\infty, -1\rangle$, pri čemu je n broj vrhova.

		Donja ograda	**Gornja ograda**
mN_λ	ekstremalni graf (vrh)	metlica (početak)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	$\min\limits_{2\le D\le n-1} \frac{D^\lambda + \frac{1}{n-D}\sum\limits_{i=1}^{D-1} i^\lambda}{D^{1+\lambda} + \frac{1}{n-D}\sum\limits_{i=1}^{D-1} i^{1+\lambda}}$	1
MN_λ	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)	zvijezda (centar)
	granična vrijednost	1	$2^\lambda(n-2)+1$
$m\nu_\lambda$	ekstremalni graf (vrh)	zvijezda (list)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	$-(n-2)2^\lambda$	0
$M\nu_\lambda$	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)	zvijezda (centar)
	granična vrijednost	0	$(n-1)(n-2)2^\lambda$

Tablica 5.2: Rezultati za $d(u,v)^{\lambda}$, $\lambda \in \langle -1, 0\rangle$, pri čemu je n broj vrhova.

		Donja ograda	**Gornja ograda**
mN_{λ}	ekstremalni graf (vrh)	put (kraj)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	$\frac{\sum_{i=1}^{n-1} i^{\lambda}}{\sum_{i=1}^{n-1} i^{1+\lambda}}$	1
MN_{λ}	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)	zvijezda (centar)
	granična vrijednost	1	$2^{\lambda}(n-2)+1$
$m\nu_{\lambda}$	ekstremalni graf (vrh)	put (kraj puta)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	$\sum_{i=1}^{n-1}(i^{\lambda} - i^{\lambda+1})$	0
$M\nu_{\lambda}$	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)	zvijezda (centar)
	granična vrijednost	0	$(n-1)(n-2)2^{\lambda}$

U Poglavlju 2.2 predstavljeni su eksponencijalni mrežni deskriptori u kojima je količina komunikacije utežena s $\lambda^{d(u,v)}$, pri čemu je $d(u,v)$ udaljenost dva vrha u i v, a $\lambda \in \langle 0,1\rangle$. Osnovne definicije zadane su sa

$$t_\lambda^e(u) = \sum_{v \in V\setminus\{u\}} d(u,v)\cdot\lambda^{d(u,v)},$$

$$c_\lambda^e(u) = \sum_{v\in[u]}\sum_{\{k,l\}\in\binom{V}{2}} \frac{s_{uv}^{kl}}{s^{kl}}\cdot\lambda^{d(u,v)},$$

$$N_\lambda^e(u) = \frac{c_\lambda^e(u)}{t_\lambda^e(u)},$$

$$\nu_\lambda^e(u) = c_\lambda^e(u) - t_\lambda^e(u).$$

Analogno kao i prije, definirane su minimalne i maksimalne vrijednosti i proučavane gornje i donje ograde tih vrijednosti:

$$mt_\lambda^e(G) = \min\left\{t_\lambda^e(u) : u\in V\right\},$$

$$Mt_\lambda^e(G) = \max\left\{t_\lambda^e(u) : u\in V\right\},$$

$$mc_\lambda^e(G) = \min\left\{c_\lambda^e(u) : u\in V\right\},$$

$$Mc_\lambda^e(G) = \max\left\{c_\lambda^e(u) : u\in V\right\},$$

$$mN_\lambda^e(G) = \min\left\{N_\lambda^e(u) : u\in V\right\},$$

$$MN_\lambda^e(G) = \max\left\{N_\lambda^e(u) : u\in V\right\},$$

$$m\nu_\lambda^e(G) = \min\left\{\nu_\lambda^e(u) : u\in V\right\},$$

$$M\nu_\lambda^e(G) = \max\left\{\nu_\lambda^e(u) : u\in V\right\}.$$

Rezultati se mogu pronaći u Tablicama 5.3 i 5.4.

Tablica 5.3: Rezultati za $\lambda^{d(u,v)}$, $\lambda \in \langle 0,1\rangle$, pri čemu je n broj vrhova.

		Donja ograda
mt^e_λ	ekstremalni graf (vrh)	metlica (početak)
	granična vrijednost	$A_n^{(*)}$
Mt^e_λ	ekstremalni graf (vrh)	*otvoren problem*
	granična vrijednost	–
mc^e_λ	ekstremalni graf (vrh)	put (kraj)
	granična vrijednost	$\frac{\lambda^D-\lambda}{\lambda-1}$
Mc^e_λ	ekstremalni graf (vrh)	*otvoren problem*
	granična vrijednost	–
mN^e_λ	ekstremalni graf (vrh)	metlica (početak)
	granična vrijednost	$B_n^{(**)}$
MN^e_λ	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	1
$m\nu^e_\lambda$	ekstremalni graf (vrh)	metlica (početak)
	granična vrijednost	$C_n^{(***)}$
$M\nu^e_\lambda$	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	0

$$^*A_n = \min_{1\leqslant D\leqslant n-1} \frac{\lambda\left[1-(D+1)\lambda^D+D\lambda^{D+1}\right]}{(\lambda-1)^2} + (n-D-1)D\cdot\lambda^D,$$

$$^{**}B_n = \min_{1\leqslant D\leqslant n-1} \frac{\lambda^D+\frac{1}{n-D}\left(\frac{\lambda^D-\lambda}{\lambda-1}\right)}{D\lambda^D+\frac{1}{n-D}\left[\frac{\lambda-D\lambda^D+(D-1)\lambda^{D+1}}{(\lambda-1)^2}\right]},$$

$$^{***}C_n = \min_{1\leqslant D\leqslant n-1} \frac{\lambda\left[D\lambda^D-\lambda-(D-1)\lambda^{D+1}\right]}{(\lambda-1)^2} + (n-D-1)(\lambda^D - D\cdot\lambda^D)$$

Tablica 5.4: Rezultati za $\lambda^{d(u,v)}$, $\lambda \in \langle 0, 1\rangle$, pri čemu je n broj vrhova.

		Gornja ograda
mt^e_λ	ekstremalni graf (vrh)	potpuni graf (bilo koji vrh)$^{(*)}$
	granična vrijednost	$(n-1)\lambda$
Mt^e_λ	ekstremalni graf (vrh)	metlica (početak)
	granična vrijednost	$D_n^{(**)}$
mc^e_λ	ekstremalni graf (vrh)	potpuni graf (bilo koji vrh)$^{(*)}$
	granična vrijednost	$(n-1)\lambda$
Mc^e_λ	ekstremalni graf (vrh)	zvijezda (centar)
	granična vrijednost	$(n-1)\left[\lambda + \frac{1}{2}(n-2)\lambda^2\right]$
mN^e_λ	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	1
MN^e_λ	ekstremalni graf (vrh)	zvijezda (centar)
	granična vrijednost	$\frac{1}{2}(n-2)\lambda + 1$
$m\nu^e_\lambda$	ekstremalni graf (vrh)	vršno tranzitivni graf (bilo koji vrh)
	granična vrijednost	0
$M\nu^e_\lambda$	ekstremalni graf (vrh)	zvijezda (centar)
	granična vrijednost	$\frac{1}{2}(n-1)(n-2)\lambda^2$

* Dokazano za $\lambda \in \langle 0, \frac{1}{2}\rangle$

$^{**}D_n = \max\limits_{1 \leqslant D \leqslant n-1} \frac{\lambda\left[1-(D+1)\lambda^D + D\lambda^{D+1}\right]}{(\lambda-1)^2} + (n-D-1)D \cdot \lambda^D$

U Poglavlju 3 bavili smo se kurikulnim mrežama. Kurikulna mreža je jednstavni usmjereni graf u kojem vrhovi predstavljaju eduacijske jedinice, a brid između dvaju vrhova označava da je znanje jedne edukacijske jedinice potrebno za učenje druge. Definirali smo mjere složenosti za pojedinu edukacijsku jedinicu:

$$s_{p,G}(v) = \sum_{uv \in E(G)} [p(v) - p(u)]$$

$$a_{p,G}(v) = \begin{cases} \frac{s_{p,G}(v)}{d_G^-(v)} & \text{if } d_G^-(v) > 0, \\ 0 & \text{if } d_G^-(v) = 0 \end{cases}$$

$$m_{p,G}(v) = \max_{uv \in E(G)} \{p(v) - p(u)\}$$

Zatim smo promatrali složenost ukupnog nastavnog plana s tri različita pristupa. U prvom pristupu smatrali smo da je složenost nastavnog plana jednaka sumi složenosti nastavih jedinica od kojih se sastoji (ili do na linearnu konstantu prosječna složenost edukacijskih jedinica) pa smo definirali sljedeće mjere:

$$s_p^1(G) = \frac{\sum_{v \in V(G)} s_{p,G}(v)}{card\,(V(G))}$$

$$a_p^1(G) = \frac{\sum_{v \in V(G)} a_{p,G}(v)}{card\,(V(G))}$$

$$m_p^1(G) = \frac{\sum_{v \in V(G)} m_{p,G}(v)}{card\,(V(G))}$$

U drugom pristupu smatrali smo da je složenost ukupnog nastavnog plana jednaka složenosti najkompleksnije jedinice:

$$s_p^\infty(G) = \max_{v \in V(G)} \{s_{p,G}(v)\}$$

$$a_p^\infty(G) = \max_{v \in V(G)} \{a_{p,G}(v)\}$$

$$m_p^{\infty}(G) = \max_{v \in V(G)} \{m_{p,G}(v)\}$$

Treći pristup predstavlja kompromis između ova dva ekstremna pristupa pa smo promatrali α - sredinu:

$$s_p^{\alpha}(G) = \left(\frac{\sum\limits_{v \in V(G)} s_{p,G}^{\alpha}(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

$$a_p^{\alpha}(G) = \left(\frac{\sum\limits_{v \in V(G)} a_{p,G}^{\alpha}(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

$$m_p^{\alpha}(G) = \left(\frac{\sum\limits_{v \in V(G)} m_{p,G}^{\alpha}(v)}{card\,(V(G))} \right)^{\frac{1}{\alpha}}$$

Cilj je bio odrediti nastavne planove s najmanjom ili najvećom složenošću te gornje i donje ograde za vrijednosti pojedinih mjera složenosti definirane sa:

$$s^{\alpha}(G) = \min_{p \in P(G)} \left\{ s_p^{\alpha}(G) \right\}$$

$$a^{\alpha}(G) = \min_{p \in P(G)} \left\{ a_p^{\alpha}(G) \right\}$$

$$m^{\alpha}(G) = \min_{p \in P(G)} \left\{ m_p^{\alpha}(G) \right\}$$

za $\alpha \geq 1$.

Promatrali smo dvije vrste grafova: slabo povezane usmjerene grafove koji ne sadrže usmjerene cikluse (grafovi tipa A) i slabo povezane usmjerene grafove koji ne sadrže usmjerene cikluse i koji su tranzitivno zatvoreni (grafovi tipa B). Rezultati se mogu pronaći u Tablicama 5.5 do 5.8.

Tablica 5.5: Rezultati za grafove tipa A i vrijednosti $\alpha \geq 1$, pri čemu je n broj vrhova grafa

		Donja ograda
$s^{\infty}(G)$	ekstremalni graf	$P_n^{(*)}$
	granična vrijednost	1
$a^{\infty}(G)$	ekstremalni graf	$P_n^{(*)}$
	granična vrijednost	1
$m^{\infty}(G)$	ekstremalni graf	$P_n^{(*)}$
	granična vrijednost	1
$s^{\alpha}(G)$	ekstremalni graf	$P_n^{(*)}$
	granična vrijednost	$\left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}}$
$a^{\alpha}(G)$	ekstremalni graf	$T_n^{(*)}$ ili $T_{n,k}^{(**)}$ ili $P_n^{(*)}$
	granična vrijednost	$\min\left\{\frac{n}{2n^{\frac{1}{\alpha}}}, \frac{\alpha}{2(\alpha-1)}\left[\frac{(n-1)(\alpha-1)}{n}\right]^{\frac{1}{\alpha}}, \left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}}\right\}$
$m^{\alpha}(G)$	ekstremalni graf	$P_n^{(*)}$
	granična vrijednost	$\left(\frac{n-1}{n}\right)^{\frac{1}{\alpha}}$

* Definirano u Poglavlju 3.3

** Graf $T_{n,k}$ prikazan je na Slici 3.3

Tablica 5.6: Rezultati za grafove tipa A i vrijednosti $\alpha \geq 1$, pri čemu je n broj vrhova grafa

		Gornja ograda
$s^{\infty}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\sum_{i=1}^{n-1} i$
$a^{\infty}(G)$	ekstremalni graf	$S_n^{(*)}$
	granična vrijednost	$n-1$
$m^{\infty}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$n-1$
$s^{\alpha}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=2}^{n} \left(\sum_{j=1}^{i-1} j \right)^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$
$a^{\alpha}(G)$	ekstremalni graf	$S_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$
$m^{\alpha}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$

*Definirano u Poglavlju 3.3

Tablica 5.7: Rezultati za grafove tipa B i vrijednosti $\alpha \geq 1$, pri čemu je n broj vrhova grafa

		Donja ograda
$s^{\infty}(G)$	ekstremalni graf	$DP_n^{(*)}$
	granična vrijednost	4
$a^{\infty}(G)$	ekstremalni graf	$DP_n^{(*)}$
	granična vrijednost	2
$m^{\infty}(G)$	ekstremalni graf	$DP_n^{(*)}$
	granična vrijednost	3
$s^{\alpha}(G)$	ekstremalni graf	$DP_n^{(*)}$
	granična vrijednost	$\frac{2n-3}{n}^{(**)}$
$a^{\alpha}(G)$	ekstremalni graf	$T_n^{(*)}$
	granična vrijednost	$\frac{1}{2}^{(**)}$
$m^{\alpha}(G)$	ekstremalni graf	$T_n^{(*)}$
	granična vrijednost	$\frac{n-1}{n}^{(**)}$

* Definirano u Poglavlju 3.3

** Dokazano za $\alpha = 1$

Tablica 5.8: Rezultati za grafove tipa B i vrijednosti $\alpha \geq 1$, pri čemu je n broj vrhova grafa

		Gornja ograda
$s^{\infty}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\sum_{i=1}^{n-1} i$
$a^{\infty}(G)$	ekstremalni graf	$S_n^{(*)}$
	granična vrijednost	$n-1$
$m^{\infty}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$n-1$
$s^{\alpha}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=2}^{n} \left(\sum_{j=1}^{i-1} j \right)^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$
$a^{\alpha}(G)$	ekstremalni graf	$S_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$
$m^{\alpha}(G)$	ekstremalni graf	$O_n^{(*)}$
	granična vrijednost	$\left(\frac{\sum_{i=1}^{n-1} i^{\alpha}}{n} \right)^{\frac{1}{\alpha}}$

*Definirano u Poglavlju 3.3

Nadalje, istražili smo neka standardna svojstva, mjere i metrike koje se pojavljuju u kurikulnim mrežama. Uočene su sličnosti u strukturama analiziranih mreža kao i neka zajednička svojstva. Konstruiran je algoritam za optimalnu ekspoziciju edukacijskih jedinica koji daje minimalno složeni poredak jedinica u odnosu na odabranu mjeru složenosti $s_1(G)$. Algoritam je testiran na svim analiziranim kurikulnim mrežama i za svaku od njih daje poboljšanje u vidu umanjene vrijednosti mjere $s_1(G)$. Algoritam je testiran i na tranzitivnim zatvaračima svih kurikulnih mreža gdje također za svaku od njih daje poboljšanje u odnosu na mjeru $s_1(G)$. Zatim su testirane i ostale mjere složenosti $s_\alpha(G)$, $m_\alpha(G)$ i $a_\alpha(G)$ za različite vrijednosti parametra α i za svaku od mjera algoritam daje poboljšanje u odnosu na ulaznu mrežu sastavljenu od strane stručnjaka. Rezultati se mogu pronaći na Slikama 3.15 do 3.19.

U završnom poglavlju bavili smo se problemom detektiranja zajednica u kurikulnim mrežama. Općenito, problem detektiranja zajednica u kompleksnim mrežama je vrlo složen, između ostalog i zato jer ovisi o definiciji pojma zajednica. U usmjerenim mrežama problem je dodatno otežan jer treba uzeti u obzir da se mijenja sematika brida pa je u razmatranje u kvalitetnoj podjeli na zajednice potrebno uzeti u obzir i usmjerenost brida. Napravljen je kratki pregled algoritama i metoda za detektiranje zajednica u usmjerenim i neusmjerenim mrežama. Definirali smo problem detektiranja zajednica u kurikulnim mrežama kao problem grupiranja edukacijskih jedinica u veće cjeline koje se mogu proučavati uzastopno, na primjer, grupiranje nastavnih tema u nastavne cjeline koje se obrađuju u nizu, jedna za drugom. Drugim riječima, ako je dan parcijalni uređaj edukacijskih jedinica $x_1 \prec x_2 \prec ... \prec x_n$ (znak "$\prec$" koristimo u smislu da se edukacijska jedinica x_i obrađuje i uči prije edukacijske jedinice x_j), želimo podijeliti kurikulnu mrežu na zajednice $A_1, A_2, ..., A_k$ koje možemo urediti tako da vrijedi:

ako je $x_i \prec x_j$, $x_i \in A_p$ i $x_j \in A_q$ onda je $A_p \prec A_q$ ili $A_p = A_q$.

Pošto, koliko nam je poznato, nijedan od dostupnih algoritama ne uzima u obzir valjani poredak dobivenih zajednica, razvili smo OLPAm+ algoritam koji se temelji na propagaciji labela i maksimizaciji modularnosti Q_d koja je prilagođena usmjerenim mrežama. Algoritam je testiran na kurikulnim mrežama definiranim u Poglavlju 3. Rezultati se mogu pronaći u Tablici 4.2. Pošto nam nije poznat nijedan algoritam za detektiranje zajednica koji udovoljava zahtjevu za valjanim poretkom zajednica, predloženi algoritam uspoređen je s LPAm+ algoritmom i Girvan–Newmanovim algoritmom da bi se dobio uvid koliko traženi uvjet ima utjecaja na vrijednost modularnosti. Rezultati se mogu pronaći u Tablici 4.4.

Predloženi algoritam je, koliko nam je poznato, prvi algoritam koji radi detektiranje zajednica u kurikulnim mrežama. Dobiveni rezultati su dosta dobri u usporedbi s navedenim LPAm+ i Girvan–Newmanovim algoritmom. Algoritam je za potrebe za koje je razvijen vrlo brz i efikasan.

Konačno, predložen je egzaktni algoritam za određivanje optimalnog redoslijeda zajednica koji rekurzivnim smještanjem vrhova u odgovarajuće zajednice pronalazi najbolje rješenje. Algoritam radi brzo i efikasno, a rezultati dobiveni primjenom na kurikulne mreže i usporedba s rezultatima dobivenim pomoću ostalih spomenutih algoritama dani su u Tablici 4.5.

Bibliografija

[1] Adamic, L.A.; Huberman, B.A., The nature of markets in the World Wide Web, *Q. J. Electronic Commerce* (1999), IEA5, http://dx.doi.org/10.2139/ssrn.166108.

[2] Agarwal, G.; Kempe, D., Modularity-maximizing graph communities via mathematical programming, *The European Physical Journal B*, **66 (3)** (2008), pp. 409–418.

[3] Ahn, Y.-Y.; Bagrow, J.P.; Lehmann, S., Link communities reveal multi-scale complexity in networks, *Nature* **66** (2010), pp. 761–764.

[4] Albert, R.; Jeong, H; Barabási, A.–L., Diameter of the World wide web, *Nature* **401** (1999), pp. 130–131.

[5] Albert, R.; Barabási, A.-L., Statistical mechanics of complex networks, *Rev. Mod. Phys.* **74(1)** (2002), pp. 47–94.

[6] Aldous, D., Spatial transportation networks with transfer costs: Asymptotic optimality of hub and spoke models, *Mathematical Proceedings of the Cambridge Philosophical Society* **145** (2008), 471–487.

[7] Alpacan, T.; Basar, T., *Network Security: A Decision and Game–Theoretic Approach*, Cambridge University Press, New York, 2011.

[8] Antunović, S.; Kokan, T.; Vojković, T.; Vukičević, D., Generalised Network Descriptors, *Glas. Mat.* **48** (2013), pp. 211–230.

[9] Antunović, S.; Kokan, T.; Vojković, T.; Vukičević, D., Exponential Generalised Network Descriptors, poslano u *Glas. Mat.*.

[10] Antunović, S.; Vukičević, D., Curriculum content sequencing using complex networks theory, u pripremi.

[11] Arenas, A.; Duch, J.; Fernández, A.; Gómez, S., Size reduction of complex networks preserving modularity, *New Journal of Physics*, **9**, (2007), DOI:10.1088/1367-2630/9/6/176.

[12] Atkinson, M.D.; Sack, J.-R.; Santoro, N.; Strothotte, T., Min–max heaps and generalized priority queues , *Communications of the ACM* **29 (10)** (1986), pp. 996–1000.

[13] Baavar, J.R.; Maritan, A.; Rinaldo, A., Size and form in efficient transportation networks, *Nature* **399** (1999), pp. 130–132.

[14] Barabási, A.-L.; Jeong, H.; Ravasz, E.; Néda, Z.; Schuberts, A.; Vicsek, T., Evolution of the social network of scientific collaborations, *Physica A* **311** (2002), pp 590–614.

[15] Barber, M. J.; Clark, J. W., Detecting network communities by propagating labels under constraints, *Phys. Rev. E*, **80** (2009), DOI:10.1103/PhysRevE.80.026129.

[16] Barber, M.J., Detecting Hierarchical Communities in Networks: A New Approach, C.C. Bernido et al. (eds.), *Stochastic and Infinite Dimensional Analysis*, Trends in Mathematics, Springer International Publishing, Switzerland, 2016.

[17] Barrow, R.; Milburn, G., *A critical dictionary of educational concepts*, Harvester Wheatsheaf, New York, 1990.

[18] Barthélemy, M.; Barrat, A.; Pastor-Satorras, R.; Vespignani, A., Velocity and hierarchical spread of epidemic outbreaks in scale-free networks, *Phys. Rev. Lett.***92** (2004), DOI: 10.1103/PhysRevLett.92.178701.

[19] Barthélemy, M.; Barrat, A.; Pastor-Satorras, R.; Vespignani, A., Dynamical patterns of epidemic outbreaks in complex heterogeneous networks, *J. Theor. Biol.* **235** (2005),pp. 275–288.

[20] Barthélemy, M., Spatial Networks, *Physics Report* **499 (1–3)** (2011), pp. 1–101.

[21] Bavelas, A., A mathematical model for group structure, *Applied Anthropology* **7(3)** (1948), pp. 16–30.

[22] Beard, D.A.; Liang, S.; Qian, H., Energy Balance for Analysis of Complex Metabolic Networks, *Biophysical Journal* **83** (2002), pp. 79–86.

[23] Bearman, P.S.; Moody, J.; Stovel, K., Chains of affection: The structure of adolescent romantic and sexual networks, *Am. J. Sociol.* **110** (2004), pp. 44–91.

[24] Beauchamp, G., *Basic components of a curriculum theory* In A. Bellack and H. Kliebard (eds.), Curriculum and evaluation (p.22), McCutchan, Berkeley (1977).

[25] Bernard, H.R.; Killworth, P.D.; Evans, M.J.; McCarthy, C.; Shelley, G.A., Studying social relations cross–culturally, *Ethnology* **2** (1988), pp.155–179.

[26] Bianconi, G.; Barabási, A.–L., Bose–Einstein condensation in complex networks, *Phys. Rev. Lett* **86** (2001), pp. 5632–5635.

[27] Bollobás, B., *Modern Graph Theory*, Springer, New York, 1998.

[28] Boguñá, M.; Pastor-Satorras, R.; Díaz-Guilera, A.; Arenas, A., Models of social networks based on social distance attachment, *Phys. Rev. E* **70** (2004), DOI: 10.1103/PhysRevE.70.056122.

[29] Boone, C.; Bussey, H.; Andrews, B.J., Exploring genetic interactions and networks with yeast, *Nature Reviews Genetics* **8** (2007), pp. 437–449.

[30] Borgati, S.P.; Everett, M.G., A graph–theoretical framework for classifying centrality measures, *Social Networks* **28(4)** (2006),pp. 466–484.

[31] Bradde, S.; Caccioli, F.; Dall'Asta, L.; Bianconi, G., Critical Fluctuations in Spatial Complex Networks, *Phys. Rev. Lett* **104** (2010), DOI: 10.1103/PhysRevLett.104.218701.

[32] Brandes, U., A Faster Algorithm for Betweenness Centrality, *Journal of Mathematical Sociology* **25(2)** (2001), pp. 163–177.

[33] Brandes, U.; Delling, D.; Gaertler, M.; Göerke, R.; Hoefer, M.; Nikoloski, Z.; Wagner, D., On Finding Graph Clusterings with Maximum Modularity, *Lecture Notes in Computer Science, vol. 4769*, Springer, pp. 121–132.

[34] Braslavsky, C., *The curriculum*, IBE UNESCO, Geneva, 2003.

[35] Bruner, J.S., *The process of education*, Harvard University Press, Camridge, 1960.

[36] Brusilovsky, P., Adaptive and Intelligent Technologies for Web-based Education, *Kunstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching* **4** (1999), pp. 19–25.

[37] Bullmore, E.; Sporns, O., Complex brain networks: graph theoretical analysis of structural and functional systems, *Nature Reviews Neuroscience* **10** (2009), pp. 186–198.

[38] Caporossi, G.; Paiva, M.; Vukičević, D.; Segatto, M., Centrality and Betweenness: Vertex and Edge decomposition of the Wiener Index, *MATCH Commun. Math. Comput. Chem* **68** (2012), pp. 293–302.

[39] Carvalho, R.; Buzna, L.; Bono, F.; Gutierrez, E.; Just, W.; Arrowsmith, D., Robustness of trans-European gas networks, *Phys. Rev. E* **80** (2009), DOI:https://doi.org/10.1103/PhysRevE.80.016106.

[40] de Castro, R.; Grossman, J.W., Famous trails to Paul Erdős, *Math. Intell* **21** (1999), pp. 51–63.

[41] Chen, C.-M.; Liu, C.-Y.; Chang, M.-H., Personalized curriculum sequencing utilizing modified item response theory for web-based instruction, *Expert Systems with Applications* **30(2)** (2006), pp. 378–396.

[42] Clauset, A.; Newman, M. E. J.; Moore, C., Finding community structure in very large networks, *Phys. Rev. E*, **70** (2004), DOI:https://doi.org/10.1103/PhysRevE.70.066111.

[43] Cohen, R.; Erez, K.; Ben–Avraham, D.; Havlin, S., Resilience of the Internet to random breakdown, *Phys. Rev. Lett* **85** (2000), pp. 4626–4628.

[44] Compeau, P.E.C.; Pevzner, P.A.; Tesler, G., How to apply de Bruijin graphs to genome assembley, *Nature Biotechnology* **29** (2011), pp. 987–991.

[45] Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C., *Introduction to Algorithms*, MIT Press and McGraw-Hill, Cambridge, 2001.

[46] Crucitti, P.; Latora, V.; Porta, S., Centrality measures in spatial networks of urban streets, *Phys. Rev. E* **73** (2006), DOI: 10.1103/PhysRevE.73.036125.

[47] Danon, L.; Duch, J.; Daz-Guilera, A.; Arenas, A., Comparing community structure identification, *J. Stat. Mech.* (2005), DOI:10.1088/1742-5468/2005/09/P09008.

[48] Derenyi, I.; Palla, G.; Vicsek, T., Clique percolation in random networks, *Physical Review Letters* **94** (2005), DOI: 10.1103/PhysRevLett.94.160202.

[49] Dijkstra, E.W., A note on two problems in connexion with graphs, *Numer. Math.* **1** (1959), pp. 269–271.

[50] Dewey,J., *The child and the curriculum*, University of Chicago Press, Phoenix Books, Chicago, 1902.

[51] Dodds, P.S.; Rothman, D.H., Geometry of river networks, *Phys. Rev. E* **63** (2000), DOI: 10.1103/PhysRevE.63.016117.

[52] Donges, J.F.; Zou, Y.; Marwan, N.; Kurths, J., Complex networks in climate dynamics, *Eur. Phys. J. Special Topics* **174** (2009), pp. 157–179.

[53] Du, N.; Wang, B.; Wu, b., Communuity Detection in Complex Networks, *Journal of Computer Science and Technology* **23 (4)** (2008), pp. 672–683.

[54] Dunne, J.A.; Williams, R.J.; Martinez, N.D., Food-web structure and network theory: The role of connectance and size, *Proc. Natl. Acad. Sci. USA* **99** (2002), pp. 12917–12922.

[55] Duch, J.; Arenas, A., Community detection in complex networks using extremal optimization, *Phys. Rev. E* **72** (2005), DOI:https://doi.org/10.1103/PhysRevE.72.027104.

[56] Entringer, R.C.; Jackson, D.E.; Snyder, D.A., Distance in graphs, *Czechoslovak Mathematical Journal* **26 (101)** (1976),pp. 283–296.

[57] Estrada, E.; Hatano, N.; Benzi, M., The physics of communicability in complex networks, *Physics Report* **514 (3)** (2012), pp. 89–119.

[58] Falkowski, T.; Barth, A.; Spiliopoulou, M., DENGRAPH: A Density-based Community Detection Algorithm, *IEEE/WIC/ACM International Conference on Web Intelligence* (2007), pp. 112–115.

[59] Feld, S., Why your friends have more friends than you do, *Am. J. Sociol.* **96** (1991), pp. 1464–1477.

[60] Ferrer i Cancho, R.; Janssen, C.; Solé, R.V., Topology of technology graphs: Small world patterns in electronic circuits, *Phys. Rev. E* **64** (2001), DOI: 10.1103/PhysRevE.64.046119.

[61] Fortunato, S., Community detection in graphs, *Physics Reports* **486 (3-5)** (2010), pp. 75174.

[62] Fortunato, S.; Barth élemy, M., Resolution limit in community detection, *PNAS* **104(1)** (2006), pp. 36–41.

[63] Gagné, R.M., *The conditions of learning* (2nd ed.), Holt, Rinehart & Winston, New York, 1970.

[64] Gastner, M.T. ; Newman, M.E.J., Optimal design of spatial distribution networks, *Phys. Rev. E* **74** (2006), DOI: 10.1103/PhysRevE.74.016117 .

[65] Girvan, M.; Newman, M.E.J., Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA* **99** (2002), pp. 7821–7826.

[66] Goodson, I. F., *Studying curriculum*,Teachers College Press, New York (1994).

[67] Gregory, S., Finding overlapping communities in networks by label propagation, *New Journal of Physics* **12**, (2010), DOI:10.1088/1367-2630/12/10/103018.

[68] Guelzim, N.; Bottani, S.; Bourgine, P.; Képès, F., Topological and causal structure of the yeast transcriptional regulatory network, *Nature Genetics* **31** (2002), pp. 60–63.

[69] Guimerá, R.; Amaral, L.A.N., Functional cartography of complex metabolic networks, *Nature* **433** (2005), pp. 895–900.

[70] Hayward, R.; McDiarmid, C., Average Case Analysis of Heap Building by Repeated Insertion, *J. Algorithms* **12**, pp. 126–153.

[71] Hazewinkel, M., Greedy algorithm in *Encyclopedia of Mathematics*, Springer (2001)

[72] Hethcote, H.W., The mathematics of infectious diseases, *SIAM Rev.* **42** (2000), pp. 599–653.

[73] Hoff, P.; Raftery, A.; Handcock, M., Latent space approaches to social network analysis, *Journal of the American Statistical Association* **97** (2002), pp. 1090–1098.

[74] Holme, P.; Jun Kim, B.; No Yoon, C.; Han, S.K., Attack vulnerability of complex networks, *Phys. Rev. E* **65** (2002), DOI: https://doi.org/10.1103/PhysRevE.65.056109.

[75] Holme, P.; Saramäki, J., Temporal Networks, *Physics Reports* **519 (3)**, (2012), pp. 97–125.

[76] Hu, W., Finding Statistically Significant Communities in Networks with Weighted Label Propagation, *Social Networking* **2** (2013), pp. 138–146.

[77] Hu, Q.; Connolly Knox, C.; Kapucu, N., What have we learned after September 11, 2001? A Network study of the Boston Marathon bombings response, *Public Administration Review* **74(6)** (2014), DOI: 10.1111/puar.12284.

[78] Huberman, B.A., *The Laws of the Web*, MIT Press, Cambridge, 2001.

[79] Huxham, M.; Beaney, S.; Raffaelli, D., Do parasites reduce the chances of triangulation in a real food web?, *Oikos* **76** (1996), pp. 284–300.

[80] Hyoungshick, K.; Anderson, R., Temporal node centrality in complex networks, *Phys. Rev E* **85** (2012), DOI: 10.1103/PhysRevE.85.026107.

[81] Jeong, H.; Mason, S.; Barabási, A.–L.; Oltvai, Z.N., Lethality and centrality in protein networks, *Nature* **411** (2001), pp. 41–42.

[82] Jeong, H.; Tombor, B.; Albert, R.; Oltvai, Z.N.; Barabási, A.–L., The large-scale organization of metabolic networks, *Nature* **407** (2000), pp. 651–654.

[83] Kadesch, R.R., *Problem Solving Across the Disciplines*, Prentice Hall, Indiana, SAD, 1997.

[84] Kahn, A. B., Topological sorting of large networks, *Communications of the ACM* **5 (11)**, (1962), pp. 558–562.

[85] Kelly, A. V., *The curriculum: Theory and practice*, Sage Publications, London, 2004.

[86] Kim, Y.; Son, S.-W.; Jeong, H., LinkRank: Finding communities in directed networks, *Phys. Rev. E* **81** (2010), DOI:10.1103/PhysRevE.81.016103.

[87] Kitsak, M.; Gallos, L.K.; Havlin, S.; Liljeros, F.; Muchnik, L.; Stanley, H.E.; Makse, H.A., Identification of influential spreaders in complex networks, *Nature Physics* **6**, (2010), pp. 888–893.

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

[89] Knuth, D.E.; Szwarcfiter, J.L., A strctured program to generate all topological soring arrangements, *Information Processing Letters* **2 (6)** (1974), pp. 153–157.

[90] Krapivsky, P.L.; Redner, S.A., A statistical physics perspective on Web growth, *Comput. Netw.* **39** (2002), pp. 261–276.

[91] Lancichinetti, A.; Fortunato, S.; Kertész, J., Detecting the overlapping and hierarchical community structure in complex networks, *New Journal of Physics* **11** (2009), DOI : 10.1088/1367-2630/11/3/033015.

[92] Lancichinetti, A.; Fortunato, S.; Radicchi, F., Benchmark graphs for testing community detection algorithms, *Physical Review E* **93** (2008), DOI : 10.1103/PhysRevE.78.046110.

[93] De Leenheer, P.; Angeli, D.; Sontag, E. D., Monotone chemical reaction networks, *J. Math. Chem.* **41** (2007), pp. 295–314.

[94] Leicht, E. A.; Clarkson, G.; Shedden, K.; Newman, M.E.J., Large–scale structure of time evolving citation networks, *Eur. Phys. J. B* **59** (2007), pp. 75–83.

[95] Leicht, E. A.; Newman, M. E. J., Community structure in directed networks,*Phys. Rev. Lett.*, **100** (2008), DOI : 10.1103/PhysRevLett.100.118703.

[96] Leung, . X. Y.; Hui, P.; Lió, P.; Crowcroft, J., Towards real-time community detection in large networks, *Phys. Rev. E* **79** (2009), DOI : 10.1103/PhysRevE.79.066107.

[97] Liu, X.; Murata, T.; Advanced modularity-specialized label propagation algorithm for detecting communities in networks, *Physica A* **389** (2010), pp. 1493–1500.

[98] Liu, Y.; Wang, Q.; Yao, Q.; Liu, Y., E-mail Community Detection Using Artificial Ant Colony Clustering, *Advances in Web and Network Technologies and Information Management* Springer, Berlin / Heidelberg (2007), pp. 287-298.

[99] Longstreet, W. S.; Shane, H. G., *Curriculum for a new millennium*, Allyn and Bacon, Boston, 1993.

[100] Lou, H.; Li, S.; Zhao, Y., Detecting community structure using label propagation with weighted coherent neighborhood propinquity, *Physica A: Statistical Mechanics and its Applications* **392 (14)** (2013)., pp. 3095–3105.

[101] MacArthur, R.H., Fluctuations in animal populations and a measure of community stability, *Ecology* **36 (3)** (1955), pp. 533–536.

[102] Malliaros, F.D.; Vazirgiannis, M.; Clustering and Community Detection in Directed Networks: A survey, *Physics reports* **533(4)** (2013.), pp. 95–142.

[103] de–Marcos, L.; Barchino, R.; Martinez, J.-J.; Gutierrez, J.-A., Competency-based Intelligent Curriculum Sequencing using Particle Swarms, *Advanced Learning Technologies* (2008), pp. 295–297.

[104] Maritan, A.; Rinaldo, A.; Rigon, R.; Giacometti, A.; Rodriguez–Iturbe, I., Scaling laws for river networks, *Phys. Rev. E* **53** (1996), pp. 1510–1515.

[105] Marsh, C. J., *Perspectives: Key concepts for understanding curriculum 1* , The Falmer Press, London and Washington, 1997.

[106] Mastropieri, M.A.; Scruggs, T.E., *Encyclopedia of Applied Psychology*, Academic Press, Cambridge, 2004.

[107] Massen, C. P.; Doye, J. P. K., Identifying communities within energy landscapes, *Phys. Rev. E* **71** (2005), DOI: https://doi.org/10.1103/PhysRevE.71.046101.

[108] McAuley, J.; Leskovec, J., Learning to discover social circles in ego networks, *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[109] Meilă, M.; Pentney, W., Clustering by weighted cuts in directed graphs, *SDM '07: Proceedings of the 2007 SIAM International Conference on Data Mining* (2007), pp. 135–144.

[110] Miligram, S., The small world problem, *Psycology Today* **2** (1967), pp. 60–67.

[111] Mollison, D., Spatial contact models for ecological and epidemic spread, *J. R. Stat. Soc. B* **39** (1977), pp. 283–326.

[112] Moreno, Y.; Pastor-Satorras, R.; Vespignani, A., Epidemic outbreaks in complex heterogenous networks, *Eur. Phys. J. B* **26** (2002), pp. 521–529.

[113] Motter, A.E.; Lai, Y.-C., Cascade–based attacks on complex networks, *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* **66(6)** (2002), DOI: 10.1103/PhysRevE.66.065102.

[114] Muff, S.; Rao, F.; Caflisch, A., Local modularity measure for network clusterizations, *Phys. Rev. E* **72** (2005.), DOI : 10.1103/PhysRevE. 72.056107.

[115] Newman, M.E.J., The Structure and Function of Complex Networks , *SIAM REVIEW* **45(2)** (2003), 167–256.

[116] Newman, M.E.J., *Networks: An Introduction*, Oxford University Press, Oxford, 2010.

[117] Newman, M. E. J., Detecting community structure in networks, *Eur. Phys. J. B.*, **38 (2)** (2004), pp. 321330.

[118] Newman, M. E. J., Fast algorithm for detecting community structure in networks, *Phys. Rev. E.*, **69 (6)** (2004), DOI: 10.1103/PhysRevE.69.066133.

[119] Newman, M. E. J.; Barabási, A.–L.; Watts, D.J., *The Structure and Dynamics of Networks*, Princeton University Press, New Jersey, SAD, 2006.

[120] Newman, M. E. J.; Girvan, M., Finding and evaluating community structure in networks, *Phys. Rev. E* **69** (2004), DOI : 10.1103/PhysRevE.69.026113.

[121] Newman, M. E. J., Modularity and community structure in networks, *Proceedings of the National Academy of Sciences* **103 (23)** (2006), pp. 8577–8582.

[122] Jokić, B.; Baranović, B.; Hitrec, S.; Reškovac, T.; Ristić Dedić, Z.; Vuk, B.; Vuk, R., Okvir nacionalnog kurikula (prijedlog), veljača 2016.

[123] Ozkanlar, A.; Clark, A.E., ChemNetworks: a complex network analysus tool for chemical systems, *J. Comput. Chem* **35(6)** (2014), pp. 495–505.

[124] Palla, G.; Derényi, I.; Farkas, I.;Vicsek, T., Uncovering the overlapping community structure of complex networks in nature and society, *Nature* **435** (2005), pp. 814–818 .

[125] Palla, G.; Barabási, A.-L.; Vicsek, T., Quantifying social group evolution, *Nature* **446** (2007), 664–667, DOI:10.1038/nature05670.

[126] Posner, G.J.; Strike, K.A., A Categorization Scheme for Principles of Sequencing Content, *Review of Educational Research* **46 (4)** (1976), pp. 665–690.

[127] Pratt, D., *Curriculum planning: A handbook for professionals*, Harcourt Brace College Publishers, Fort Worth, 1994.

[128] Previšić, V., Pedagogija i metodologija kurikuluma, u *Kurikulum: teorije, metodologija, sadržaji, struktura* Školska knjiga, Zagreb, (2007), pp. 15–37.

[129] Radicchi, F.; Fortunato, S.; Markines, B.; Vespignani, A., Diffusion of scientific credits and the ranking of scientist, *Phys. Rev. E.* **80** (2009), DOI : 10.1103/PhysRevE.80.056103.

[130] Rappoport, D.; Galvin, C.J.; Zubarev, D.Y.; Aspuru-Guzik, A., Complex Chemical Reaction Networks from Heuristic-Aided Quantum Chemistry, *Journal of Chemical Theory and Computation* **10(3)** (2014), pp. 897–907.

[131] Raghavan, U.N.; Albert, R.; Kumara, S., Near linear time algorithm to detect community structures in large-scale networks, *Phys .Rev. E* **76** (2007), DOI : 10.1103/PhysRevE.76.036106.

[132] Rokach, L.; Maimon, O., Clustering methods, *Data mining and knowledge discovery handbook*, Springer, SAD, 2005.,pp. 321-352.

[133] Rouvray, D.H.; King, R.B., *Topology in Chemistry: Discrete Mathematics of Molecules*, Woodhead Publishing, Cambridge, 2002.

[134] Rubinov, M.; Sporns, O., Complex network measures of brain connectivity: Uses and interpretations, *NeuroImage* **52 (3)** (2010), pp. 1059–1069.

[135] Satuluri, V.; Parthasarathy, S., Symmetrizations for clustering directed graphs, *EDBT '11: Proceedings of the 14th International Conference on Extending Database Technology* (2011), pp. 343–354.

[136] Schuetz, P.; Caflisch, A., Efficient modularity optimization by multistep greedy algorithm and vertex refinement, *Phys. Rev. E* **77** (2008), DOI : 10.1103/PhysRevE.77.046112.

[137] Shin, W.-Y.; Singh, B.C.; Cho, J.; Everett, A.M., A New Understanding of Friendships in Space: Complex Networks meet Twitter, *Journal of information science* **41(6)** (2015), pp. 751–764.

[138] Skupnjak, D., Kurikulum i profesionalni razvoj učitelja u Hrvatskoj, *Napredak* **152(2)** (2011), pp. 305–324.

[139] Smith, M. K., Curriculum theory and practice: The Encyclopaedia of Informal Education, (1996, 2000), www.infed.org/biblio/b-curric.htm.

[140] Speidel, L.; Takaguchi, T.; Masada, N., Community detection in directed acyclic graphs, *European Physical Journal B 88* (2015), DOI: 10.1140/epjb/e2015-60226-y.

[141] Statista–The Statistical Portal (www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/)

[142] Steinhaeuser, K.; Chawla, N.V.; Ganguly, A.R., Complex networks as an unified framework for descriptive analysis and predictive modeling in climate science, *Statistical Analy Data Mining* **4** (2010), pp. 497–511.

[143] Stern, M.; Woolf, B.P.; Kurose, J.F., Intelligence on the Web?, *Artificial Intelligence in Education* (1997), pp. 490–497.

[144] Stern, M.; Steinberg, J.; Lee, H.I.; Padhye, J.; Kurose, J., MANIC: Multimedia Asyn- chronous Networked Individualized Courseware, *Proceedings of World Conference on Educational Multimedia and Hypermedia, Calgary, Canada, 1997.* (1997), IOS10-oai:CiteSeerX.psu:10.1.1.123.9236.

[145] Stern, M.K.; Woolf, B.P. , *Intelligent Tutoring Systems: Curriculum Sequencing in a Web - Based Tutor*, Springer Berlin Heidelberg, Berlin, 1998.

[146] Su, S. W., The Various Concepts of Curriculum and the Factors Involved in Curricula-making, *Journal of Language Teaching and Research* **3 (1)** (2012), pp. 153–158.

[147] Suchenek, M.A., Elementary Yet Precise Worst–Case Analysis of Floyd's Heap Construction Program, *Fundamenta Informaticae* **120 (1)** (2012), pp. 75–92.

[148] Suppes, P., Mathematical concept formation in children, *American Psychologists* **21** (1966), pp. 139–150.

[149] Taba, H., *Curriculum development:Theory and practice*, Harcourt, Brace & World, New York, 1962.

[150] Tarjan, R.E., Edge-disjoint spanning trees and depth-first search, *Acta Informatica* **6 (2)** (1976.), pp. 171–185.

[151] Travers, J.; Miligram, S., An experimental study of the small world problem, *Sociometry* **32** (1969), pp. 425–443.

[152] Tyler, R.W., *Basic princeplesof curriculum and instruction*, University od Chicago, Chicago ,1950.

[153] Ugander, J.; Karrer, B.; Backstrom, L.; Marlow, C., The Anatomy ot the Facebook Social Graph, (2011)., arXiv preprint arXiv:1111.4503.

[154] Van den Broeck, W.; Gioannini, C.; Goncalves, B.; Quaggiotto, M.; Colizza, V.; Vespignani, A., The GLEaMviz computational tool, a publicaly available software to explore realistic epidemic spreading scenarios at the global scale, *BMC Infect Dis* (2011), DOI: 10.1186/1471-2334-11-37.

[155] Veljan, D., *Kombinatorna i diskretna matematika*, Algoritam, Zagreb, 2001.

[156] Vican, D.; Bognar, L., Previšić, V., Hrvatski nacionalni kurikulum, *Kurikulum: teorije, metodologija, sadržaji, struktura*,Školska knjiga, Zagreb (2007), pp. 157–204.

[157] Venkatasubramanian, V.; Yuan, Y., Analysis od 1996 Western American Electric Blackouts, rad predstavljen na *Bulk Power System Dyamics and Control - VI. 2004.*, Cortina d'Ampezzo, Italija, pp. 685–721.

[158] Vespignani, A., Modeling dynamical processes in complex socio–technological systems, *Nature Physics* **8** (2012), pp. 32–39.

[159] Vukičević, D.; Caporossi, G., Network descriptors based on betweenness centrality and transmission and their extremal values, *Discrete Applied Mathematics* **161 (16-17)** (2013), pp. 2678-2686.

[160] Vukičević, D.; Zlatić, V., Security of networks with distributed keys, u pripremi.

[161] Wagner, A.; Fell, D.A., The small world inside large metabolic networks, *P. Roy. Soc. B-Biol. Sci.* **268** (2001), pp. 1803–1810

[162] Watts, D. J.; Strogatz, S., Collective dynamics of small–world networks, *Nature* **393** (1998), pp. 440–442.

[163] West,D.B., *Introduction to Graph Theory, 2nd ed.*, Pearson Education Inc, New Jersey, 2001.

[164] Wiener, H., Structural Determination of Paraffin Boiling Points, *J. Amer. Chem. Soc* **1 (69)** (1947), pp. 17–20.

[165] Williams, J. W. J., Algorithm 232 – Heapsort, *Communications of the ACM* **7 (6)** 1964), pp. 347–348.

[166] White, J.G.; Southgate, E.; Thompson, J.N.; Brenner, S., The structure of the nervous system of the nematode, Caenorhabditis Elegans, *Philos. T. R. Soc. Lond.* **314** (1986), pp. 1–340.

[167] Xiaoran, Y.; Jensen, J. E.; Krzakala, F.; Moore, C.; Shalizi, C. R.; Zdeborova, L.; Zhang, P.; Zhu, Y., Model Selection for Degree-corrected Block Models, *J Stat Mech* (2012), DOI: 10.1088/1742-5468/2014/05/P05007.

[168] Xie, J.; Szymanski, B.K.; Liu, X., SLPA: Uncovering Overlapping Communities in Social Networks via A Speaker-listener Interaction Dynamic Process, *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops* (2011), pp. 344-349.

[169] Xing, Y.; Meng, F.; Zhou, Y.; Zhu, M.; Shi, M.; Sun, G., A Node Influence Based Label Propagation Algorithm for Community Detection in Networks, *The Scientific World Journal* **2014** (2014), DOI: 10.1155/2014/627581.

[170] Yang, Z.; Algesheimer, R.; Tessone, C. J., A Comparative Analysis of Community Detection Algorithms on Artificial Networks, *Scientific Reports* **6** (2016), DOI:10.1038/srep30750.

[171] Zachary, W.W., An Information Flow Model for Conflict and Fission in Small Groups, *Journal of Anthropological Research* **33(4)** (1977), pp. 452–473.

[172] Zanetti, S.; Sousa, E.; Oliveira, V.; Almeida, F.; Bernardo, S., Estimating Evapotranspiration Using Artificial Neural Network and Minimum Climatological Data, *J.Irrig.Drain Eng* **133:2(83)** (2007),pp. 83–89.

[173] Zapata, M., Sequencing of Contents and Learning Objects, *RED. Revista de Educacioon a Distancia.* **13** (2005), http://www.um.es/ead/red/13/

[174] Zapata, M., Sequencing of Contents and Learning Objects - Part II, *RED. Revista de Educacion a Distancia.* **14** (2006), http://www.um.es/ead/red/14/

[175] Zapata, M., Sequencing of Contents and Learning Objects - Part III, *RED. Revista de Educacion a Distancia.* **15** (2006), http://www.um.es/ead/red/15/

[176] Zhang, A.; Ren, G.; Lin, Y.; Jia, B.; Cao, H.; Zhang, J.; Zhang, S., Detecting Community Structures in Networks by Label Propagation with Prediction of Percolation Transition, *The Scientific World Journal* **2014** (2014), http://dx.doi.org/10.1155/2014/148686.

[177] Zhou, D.; Schölkopf, B.; Hofmann, T., Semi-supervised learning on directed graphs, *NIPS '05: Advances in Neural Information Processing Systems*, (2005), pp. 16331640.

[178] Ziv, E.; Middendorf, M.; Wiggins, C., Information–theoretic approach to network modularity, *Phys. Rev. E* **71** (2005), DOI: 10.1103/PhysRevE.71.046117.

Dodatak A

Kurikulne mreže

A.1 *Skup racionalnih brojeva*

1. **Prirodni broj**
2. Skup prirodnih brojeva $\mathbb{N} \leftarrow 1$
3. Zbrajanje prirodnih brojeva $\leftarrow 1, 2$
4. Oduzimanje prirodnih brojeva $\leftarrow 1, 2, 3$
5. Množenje prirodnih brojeva $\leftarrow 1, 2, 3$
6. Dijeljenje prirodnih brojeva $\leftarrow 1, 2, 5$
7. Višekratnik $\leftarrow 1, 2, 3, 5$
8. Najmanji zajednički višekratnik $\leftarrow 1, 2, 3, 5, 7$
9. Djelitelj (mjera) $\leftarrow 1, 2, 6, 7$
10. Najveći zajednički djelitelj(mjera) $\leftarrow 1, 2, 6, 7, 8$
11. Nula

12. Negativni broj $\leftarrow 1, 4$
13. Zbrajanje negativnih brojeva $\leftarrow 1, 3, 4, 11, 12,$
14. Oduzimanje negativnih brojeva $\leftarrow 1, 3, 4, 11, 12, 13$
15. Množenje negativnih brojeva $\leftarrow 1, 3, 5, 11, 12, 13$
16. Dijeljenje negativnih brojeva $\leftarrow 1, 6, 11, 12, 13, 15$
17. **Cijeli broj** $\leftarrow 1, 11, 12$
18. Skup cijelih brojeva $\mathbb{Z} \leftarrow 1, 2, 11, 12, 13, 17$
19. Zbrajanje cijelih brojeva $\leftarrow 1, 3, 4, 11, 12, 13, 14, 17, 18$
20. Oduzimanje cijelih brojeva $\leftarrow 1, 4, 11, 12, 14, 17, 18, 19$
21. Množenje cijelih brojeva $\leftarrow 1, 5, 11, 12, 15, 17, 18, 19$
22. Dijeljenje cijelih brojeva $\leftarrow 1, 6, 11, 12, 16, 17, 18, 21$
23. Uspoređivanje cijelih brojeva $\leftarrow 1, 11, 12, 17, 18$
24. **Razlomak** $\leftarrow 1, 2, 6, 11, 17, 18, 22$
25. Brojnik $\leftarrow 1, 11, 17, 24$
26. Nazivnik $\leftarrow 1, 17, 24, 25$
27. Recipročna vrijednost $\leftarrow 22, 24, 25, 26$
28. Skraćivanje razlomaka $\leftarrow 9, 10, 22, 23, 24, 25, 26$
29. Proširivanje razlomaka $\leftarrow 21, 23, 24, 25, 26$
30. Najmanji zajednički nazivnik $\leftarrow 7, 8, 24, 25, 26, 29$

31. Uspoređivanje razlomaka $\leftarrow 24, 25, 26, 28, 29, 30$
32. Zbrajanje razlomaka $\leftarrow 19, 24, 25, 26, 28, 29, 30$
33. Oduzimanje razlomaka $\leftarrow 20, 24, 25, 26, 28, 29, 30, 32$
34. Množenje razlomaka $\leftarrow 21, 24, 25, 26, 28$
35. Dijeljenje razlomaka $\leftarrow 22, 24, 25, 26, 27, 34$
36. Dvojni razlomak $\leftarrow 24, 25, 26, 27, 28$
37. **Decimalni broj** $\leftarrow 1, 6, 17, 22, 24, 25, 26$
38. Zbrajanje decimalnih brojeva $\leftarrow 3, 13, 19, 24, 32, 37$
39. Oduzimanje decimalnih brojeva $\leftarrow 4, 14, 20, 24, 33, 37, 38$
40. Množenje decimalnih brojeva $\leftarrow 5, 15, 21, 24, 34, 37, 38$
41. Dijeljenje decimalnih brojeva $\leftarrow 6, 16, 22, 24, 35, 37, 40$
42. **Racionalni broj** $\leftarrow 1, 11, 12, 17, 24, 37$
43. Zbrajanje racionalnih brojeva $\leftarrow 24, 32, 37, 38, 42$
44. Oduzimanje racionalnih brojeva $\leftarrow 24, 33, 37, 39, 42$
45. Množenje racionalnih brojeva $\leftarrow 24, 34, 37, 40, 42, 43$
46. Dijeljenje racionalnih brojeva $\leftarrow 24, 33, 37, 41, 42, 45$
47. Skup racionalnih brojeva $\mathbb{Q} \leftarrow 2, 18, 24, 32, 33, 34, 35, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46$

A.2 *Elementarne funkcije*

1. **Skup**
2. Element skupa ← 1
3. Prazan skup← 1, 2
4. Neprazan skup ← 1, 2, 3
5. Podskup ← 1, 2, 3
6. Skup realnih brojeva ← 1, 2, 4
7. Kartezijev produkt skupova ← 1, 2, 4, 5, 6
8. Binarna relacija ← 1, 4, 5, 6, 7
9. Totalna binarna relacija ← 1, 2, 3, 5, 6, 7, 8
10. Funkcionalna binarna relacija ← 1, 2, 3, 5, 6, 7, 8
11. **Domena** ← 1, 4, 5, 6
12. Kodomena ← 1, 4, 5, 6
13. Funkcija ← 5, 6, 8, 9, 10, 11, 12,
14. Jednakost funkcija ← 11, 12, 13
15. Slika skupa u odnosu na funkciju ← 1, 4, 11, 12, 13
16. Praslika skupa u odnosu na funkciju ← 1, 4, 11, 12, 13
17. Kompozicija funkcija ← 4, 5, 6, 11, 12, 13
18. Identiteta ← 11, 12, 13, 17

19. Graf funkcije $\leftarrow 4, 5, 7, 11, 12, 13$
20. Injekcija $\leftarrow 11, 12, 13, 19$
21. Surjekcija $\leftarrow 11, 12, 13, 19$
22. Bijekcija $\leftarrow 13, 20, 21$
23. Inverzna funkcija $\leftarrow 13, 17, 18, 19, 20, 21, 22$
24. Parna funkcija $\leftarrow 11, 12, 13, 19$
25. Neparna funkcija$\leftarrow 11, 12, 13, 19$
26. Periodična funkcija $\leftarrow 11, 12, 13, 19$
27. Period $\leftarrow 26$
28. Omeđena funkcija $\leftarrow 11, 12, 13, 15, 19$
29. Rastuća funkcija $\leftarrow 11, 12, 13, 15, 19$
30. Padajuća funkcija $\leftarrow 11, 12, 13, 15, 19$
31. Monotona funkcija $\leftarrow 13, 29, 30$
32. Strogo rastuća funkcija $\leftarrow 13, 15, 19, 29$
33. Strogo padajuća funkcija $\leftarrow 13, 15, 19, 30$
34. Strogo monotona funkcija $\leftarrow 13, 32, 33$
35. Restrikcija funkcije $\leftarrow 4, 5, 11, 12, 13$
36. **Opća potencija** $\leftarrow 5, 6, 11, 12, 13, 19$
37. Polinom $\leftarrow 5, 6, 11, 12, 13, 36$

38. Stupanj polinoma $\leftarrow 13, 36, 37$
39. Koeficijenti $\leftarrow 37$
40. Vodeći koeficijent $\leftarrow 37, 39$
41. Nultočke polinoma $\leftarrow 37, 38$
42. Faktorizacija polinoma $\leftarrow 37, 38, 39, 40, 41$
43. Dijeljenje polinoma $\leftarrow 37, 38, 39, 40, 42$
44. Racionalna funkcija $\leftarrow 4, 5, 6, 11, 12, 13, 19, 37, 42, 43$
45. Prava racionalna funkcija $\leftarrow 37, 38, 42, 43, 44$
46. Neprava racionalna funkcija $\leftarrow 37, 38, 42, 43, 44$
47. Nultočke racionalne funkcije $\leftarrow 37, 38, 41, 44$
48. Rastav funkcije na parcijalne razlomke$\leftarrow 37, 38, 41, 42, 44$
49. Iracionalne funkcije $\leftarrow 36, 44$
50. **Baza** a
51. Baza e
52. Eksponent $\leftarrow 36, 44$
53. Eksponencijalna funkcija $\leftarrow 11, 12, 13, 19, 22, 23, 34, 35, 50, 51, 52$
54. Logaritamska funkcija $\leftarrow 11, 12, 13, 19, 22, 23, 34, 35, 50, 51, 52, 53$
55. Prirodni logaritam $\leftarrow 51, 54$
56. Eksponencijalna jednadžba $\leftarrow 17, 18, 23, 50, 51, 53, 54,$

57. Logaritamska jednadžba ← 17, 18, 23, 50, 51, 53, 54,

58. **Brojevna kružnica**

59. Funkcija sinus ← 11, 12, 13, 19, 21, 25, 26, 28, 58

60. Sinusoida ← 19, 21, 25, 26, 27, 28, 59

61. Funkcija kosinus ← 11, 12, 13, 19, 24, 26, 27, 28, 58

62. Kosinusoida← 19, 21, 24, 26, 27, 28, 61

63. Funkcija tangens ← 11, 12, 13, 19, 21, 25, 26, 59, 61

64. Funkcija kotangens ← 11, 12, 13, 19, 25, 26, 28, 59, 61

65. Trigonometrijske funkcije ← 13, 23, 59, 61, 63, 64

66. Arkus sinus ← 11, 12, 13, 19, 22, 23, 34, 35, 59

67. Arkus kosinus ← 11, 12, 13, 19, 22, 23, 34, 35, 61

68. Arkus tangens ← 11, 12, 13, 19, 22, 23, 34, 35, 63

69. Arkus kotangens ← 11, 12, 13, 19, 22, 23, 34, 35, 64

70. Ciklometrijske (arkus) funkcije ← 13, 23, 66, 67, 68, 69

71. Osnovne elementarne funkcije ← 13, 37, 44, 53, 54, 65, 70

72. Sinus hiperbolni ← 6, 11, 12, 13, 19, 22, 25, 32, 34, 51, 53

73. Kosinus hiperbolni ← 6, 11, 12, 13, 19, 21, 24, 34, 51, 53

74. Tangens hiperbolni ← 6, 11, 12, 13, 19, 22, 24, 33, 34, 51, 53, 72, 73

75. Kotangens hiperbolni ← 6, 11, 12, 13, 19, 22, 24, 33, 34, 51, 53, 72, 73, 74

76. Hiperbolne funkcije $\leftarrow 13, 23, 72, 73, 74, 75$

77. Area sinus hiperbolni $\leftarrow 6, 11, 12, 13, 19, 22, 23, 32, 34, 51, 53, 72$

78. Area kosinus hiperbolni $\leftarrow 6, 11, 12, 13, 19, 22, 23, 32, 34, 51, 53, 73$

79. Area tangens hiperbolni $\leftarrow 6, 11, 12, 13, 19, 22, 23, 32, 34, 51, 53, 74$

80. Area kotangens hiperbolni $\leftarrow 6, 11, 12, 13, 19, 22, 23, 30, 31, 51, 53, 75$

81. Area funkcije $\leftarrow 13, 23, 77, 78, 79, 80$

82. **Elementarne funkcije** $\leftarrow 13, 36, 37, 44, 53, 54, 59, 61, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81$

83. Algebarske funkcije $\leftarrow 13, 17, 36, 44, 82,$

84. Transcedentne funkcije $\leftarrow 13, 17, 53, 54, 65, 70, 76, 81, 82, 83$

A.3 *Integral*

1. **Sud**
2. (Semantička) vrijednost suda ← 1
3. Složeni sud, logičke operacije ← 1, 2
4. Semantičke tablice ← 3
5. Negacija suda ← 3, 4
6. Konjunkcija sudova ← 3, 4
7. (Ekskluzivna) Disjunkcija sudova ← 3, 4
8. Implikacija sudova ← 3, 4
9. Ekvivalencija sudova ← 3, 4, 8
10. (Semantička) jednakost sudova ← 5, 6, 7, 8, 9
11. Svojstva logičkih operacija ← 5, 6, 7, 8, 9, 10
12. Kvantifikatori ← 3
13. Predikat ← 5, 6, 7, 8, 9, 11, 12
14. Tautologija ← 10, 13
15. **Skup (ne definira se)**
16. Element skupa ← 15
17. Prazan skup ← 16
18. Zadanost skupa ← 16

19. Podskup $\leftarrow 16, 8$
20. Partitivni skup $\leftarrow 16, 17, 19$
21. Jednakost skupova $\leftarrow 9, 16, 17, 19$
22. Univerzalni skup $\leftarrow 16, 17, 19$
23. Komplement skupa $\leftarrow 5, 16, 17, 22$
24. Presjek skupova $\leftarrow 6, 16, 17$
25. Disjunktni skupovi $\leftarrow 24$
26. Unija skupova $\leftarrow 7, 16, 17$
27. Particija skupa $\leftarrow 19, 21, 24, 25, 26$
28. Razlika skupova $\leftarrow 16, 21, 23, 24$
29. Simetrična razlika $\leftarrow 21, 25, 26, 28$
30. Svojstva skupovnih operacija $\leftarrow 21, 23, 24, 25, 26, 28, 29$
31. Uređeni par $\leftarrow 15, 16$
32. Jednakost uređenih parova $\leftarrow 31$
33. Kartezijev produkt skupova $\leftarrow 15, 16, 31$
34. Binarna relacija $\leftarrow 19, 20, 33$
35. Svojstva binarne relacije $\leftarrow 6, 8, 12, 34$
36. Relacija ekvivalencije $\leftarrow 35$
37. Uređajna relacija $\leftarrow 35$

38. Uređeni skup ← 37
39. Međa skupa ← 7, 38
40. Minimum, maksimum skupa ← 39
41. **Funkcija** ← 13, 15, 16, 34
42. Domena ← 41
43. Kodomena ← 41
44. Argument funkcije ← 42
45. Vrijednost funkcije ← 43, 44
46. Slika funkcije ← 19, 45
47. Zadanost funkcije ← 41, 42, 43, 45
48. Jednakost funkcija ← 13, 21, 41, 42, 44, 45
49. Restrikcija funkcije ← 13, 19, 41, 42, 44, 45
50. Injekcija ← 13, 41, 42, 44, 45
51. Surjekcija ← 13, 21, 43, 46
52. Bijekcija ← 50, 51
53. Ekvipotentni skupovi ← 52
54. Beskonačan skup ← 19, 52
55. Konačan skup ← 54
56. Kompozicija funkcija ← 13, 19, 41, 42, 43, 44, 45, 46

57. Identiteta $\leftarrow 13, 21, 41, 42, 43, 44, 45, 46$
58. Inverzna funkcija $\leftarrow 52, 56, 57$
59. **Skup prirodnih brojeva** $\leftarrow 15, 16, 19, 21, 50, 51$
60. Operacije na skupu $\leftarrow 33, 59$
61. Uređaj na skupu $\leftarrow 37, 38, 59, 60$
62. Interval na skupu $\leftarrow 59, 61$
63. Kardinalni broj skupa $\leftarrow 53, 62$
64. Prebrojiv skup $\leftarrow 53, 59$
65. Diskretan skup $\leftarrow 17, 21, 61$
66. Skup cijelih brojeva $\leftarrow 19, 26, 59, 60$
67. Operacije na skupu $\leftarrow 60, 66$
68. Uređaj na skupu $\leftarrow 61, 66, 67$
69. Skup racionalnih brojeva $\leftarrow 19, 33, 34, 36, 59, 66, 67$
70. Operacije na skupu $\leftarrow 67, 69$
71. Uređaj na skupu $\leftarrow 68, 69, 70$
72. Gust skup $\leftarrow 54, 71$
73. Točka (ne definira se)
74. Skup točaka $\leftarrow 15, 16, 73$
75. Pravac (ne definira se) $\leftarrow 74$

76. Dužina $\leftarrow 19, 75$
77. Ishodište $\leftarrow 41, 42, 44, 45, 66, 75$
78. Jedinična točka $\leftarrow 41, 42, 44, 45, 66, 75$
79. Jedinična dužina $\leftarrow 76, 77, 78$
80. Brojevni pravac (koordinatni sustav na pravcu) $\leftarrow 59, 66, 69, 75, 79$
81. Skup iracionalnih brojeva $\leftarrow 23, 69, 80$
82. Skup realnih brojeva $\leftarrow 19, 25, 26, 69, 81$
83. Operacije na skupu $\leftarrow 70, 82$
84. Uređaj na skupu $\leftarrow 71, 80, 82$
85. Intervali na skupu $\leftarrow 19, 80, 84$
86. Svojstva operacija u skupu $\leftarrow 83$
87. **Realna funkcija realnog argumenta** $\leftarrow 19, 41, 42, 43, 44, 45, 46, 82$
88. Prirodno područje definicije funkcije $\leftarrow 87$
89. Konstantna funkcija $\leftarrow 13, 87$
90. Rastuća funkcija $\leftarrow 13, 84, 87$
91. Padajuća funkcija $\leftarrow 13, 84, 87$
92. Monotona funkcija $\leftarrow 89, 90, 91$
93. Pozitivna funkcija $\leftarrow 13, 84, 85, 87$
94. Negativna funkcija $\leftarrow 13, 84, 85, 87$

95. Omeđena funkcija $\leftarrow 13, 84, 85, 87$
96. Razlomljena funkcija $\leftarrow 27, 87$
97. Apsolutna vrijednost broja $\mid x \mid \leftarrow 96$
98. Funkcija predznaka $\leftarrow 96$
99. Jednakost $\leftarrow 34, 36, 82$
100. Jednadžba $\leftarrow 87, 89$
101. Rješenje jednadžbe $\leftarrow 99, 100$
102. Skup rješenja jednadžbe $\leftarrow 15, 16, 101$
103. Nejednadžba $\leftarrow 84, 87, 89$
104. Rješenje nejednadžbe $\leftarrow 85, 103$
105. **Ravnina** $\leftarrow 74$
106. Koordinatni sustav u ravnini $\leftarrow 24, 31, 80, 105$
107. Apscisna os $\leftarrow 80, 106$
108. Ordinatna os $\leftarrow 80, 106$
109. Kvadratni $\leftarrow 19, 105, 106$
110. Apscisa točke $\leftarrow 106, 107$
111. Ordinata točke $\leftarrow 106, 108$
112. Duljina $\leftarrow 19, 41, 42, 43, 44, 45, 76, 82, 84, 105$
113. Udaljenost $\leftarrow 33, 41, 42, 43, 44, 45, 82, 84, 97, 105$

114. Geometrijski lik ← 19, 105

115. Opseg lika← 112, 114

116. Ploština lika ← 27, 41, 42, 43, 44, 45, 83, 84, 99, 114

117. Nultočka funkcije ← 24, 87, 99, 107, 117

118. Graf funkcije ← 87, 106

119. Pomak grafa po osi x ← 56, 107

120. Pomak grafa po osi y ← 56, 108

121. Funkcijska jednadžba ← 56, 100

122. Parna funkcija ← 87, 99, 117

123. Neparna funkcija ← 87, 99, 117

124. Periodička funkcija ← 87, 99, 117

125. Linearna funkcija ← 87

126. Graf linearne funkcije ← 75, 117, 125

127. Jednadžba pravca ← 125

128. Koeficijent smjera pravca ← 127

129. Odsječak pravca na y-osi ← 108, 127

130. Paralelni pravci ← 25, 36, 75, 99, 105, 128

131. Okomiti pravci ← 24, 34, 75, 99, 105, 128

132. **Niz brojeva** ← 19, 87, 96

133. Članovi niza ← 132

134. Opći član niza ← 132, 133

135. Zadanost niza ← 47, 134

136. Grafički prikaz niza ← 117, 132

137. Konačan niz ← 55, 133

138. Beskonačan niz ← 54, 133

139. N-ta parcijalna suma niza ← 19, 55, 62, 83, 133

140. Aritmetički niz ← 83, 133

141. Geometrijski niz ← 83, 133

142. Konstantni niz ← 89, 132

143. Rastući niz ← 90, 132

144. Padajući niz ← 91, 132

145. Monotoni niz ← 92, 132, 142, 143, 144

146. Omeđeni niz ← 95, 132

147. Limes (granična vrijednost) niza ← 13, 54, 55, 84, 85, 113, 133, 136

148. Konvergentan niz ← 147

149. Divergentan niz ← 148

150. Gomilište niza ← 147, 148, 149

151. Podniz ← 19, 56, 90, 132, 133, 136

152. Red realnih brojeva ← 83, 132, 133

153. N-ti član reda ← 152

154. K-ta parcijalna suma reda ← 139, 152

155. Niz parcijalnih suma reda ← 154

156. Konvergentan red ← 148, 155

157. Suma reda ← 147, 156

158. Divergentan red ← 156

159. Harmonijski red ← 152

160. Red s pozitivnim članovima ← 152

161. Minoranta reda ← 84, 160

162. Majoranta reda ← 84, 160

163. Kriterij konvergencije ← 156, 161, 162

164. Apsolutno konvergentan red ← 97, 156

165. Alternirani red ← 98, 152

166. Alternirani harmonijski red ← 159, 165

167. Niz funkcija ← 132

168. N-ti član niza funkcija ← 133, 167

169. Limes niza funkcija ← 147, 168

170. Konvergentan niz funkcija ← 148, 169

171. Jednoliko konvergentan niz funkcija ← 170

172. Red funkcija ← 167, 168

173. N-ti član reda funkcija ← 172

174. K-ta parcijalna suma reda funkcija ← 154, 172, 173

175. Niz parcijalnih suma reda funkcija ← 155, 174

176. Konvergentan red funkcija ← 156, 175

177. Apsolutno konvergentan red funkcija ← 164, 172

178. Jednoliko konvergentan red funkcija ← 171, 175

179. Red potencija ← 172

180. **Limes (granična vrijednost) funkcije** ← 13, 28, 84, 85, 87, 113, 117

181. Konvergentna funkcija ← 180

182. Divergentna funkcija ← 181

183. Uklještene funkcije ← 26, 84, 85, 180

184. Vrste limesa ← 54, 55, 180

185. Limes funkcije s lijeva ← 24, 180, 184

186. Limes funkcije s desna ← 24, 180, 184

187. Neprekidnost funkcije u točki ← 180, 181

188. Neprekidnost funkcije na intervalu ←, 19, 187

189. Prirast argumenta funkcije ← 87, 113, 117

190. Prirast vrijednosti funkcije ← 87, 113, 117

191. Uklonjivi prekid ← 99, 185, 186, 187

192. Prekid prve vrste ← 184, 185, 186, 187

193. Prekid druge vrste ← 184, 185, 186, 187

194. Asimptota funkcije ← 75, 87, 106, 110, 113, 148

195. Vertikalna asimptota ← 85, 127, 184, 185, 186, 194

196. Horizontalna asimptota ← 127, 184, 194

197. Kosa asimptota ← 127, 184, 194

198. **Derivacija funkcije u točki** ← 87, 180, 187, 189, 190

199. Derivacija funkcije ← 198

200. Sekanta krivulje ← 24, 75, 87, 117, 127, 128

201. Tangenta krivulje ← 198, 200

202. Normala krivulje ← 131, 201

203. Derivacija s lijeva ← 185, 198

204. Derivacija s desna ← 186, 198

205. Pravila deriviranja ← 56, 57, 58, 83, 89, 198

206. Diferencijal funkcije ← 110, 111, 189, 190, 201

207. Druga derivacija funkcije ← 198

208. N-ta derivacija funkcije ← 207

209. Neodređeni oblici ← 180, 184, 185, 186

210. L'Hospitalovo pravilo ← 85, 87, 187, 198, 209

211. Monotonost funkcije ← 85, 90, 91, 199

212. Kritine točke funkcije ← 187, 198

213. Lokalni ekstremi funkcije ← 26, 84, 85, 188, 207, 212

214. Globalni ekstremi ← 87, 95

215. Zakrivljenost funkcije ← 84, 85, 207

216. Točka pregiba ← 215

217. Tok funkcije ← 87, 88, 117, 118, 122, 123, 124, 195, 196, 197, 210, 211, 213, 215, 216

218. Primitivna funkcija ← 48, 87, 199, 206

219. **Neodređeni integral** ← 15, 16, 120, 218

220. Podintegralna funkcija ← 219

221. Određeni integral ← 85, 87, 219

222. Grafička interpretacija određenog integrala u ravnini ← 85, 106, 107, 108, 110, 111, 116, 117, 220

223. Nepravi integral ← 180, 187, 220, 221, 222

A.4 *Fizika za osnovnu školu*

1. **Duljina**
2. Ploština ← 1
3. Obujam ← 1
4. Masa
5. Gustoća ← 3, 4
6. **Sila** ← 4
7. Koeficijent elastičnosti ← 1, 6
8. Gravitacijsko ubrzanje
9. Težina ← 4, 8
10. Koeficijent trenja
11. Sila trenja ← 9, 10
12. Tlak ← 2, 6
13. Hidrostatski tlak ← 1, 5, 8
14. Sila uzgona ← 3, 5, 8
15. **Rad** ← 1, 6
16. Gravitacijska potencijalna energija ← 1, 4, 8
17. Vrijeme
18. Snaga ← 15, 17

19. Temperatura
20. Specifični toplinski kapacitet
21. Toplina ← 19, 20
22. Naboj
23. **Električna struja** ← 17, 18, 21, 22
24. Napon ← 15, 18, 22
25. Električni otpor ← 23,24
26. **Brzina** ← 1, 17
27. Akceleracija ← 4, 6, 17, 26
28. Frekvencija ← 4, 6
29. Brzina vala ← 17, 28
30. **Žarišna daljina** ← 1
31. Jakost leće ← 30

A.5 *Model primarne proizvodnje*

1. **Fotosinteza**
2. Svjetlost ← 1
3. Funkcija svjetlosti ← 2
4. Parametar ← 1, 3
5. Parametrizacija ← 3, 4
6. Iradijanca ← 2, 3
7. Biomasa ← 1, 2, 4, 5, 6
8. Funkcija svjetlosnog zasićenja ← 1, 2, 3, 4, 5, 7
9. Profil biomase ← 7
10. Trenutna proizvodnja ← 1, 2, 6, 7, 8
11. **Sunčevo zračenje** ← 1, 2, 3, 6
12. Raspršenje ← 2, 3, 11
13. Apsorpcija ← 2, 3, 11
14. Koeficijent atenuacije ← 12, 13
15. Profil iradijance ← 6, 11, 14
16. Proizvodnja pri dubini ← 1, 2, 10, 11, 14, 15
17. Dnevni hod sunčevog zračenja ← 1, 2, 3, 6, 11
18. Dnevni hod iradijance ← 2, 6, 15

19. **Integral** $\leftarrow 9, 16, 17, 18$
20. Dnevna proizvodnja pri dubini $\leftarrow 1, 2, 6, 7, 11, 19$
21. Analitičko rješenje integrala$\leftarrow 19$
22. Numeričko rješenje integrala $\leftarrow 19$
23. Dnevna proizvodnja vodenog stupca $\leftarrow 1, 2, 9, 18, 19, 20$
24. Miješanje $\leftarrow 7$
25. Uniformni profil biomase $\leftarrow 1, 9, 24$
26. Analitičko rješenje za proizvodnju vodenog stupca $\leftarrow 21, 23, 25$
27. Numeričko rješenje za proizvodnju vodenog stupca $\leftarrow 9, 22, 23, 26$
28. Primarna proizvodnja $\leftarrow 1, 2, 6, 7, 9, 14, 17, 18, 23$

A.6 *Obrada podataka*

1. **Podatak**
2. Unos s tipkovnice ← 1
3. Drugi oblici unosa ← 1
4. Ispis na zaslonu ← 1, 2, 3
5. **Broj** ← 1, 2, 3, 4
6. Cijeli broj ← 1, 2, 3, 4, 5
7. Decimalni broj ← 1, 2, 3, 4, 5
8. Zbrajanje ← 1, 4, 5, 6, 7
9. Oduzimanje ← 1, 4, 5, 6, 7, 8
10. Množenje ← 1, 4, 5, 6, 7, 8
11. Dijeljenje ← 1, 4, 5, 6, 7, 8
12. Cjelobrojno dijeljenje ← 1, 4, 5, 6, 7, 11
13. Ostatak pri dijeljenju ← 1, 5, 6, 7, 11, 12
14. Potenciranje ← 1, 5, 6, 7, 10
15. Zaokruživanje ← 1, 5, 7
16. **String** ← 1, 2, 3, 4
17. Komadanje ← 16
18. Podstring ← 16, 17

19. Konkatenacija ← 16, 17, 18
20. Invertiranje ← 16, 18
21. Kapitalizacija ← 16
22. Mjerenje ← 16, 17, 18, 19
23. Uvišestručenje ← 16, 18, 19
24. Pretraživanje ← 16
25. Postojanje podstringa ← 16, 17, 18
26. Pozicija podstringa ← 16, 17, 18, 24, 25
27. Broj ponavljanja podstringa ← 16, 17, 18, 25
28. **Niz** ← 1, 2, 3, 4, 5, 6, 7, 16
29. Dodavanje elementa u niz ← 5, 6, 7, 16, 28
30. Brisanje elementa iz niza ← 5, 6, 7, 16, 28, 29
31. Zamjena elemenata u nizu ← 5, 6, 7, 28
32. Dohvaćanje elementa iz niza ← 5, 6, 7, 28
33. Varijabla ← 1, 2, 3, 4, 5, 16, 28
34. **Datoteka** ← 3, 33
35. Otvaranje datoteke ← 34
36. Zatvaranje datoteke ← 34, 35
37. Upis u datoteku ← 34, 35, 36

38. Čitanje iz datoteke ← 34, 35, 36, 37
39. **Petlja**
40. Siguran ulazak u petlju ← 39
41. Potencijalni ulazak u petlju ← 39, 40
42. Poznati broj ponavljanja ← 39
43. Nepoznati broj ponavljanja ← 39
44. Grananje
45. Višestruko grananje ← 44
46. Prisilan izlazak iz tekućeg kruga ← 39, 43
47. Prisilan izlazak iz petlje ← 39, 43, 44
48. Ulazni parametar ← 1, 2, 3, 34
49. Izlazni parametar ← 1, 2, 3, 4, 34, 48
50. Adresa parametra ← 1, 2, 3, 34
51. Vrijednost parametra ← 1, 2, 3, 34
52. Rekurzija ← 39, 40, 41, 42, 44, 48, 49
53. Funkcija ← 39, 40, 41, 42, 44, 48, 49, 52
54. Obrada podataka ← 1, 2, 3, 4, 5, 6, 7, 16, 18, 28, 33, 34

Dodatak B

Algoritmi

B.1 Algoritam za optimalnu ekspoziciju

Listing B.1.1: Algoritam.h

```
typedef struct Vector{
short start;
        short end;
}Vector;

typedef struct ArrVector {
        Vector* v;
        short size;
        unsigned int S;
}ArrVector;

typedef struct Point {
        short pNum[256];
        short pDiff[256];
        short usedSize;
        unsigned int Smin;
```

```
}Point;

Point Points;
int callNumber;
short changed;

ArrVector* ReadFile(char* filename);

ArrVector* DuplicateArr(ArrVector* arrVector);

void freeArrVector(ArrVector* arrVector);

void WriteArrVector(char* filename, ArrVector* arrVector);

int Calculate_S(ArrVector* arrVector);

int isGoal(ArrVector* arrVector);

int tryChange(ArrVector* arrVector);

int Search(ArrVector *problem);
```

Listing B.1.2: Algoritam.c

```
#include "Algoritam.h"
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

ArrVector * ReadFile(char * filename)
{
        FILE* fp;
        ArrVector* rez = (ArrVector*)malloc(sizeof(ArrVector));
        fp = fopen(filename, "rt");

        fscanf(fp, "%hi␣%hi", &rez->size, &Points.usedSize);
        rez->v = (Vector*)malloc(sizeof(Vector) * rez->size);

        for (int i= 0 ;!feof(fp); i++)
        {
                fscanf(fp, "␣%hi␣%hi␣", &rez->v[i].start, &rez->v[i].end);
        }
        fclose(fp);

        Points.Smin = SHRT_MAX;

        for (int i = 0; i < Points.usedSize; i++)
        { Points.pNum[i] = i + 1;}
        return rez;
}

ArrVector* DuplicateArr(ArrVector * arr)
{
        ArrVector* rez = (ArrVector*)malloc(sizeof(ArrVector));
        rez->v = (Vector*)malloc(sizeof(Vector) * arr->size);
```

```
        rez->size = arr->size;
        rez->S = arr->S;

        for (int i = 0; i < arr->size; i++)
        {
                rez->v[i].start = arr->v[i].start;
                rez->v[i].end = arr->v[i].end;
        }

        return rez;
}

void freeArrVector(ArrVector * arr)
{
        free(arr->v);
        free(arr);
}

void WriteArrVector(char * filename, ArrVector* arr)
{
        FILE* fp;
        fp = fopen(filename, "w");
        fprintf(fp,"%hi %hi\n", arr->size, Points.usedSize);

        for (int i = 0; i < arr->size; i++)
        {
                fprintf(fp,"%hi %hi\n", arr->v[i].start, arr->v[i].end);
        }
        fclose(fp);
}

int Calculate_S(ArrVector * arr)
```

```
{
        int S = 0;

        for (int i = 0; i < arr->size; i++)
        {
                S += (arr->v[i].end - arr->v[i].start);
        }
        return S;
}

int isGoal(ArrVector * arr)
{
        if (arr->S < Points.Smin)
        {
                Points.Smin = arr->S;
                return 1;
        }else
        return 0;
}

int tryChange(ArrVector* arr)
{
        for (int i = 0; i < arr->size; i++)
        {
                arr->v[i].start = Points.pDiff[arr->v[i].start - 1];
                arr->v[i].end = Points.pDiff[arr->v[i].end - 1];

                if (arr->v[i].start >= arr->v[i].end)
                        return 0;
        }
        arr->S = Calculate_S(arr);
        return 1;
```

```
}

int  Search(ArrVector* problem)
{ for (int as = 0; as < Points.usedSize - 1; as++)
        { for (int bs = as + 1; bs < Points.usedSize; bs++)
                { for (int be = bs + 1; be <= Points.usedSize; be++)
                        { ArrVector* solution;
                          solution = DuplicateArr(problem);
                          int i = 0;
                          while (i < Points.usedSize)
                              { if(i < as)
                                { Points.pDiff[i++] = Points.pNum[i];}
                               else if (i >= as && i < be)
                                {int j = bs;
                                 while ( j < be)
                                 { Points.pDiff[i++] = Points.pNum[j++]; }
                                 int k = as;
                                 while (k < bs)
                                 { Points.pDiff[i++] = Points.pNum[k++];}
                                }
                              else
                                 { Points.pDiff[i++] = Points.pNum[i]; }
                            }
                        if (tryChange(solution))
                          { if (isGoal(solution))
                                {WriteArrVector("Zadaci/Test.txt", solution);
                                    printf("S: %d\n ", solution->S);
                                    changed++;
                                }
                            freeArrVector(solution);
                           }
                        else
                            freeArrVector(solution);
```

```
                        }
                }
        return 0;
}
```

Listing B.1.3: Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "Algoritam.h"

void main()
{
        changed = 1;
        while (changed)
        {
                changed = 0;
                ArrVector* test = ReadFile("Zadaci/Test.txt");
                test->S = Calculate_S(test);
                printf("Current S min: %d\n\n", test->S);
                Points.Smin = test->S;
                Search(test);
                printf("\n---END OF STAGE---\n");
                freeArrVector(test);
        }
        printf("\n-----END OF SEARCH-----\n");
}
```

B.2 OLPAm+

Listing B.2.1: Point.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    class Point
    {
        public int id { get; set; }
        public int label { get; set; }
        public int inDeg { get; set; }
        public int outDeg { get; set; }
        public List<int> inN = new List<int>();
        public List<int> outN = new List<int>();
        public int inS { get; set; }
        public int outS { get; set; }
        public List<int> usedLabels = new List<int>();
    }
}
```

Listing B.2.2: Dq.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace GLPAm_
{
    class Dq
    {
        public decimal value { get; set; }
        public int label { get; set; }
    }
}
```

Listing B.2.3: Community.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    class Community
    {
        public int label { get; set; }
        public List<int> inC = new List<int>();
        public List<int> outC = new List<int>();
        public List<Point> points = new List<Point>();
    }
}
```

Listing B.2.4: DeltaQ.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    class DeltaQ
    {
        public decimal value { get; set; }
        public Community c1 { get; set; }
        public Community c2 { get; set; }
    }
}
```

Listing B.2.5: Final.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    class Final
    {
        public List<Point> points = new List<Point>();
        public decimal modularity { get; set; }
    }
}
```

Listing B.2.6: MyHeapMember.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    public class MyHeapMember<TData, TValue> where TValue :
        IComparable<TValue>, new() where TData : new()
    {
        public TData Data;

        public TValue Value;
    }
}
```

Listing B.2.7: MyHeap.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    public class MyHeap<TData, TValue> where TValue : IComparable<
        TValue>, new() where TData : new()
    {
        private MyHeapMember<TData, TValue>[] a;
```

```
        private int heapSize;

        private int heapLimit;

        private bool isEmpty;

        private int left(int index) { return 2 * index; }

        private int right(int index) { return 2 * index + 1; }

        private int parent(int index) { return (int)index / 2; }

        private void downSift(int index)
        {
            bool stopFlag = false;
            int l = left(index);
            int r = right(index);
            int swap;
            MyHeapMember<TData, TValue> temp;
            while (!stopFlag)
            {
                if ((l < heapLimit) && (r < heapLimit))
                {
                    if (a[l].Value.CompareTo(a[r].Value) < 0)
                        swap = r;
                    else
                        swap = l;
                }
                else if (l < heapLimit)
                    swap = l;
                else
                    break;
```

```
            if (a[index].Value.CompareTo(a[swap].Value) < 0)
            {
                temp = a[index];
                a[index] = a[swap];
                a[swap] = temp;

                index = swap;
                l = left(index);
                r = right(index);
            }
            else
                stopFlag = true;
        }
    }

    private void upSift(int index)
    {
        bool stopFlag = false;
        int p = parent(index);
        MyHeapMember<TData, TValue> temp;
        while (!stopFlag)
        {
            if ((p > 0) && (a[p].Value.CompareTo(a[index].Value)
                < 0))
            {
                temp = a[index];
                a[index] = a[p];
                a[p] = temp;

                index = p;
                p = parent(index);
            }
            else
```

```
                stopFlag = true;
            }
        }

         public MyHeap(int size)
        {
            heapSize = size;
            heapLimit = 1;
            a = new MyHeapMember<TData, TValue>[size + 1];
            isEmpty = true;
        }

        public bool HeapIsEmpty() { return isEmpty; }

        public void InsertItem(TData data, TValue value)
        {
            int index;
            index = heapLimit;
            a[index] = new MyHeapMember<TData, TValue>();
            a[index].Data = data;
            a[index].Value = value;
            heapLimit++;
            upSift(index);
            isEmpty = false;
        }

        public void InsertItem(MyHeapMember<TData, TValue> heapMember
            )
        {
            int index;
            index = heapLimit;
            a[index] = heapMember;
            heapLimit++;
```

```
            upSift(index);
            isEmpty = false;
        }

        public MyHeapMember<TData, TValue> ReturnMax() { return a[1];
            }

        public void RemoveMax()
        {
            int small = heapLimit - 1;
            int index = 1;
            if (heapLimit == 2)
            {
                isEmpty = true;
                heapLimit--;
            }
            else
            {
                a[index] = a[small];
                heapLimit--;
                downSift(index);
            }
            a[heapLimit] = null;
        }
    }
}
```

Listing B.2.8: Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GLPAm_
{
    class Program
    {
        static void Main(string[] args)
        {
           Final final = new Final();

            List<int> column1 = new List<int>();
            List<int> column2 = new List<int>();
            int m = 0;

         ReadFile(ref column1, ref column2, ref m);

          for (int i = 1; i <= 100; i++)
            {
                decimal modularity = 0;

                List<Point> points = new List<Point>();
                int n = 0;

                ReadPoints(ref points, column1, column2, ref n);

                int[,] AdMat = new int[n, n];
                decimal[,] ModMat = new decimal[n, n];
```

```
            CalculateAdMat(ref AdMat, column1, column2, n);
            CalculateDegs(AdMat, n, ref points);
            CalculateModMat(ref ModMat, AdMat, points,n,m);

            GLPAm(ref points, ModMat, n, m, ref modularity);

            if (modularity > final.modularity || i == 1)
             {
                      final.points.Clear();
                      final.points.AddRange(points);
                      final.modularity = modularity;
                     }
        }
         bool ok = Check(final.points, column1, column2);
        }

         static void ReadFile(ref List<int> column1, ref List<int>
            column2, ref int m)
        {
            string directory = AppDomain.CurrentDomain.BaseDirectory
               + "\\Network.txt";
            string[] lines = System.IO.File.ReadAllLines(directory);

       int num_lines = 0;
            foreach (string line in lines)
            {
                string[] numbers = line.Split(' ');
                column1.Add(Int32.Parse(numbers[0]));
                column2.Add(Int32.Parse(numbers[1]));
                num_lines++;
            }
         m = num_lines;
```

```
        }

        static void ReadPoints(ref List<Point> points, List<int>
            column1, List<int> column2, ref int n)
        {
            points.Clear();

            for (int i = 0; i < column1.Count; i++)
            {
                bool contains = false;
                foreach (Point point in points)
                {
                    if (point.id == column1[i])
                    {
                        contains = true;
                    }
                }
                if (!contains)
                {
                    Point p = new Point();
                    p.id = column1[i];
                    p.label = column1[i];
                    p.usedLabels.Add(p.label);
                    points.Add(p);
                }
            }

            for (int i = 0; i < column2.Count; i++)
            {
                bool contains = false;
                foreach (Point point in points)
                {
                    if (point.id == column2[i])
```

```
            {
                contains = true;
            }
        }
        if (!contains)
        {
            Point p = new Point();
            p.id = column2[i];
            p.label = column2[i];
            points.Add(p);
        }
    }

    for (int i = 0; i < column1.Count; i++)
    {
        int outN = column2[i];
        foreach (Point point in points)
        {
            if (point.id == column1[i] && !point.outN.
                Contains(outN))
            {
                point.outN.Add(outN);
            }
        }
    }

    for (int i = 0; i < column2.Count; i++)
    {
        int inN = column1[i];
        foreach (Point point in points)
        {
            if (point.id == column2[i] && !point.inN.Contains
                (inN))
```

```
                {
                    point.inN.Add(inN);
                }
            }
        }

        SortPoints(points);

        n = points.Count;
    }

    static void CalculateAdMat(ref int[,] AdMat, List<int>
        column1, List<int> column2, int n)
    {
        for (int k = 0; k < column1.Count; k++)
        {
            int i = column1[k] - 1;
            int j = column2[k] - 1;
            AdMat[i, j] = 1;
        }
    }

    static void CalculateDegs(int[,] AdMat, int n, ref List<Point
        > points)
    {
        foreach (Point p in points)
        {
            int inDeg = 0;
            int outDeg = 0;
            for (int i = 0; i < n; i++)
            {
                if (AdMat[i, p.id - 1] == 1)
                {
```

```
                    inDeg++;
                }
            }
            p.inDeg = inDeg;
            p.inS = inDeg;
            for (int i = 0; i < n; i++)
            {
                if (AdMat[p.id - 1, i] == 1)
                {
                    outDeg++;
                }
            }
            p.outDeg = outDeg;
            p.outS = outDeg;
        }
    }

    static void CalculateModMat(ref decimal[,] ModMat, int[,]
        AdMat, List<Point> points, int n, int m)
    {
        decimal m_decimal = m;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                ModMat[i, j] = AdMat[i, j] - ((points[j].inDeg *
                    points[i].outDeg) / m_decimal);
            }
        }
    }

    static void CalculateModularity(decimal[,] ModMat, List<Point
        > points, int n, int m, ref decimal modularity)
```

```
{
    modularity = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (points[i].label == points[j].label)
            {
                modularity = modularity + ModMat[points[i].id
                    - 1, points[j].id - 1];
            }
        }
    }
    decimal m_decimal = m;
    modularity = modularity / m_decimal;
}

static bool IteratePoints(ref List<Point> points, int m,
    decimal[,] ModMat, int n, bool biggerFirst, bool limited,
    bool step, ref decimal modularity)
{
    bool b = true;

    int iterations = 0;
    int changes = 0;
    do
    {
        iterations++;
        changes = 0;

        RandomizePoints(ref points);

        foreach (Point p in points)
```

```
        {
            List<int> differentLabels = new List<int>();

            List<int> labels = new List<int>();
            List<int> labelsIn = new List<int>();
            List<int> labelsOut = new List<int>();
            for (int i = 0; i < p.inN.Count; i++)
            {
                foreach (Point poi in points)
                {
                    if (poi.id == p.inN[i])
                    {
                        labels.Add(poi.label);
                        labelsIn.Add(poi.label);
                    }
                }
            }
            for (int i = 0; i < p.outN.Count; i++)
            {
                foreach (Point poi in points)
                {
                    if (poi.id == p.outN[i])
                    {
                        labels.Add(poi.label);
                        labelsOut.Add(poi.label);
                    }
                }
            }
            if(labelsIn.Count > 0 && labelsIn.Max() != p.
                label)
            {
                differentLabels.Add(labelsIn.Max());
            }
```

```
if (labelsOut.Count > 0 && labelsOut.Min() != p.
    label)
{
    differentLabels.Add(labelsOut.Min());
}

List<Dq> Dqs = new List<Dq>();
decimal m_decimal = m;
foreach (int dl in differentLabels)
{
    int dj = 0;
    foreach (int l in labels)
    {
        if (l == dl)
        {
            dj++;
        }
    }
    int inS = 0;
    int outS = 0;
    foreach (Point pp in points)
    {
        if (pp.label == dl)
        {
            inS = pp.inS;
            outS = pp.outS;
            break;
        }
    }
    decimal dj_dec = dj;
    decimal outDeg_dec = p.outDeg;
    decimal inDeg_dec = p.inDeg;
    decimal inS_dec = inS;
```

```
                    decimal outS_dec = outS;
                    decimal _Dq = dj_dec / m_decimal - (((
                        outDeg_dec * inS_dec) + (inDeg_dec *
                        outS_dec)) / (m_decimal * m_decimal));

                    if (_Dq > 0)
                    {
                        Dq Dq = new Dq();
                        Dq.value = _Dq;
                        Dq.label = dl;
                        Dqs.Add(Dq);
                    }
                }

                Dq maxDq = GetMaxDq(Dqs);
                if(maxDq == null)
                {
                    break;
                }

                if (p.usedLabels.Contains(maxDq.label))
                {
                    break;
                }
    changes++;


                int previousLabel = p.label;
                int newLabel = maxDq.label;
            p.label = 0;

                int inSprevious = 0;
                int outSprevious = 0;
```

```
int inSnew = 0;
int outSnew = 0;

foreach (Point pp in points)
{
    if (pp.label == previousLabel)
    {
        inSprevious = pp.inS;
        outSprevious = pp.outS;
        break;
    }
}

foreach (Point pp in points)
{
    if (pp.label == newLabel)
    {
        inSnew = pp.inS;
        outSnew = pp.outS;
        break;
    }
}

p.label = newLabel;
p.usedLabels.Add(newLabel);

inSprevious = inSprevious - p.inDeg;
outSprevious = outSprevious - p.outDeg;
inSnew = inSnew + p.inDeg;
outSnew = outSnew + p.outDeg;

foreach (Point pp in points)
{
```

```
                    if (pp.label == newLabel)
                    {
                        pp.inS = inSnew;
                        pp.outS = outSnew;
                    }
                    if (pp.label == previousLabel)
                    {
                        pp.inS = inSprevious;
                        pp.outS = outSprevious;
                    }
                }
            }

            if (step)
            {
                PrintPoints(points);
            }
            CalculateModularity(ModMat, points, n, m, ref
                modularity);

            if (limited)
            {
                b = changes != 0 && iterations != n;
            }
            else
            {
                b = changes != 0;
            }
        } while (b);
        if (changes != 0)
        {
            return false;
        }
```

```
        else
        {
            return true;
        }
    }

    static void CreateCommunities(List<Point> points, ref List<
        Community> communities)
    {
        communities.Clear();

        for (int i = 0; i < points.Count - 1; i++)
        {
            for (int j = i + 1; j < points.Count; j++)
            {
                if (points[i].label > points[j].label)
                {
                    Point temp = points[i];
                    points[i] = points[j];
                    points[j] = temp;
                }
            }
        }

        int label = points[0].label;
        Community c1_ = new Community();
        c1_.label = points[0].label;
        communities.Add(c1_);

        foreach(Point p in points)
        {
            if(p.label != label)
            {
```

```
            Community c_ = new Community();
            c_.label = p.label;
            communities.Add(c_);
            label = p.label;
        }
    }

    foreach(Community c in communities)
    {
        foreach(Point p in points)
        {
            if(p.label == c.label)
            {
                c.points.Add(p);
            }
        }
    }

    foreach(Community c1 in communities)
    {
        foreach(Community c2 in communities)
        {
            foreach(Point p1 in c1.points)
            {
                foreach(Point p2 in c2.points)
                {
                    if(p1.inN.Contains(p2.id))
                    {
                        if (!c1.inC.Contains(c2.label) && c2.
                            label!=c1.label)
                        {
                            c1.inC.Add(c2.label);
                        }
```

```
                        }
                        if (p1.outN.Contains(p2.id))
                        {
                            if (!c1.outC.Contains(c2.label) && c2
                                .label != c1.label)
                            {
                                c1.outC.Add(c2.label);
                            }
                        }
                    }
                }
            }
        }

        static void GLPAm(ref List<Point> points, decimal[,] ModMat,
            int n, int m, ref decimal modularity)
        {
            List<Community> communities = new List<Community>();
            int communitiesChanges = 0;

            do
            {
                IteratePoints(ref points, m, ModMat, n, true, false,
                    false, ref modularity);
                CreateCommunities(points, ref communities);
                JoinCommunities(communities, ref points, n, m, ref
                    communitiesChanges);
            } while (communitiesChanges != 0);
        }

        static void JoinCommunities(List<Community> communities, ref
            List<Point> points, int n, int m, ref int
```

```
        communitiesChanges)
    {
        communitiesChanges = 0;
        List<DeltaQ> DeltaQs = new List<DeltaQ>();
        MyHeap<DeltaQ, decimal> DeltaQsHeap = new MyHeap<DeltaQ,
            decimal>(2*n);
     foreach (Community c1 in communities)
        {
            //FIND MAX inC AND MIN outC
            int MaxInC = 0;
            int MinOutC = 0;
            if (c1.inC.Count > 0)
            {
                MaxInC = c1.inC.Max();
            }
            if(c1.outC.Count > 0)
            {
                MinOutC = c1.outC.Min();
            }

            if(MaxInC > 0)
            {
                foreach (Community c2 in communities.Where(c => c
                    .label == MaxInC))
                {
                    int E = 0;
                    foreach (Point p1 in c1.points)
                    {
                        foreach (Point p2 in c2.points)
                        {
                            if (p1.outN.Contains(p2.id))
                            {
                                E++;
```

```
                    }
                }
            }

            decimal E_decimal = E;
            decimal m_decimal = m;
            decimal inS1 = c1.points[0].inS;
            decimal outS1 = c1.points[0].outS;
            decimal inS2 = c2.points[0].inS;
            decimal outS2 = c2.points[0].outS;
            decimal DeltaQ_ = E_decimal / m_decimal - ((
                outS1 * inS2 + inS1 * outS2) / (m_decimal
                * m_decimal));

            if (DeltaQ_ > 0)
            {
                DeltaQ dq = new DeltaQ();
                dq.value = DeltaQ_;
                dq.c1 = c1;
                dq.c2 = c2;
                DeltaQs.Add(dq);
                MyHeapMember<DeltaQ, decimal> HeapMember
                    = new MyHeapMember<DeltaQ, decimal>();
                HeapMember.Data = dq;
                HeapMember.Value = dq.value;
                DeltaQsHeap.InsertItem(HeapMember);
            }
        }
    }

    if(MinOutC != 0)
    {
        foreach (Community c2 in communities.Where(c => c
```

```
        .label == MinOutC))
    {
        int E = 0;
        foreach (Point p1 in c1.points)
        {
            foreach (Point p2 in c2.points)
            {
                if (p1.outN.Contains(p2.id))
                {
                    E++;
                }
            }
        }
        decimal E_decimal = E;
        decimal m_decimal = m;
        decimal inS1 = c1.points[0].inS;
        decimal outS1 = c1.points[0].outS;
        decimal inS2 = c2.points[0].inS;
        decimal outS2 = c2.points[0].outS;
        decimal DeltaQ_ = E_decimal / m_decimal - ((
            outS1 * inS2 + inS1 * outS2) / (m_decimal
            * m_decimal));

        if (DeltaQ_ > 0)
        {
            DeltaQ dq = new DeltaQ();
            dq.value = DeltaQ_;
            dq.c1 = c1;
            dq.c2 = c2;
            DeltaQs.Add(dq);
            MyHeapMember<DeltaQ, decimal> HeapMember
                = new MyHeapMember<DeltaQ, decimal>();
            HeapMember.Data = dq;
```

```
                    HeapMember.Value = dq.value;
                    DeltaQsHeap.InsertItem(HeapMember);
                }
            }
        }

    }
    DeltaQ maxDeltaQ = null;
    if(DeltaQs.Count > 0)
    {
        maxDeltaQ = DeltaQsHeap.ReturnMax().Data;
    }
    if(maxDeltaQ == null)
    {
        return;
    }

    bool ok = true;

    int labelForC1 = maxDeltaQ.c2.label;

    foreach (int label in maxDeltaQ.c1.inC)
    {
        if (label > labelForC1)
        {
            ok = false;
            break;
        }
    }
    if (ok)
    {
        foreach (int label in maxDeltaQ.c1.outC)
        {
```

```
                if (label < labelForC1)
                {
                    ok = false;
                    break;
                }
            }
        }
        if (ok)
        {
            communitiesChanges++;

            int inSnew = maxDeltaQ.c2.points[0].inS + maxDeltaQ.
                c1.points[0].inS;
            int outSnew = maxDeltaQ.c2.points[0].outS + maxDeltaQ
                .c1.points[0].outS;

            foreach (Point p in points)
            {
                if (p.label == maxDeltaQ.c1.label)
                {
                    p.label = labelForC1;
                }
            }
            foreach (Point p in points)
            {
                if (p.label == labelForC1)
                {
                    p.inS = inSnew;
                    p.outS = outSnew;
                }
            }
        }
        else
```

```
        {
            ok = true;

            int labelForC2 = maxDeltaQ.c1.label;
        foreach (int label in maxDeltaQ.c2.inC)
            {
                if (label > labelForC2)
                {
                    ok = false;
                    break;
                }
            }
            if (ok)
            {
                foreach (int label in maxDeltaQ.c2.outC)
                {
                    if (label < labelForC2)
                    {
                        ok = false;
                        break;
                    }
                }
            }
            if (ok)
            {
                communitiesChanges++;

                int inSnew = maxDeltaQ.c2.points[0].inS +
                    maxDeltaQ.c1.points[0].inS;
                int outSnew = maxDeltaQ.c2.points[0].outS +
                    maxDeltaQ.c1.points[0].outS;

                foreach (Point p in points)
```

```
                {
                    if (p.label == maxDeltaQ.c2.label)
                    {
                        p.label = labelForC2;
                    }
                }
                foreach (Point p in points)
                {
                    if (p.label == labelForC2)
                    {
                        p.inS = inSnew;
                        p.outS = outSnew;
                    }
                }
            }
        }
    }
    static bool Check(List<Point> points, List<int> column1,
        List<int> column2)
    {
    bool ok = true;
        for (int i = 0; i < column1.Count; i++)
        {
            int id1 = column1[i];
            int id2 = column2[i];
            int label1 = 0;
            int label2 = 0;
            foreach (Point p in points)
            {
                if (p.id == id1)
                {
                    label1 = p.label;
                }
```

```
                if (p.id == id2)
                {
                    label2 = p.label;
                }
            }
            if (label1 > label2)
            {
                ok = false;
                break;
            }
        }
        if (ok)
        {
             return true;
        }
        else
        {
            return false;
        }
    }

    static void SortPoints(List<Point> points)
    {
        for (int i = 0; i < points.Count - 1; i++)
        {
            for (int j = i + 1; j < points.Count; j++)
            {
                if (points[i].id > points[j].id)
                {
                    Point temp = points[i];
                    points[i] = points[j];
                    points[j] = temp;
                }
```

```
            }
        }
    }

    static void RandomizePoints(ref List<Point> points)
    {
        var rnd = new Random();
        points = points.OrderBy(x => rnd.Next()).ToList();
    }

    static Dq GetMaxDq(List<Dq> DQs)
    {
        if (DQs.Count == 0)
        {
            return null;
        }

        Dq maxDq = DQs[0];
        decimal maxValue = DQs[0].value;

        foreach (Dq Dq in DQs)
        {
            if (Dq.value > maxValue)
            {
                maxValue = Dq.value;
                maxDq = Dq;
            }
        }
        return maxDq;
    }

    static DeltaQ GetMaxDeltaQ(List<DeltaQ> DQs)
    {
```

```
        if(DQs.Count == 0)
        {
            return null;
        }

        DeltaQ maxDq = DQs[0];
        decimal maxValue = DQs[0].value;

        foreach (DeltaQ Dq in DQs)
        {
            if (Dq.value > maxValue)
            {
                maxValue = Dq.value;
                maxDq = Dq;
            }
        }
        return maxDq;
    }

    static void Wait()
    {
        System.Console.ReadLine();
    }

    static void PrintNM(int n, int m)
    {
        System.Console.Out.Write("NUMBER OF POINTS = " + n.
            ToString());
        System.Console.Out.Write("\nNUMBER OF CONNECTIONS = " + m
            .ToString());
    }

    static void PrintPoints(List<Point> points)
```

```
{
    System.Console.Out.Write("\n\nPOINTS:\n");
    System.Console.Out.Write("ID" + "\t" + "LABEL" + "\t" +
        "INS" + "\t" + "OUTS" + "\n");
    foreach (Point pp in points)
    {
        System.Console.Out.Write(pp.id.ToString() + "\t" + pp
            .label.ToString() + "\t" + pp.inS.ToString() + "\t
            " + pp.outS.ToString() + "\n");
    }
}

static void PrintAdMat(int[,] AdMat, int n)
{
    System.Console.Out.Write("\n\nADMAT:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            System.Console.Out.Write(AdMat[i, j].ToString() +
                " ");
        }
        System.Console.Out.Write("\n");
    }
}

static void PrintModMat(decimal[,] ModMat, int n)
{
    System.Console.Out.Write("\n\nMODMAT:\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
```

```
String ModMatString = (ModMat[i, j].ToString());
if (ModMat[i, j] < 0)
{
    if (ModMatString.Length > 4)
    {
        ModMatString = ModMatString.Substring(0,
            4);
        System.Console.Out.Write("␣" +
            ModMatString + "␣");
    }
    if (ModMatString.Length == 2)
    {
        System.Console.Out.Write("␣" +
            ModMatString + "␣␣");
    }
}
if (ModMat[i, j] == 0)
{
    System.Console.Out.Write("␣␣0␣␣␣");
}
if (ModMat[i, j] > 0)
{
    if (ModMatString.Length > 3)
    {
        ModMatString = ModMatString.Substring(0,
            3);
        System.Console.Out.Write("␣␣" +
            ModMatString + "␣");
    }
    if (ModMatString.Length == 2)
    {
        System.Console.Out.Write("␣␣" +
            ModMatString + "␣␣");
```

```
                        }
                    }
                }
                System.Console.Out.Write("\n");
            }
        }

        static void PrintDegs(List<Point> points, int n)
        {
            System.Console.Out.Write("\n\nDEGS:\n");
            for (int i = 0; i < n; i++)
            {
                System.Console.Out.Write(points[i].inDeg.ToString() +
                    "\t" + points[i].outDeg.ToString() + "\n");
            }
        }

    }

}
```

B.3 AORZ

Listing B.3.1: Point.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AZOR
{
    class Point
    {
        public int id { get; set; }
        public int label { get; set; }
        public int inDeg { get; set; }
        public int outDeg { get; set; }
    }
}
```

Listing B.3.2: Row.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AZOR
{
    class Row
    {
```

```
        public List<Point> points = new List<Point>();
        public decimal modularity { get; set; }
    }
}
```

Listing B.3.3: Iteration.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AZOR
{
    class Iteration
    {
        public int n { get; set; }
        public List<int> best = new List<int>();
        public List<Row> rows = new List<Row>();
    }
}
```

Listing B.3.4: Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AZOR
{
    class Program
```

```
{
    static void Main(string[] args)
    {
        List<int> column1 = new List<int>();
        List<int> column2 = new List<int>();
        int m = 0;
        int n = 0;

        List<Point> points = new List<Point>();
        List<Iteration> iterations = new List<Iteration>();
        decimal modularity = 0;



        ReadFile(ref column1, ref column2, ref m);



        ReadPoints(ref points, column1, column2, ref n);

        int[,] AdMat = new int[n, n];
        decimal[,] ModMat = new decimal[n, n];
        CalculateAdMat(ref AdMat, column1, column2);
        CalculateDegs(AdMat, n, ref points);
        CalculateModMat(ref ModMat, AdMat, points, n, m);

        Iteration iteration1 = new Iteration();
        iteration1.n = n;
        Row row1 = new Row();
        row1.points = points.ToList();
        row1.modularity = CalculateModularity(ModMat, points, n,
            m);
        modularity = row1.modularity;
        iteration1.rows.Add(row1);
        iteration1.best.Add(0);
```

```
            iterations.Add(iteration1);

            for (int i = n - 2; i >= 0; i--)
            {
                Iteration iteration = new Iteration();
                iteration.n = i+1;

                foreach (int b in iterations[iterations.Count-1].best
                    )
                {
                    Row row = iterations[iterations.Count-1].rows[b];
                    iteration.rows.Add(row);
                }

                foreach (Row r in iterations[iterations.Count-1].rows
                    )
                {
                    Row row = new Row();
                    foreach (Point p in r.points)
                    {
                        Point pp = new Point();
                        pp.id = p.id;
                        pp.label = p.label;
                        pp.inDeg = p.inDeg;
                        pp.outDeg = p.outDeg;
                        row.points.Add(pp);
                    }

                    row.points[i].label = row.points[i + 1].label;
                    row.modularity = CalculateModularity(ModMat, row.
                        points, n, m);
                    iteration.rows.Add(row);
                }
```

```
            foreach (Row r in iteration.rows)
            {
                if (r.modularity >= modularity)
                {
                    modularity = r.modularity;
                }
            }

            for (int j = 0; j < iteration.rows.Count; j++)
            {
                if (iteration.rows[j].modularity == modularity)
                {
                    iteration.best.Add(j);
                }
            }

            iterations.Add(iteration);
        }

        System.Console.Out.Write("\n\nModularity = " + modularity
            .ToString());
        Wait();

        foreach (Row r in iterations[iterations.Count - 1].rows)
        {
            if (r.modularity == modularity)
            {
                PrintPoints(r.points);
                Wait();
            }
        }
    }
```

```
static void ReadFile(ref List<int> column1, ref List<int>
    column2, ref int m)
{
    string directory = AppDomain.CurrentDomain.BaseDirectory
        + "\\Network.txt";
    string[] lines = System.IO.File.ReadAllLines(directory);

    int num_lines = 0;
    foreach (string line in lines)
    {
        string[] numbers = line.Split(' ');
        column1.Add(Int32.Parse(numbers[0]));
        column2.Add(Int32.Parse(numbers[1]));
        num_lines++;
    }

    m = num_lines;
}

static void ReadPoints(ref List<Point> points, List<int>
    column1, List<int> column2, ref int n)
{
    points.Clear();
    for (int i = 0; i < column1.Count; i++)
    {
        bool contains = false;
        foreach (Point point in points)
        {
            if (point.id == column1[i])
            {
                contains = true;
            }
```

```
            }
            if (!contains)
            {
                Point p = new Point();
                p.id = column1[i];
                p.label = column1[i];
                points.Add(p);
            }
        }

        for (int i = 0; i < column2.Count; i++)
        {
            bool contains = false;
            foreach (Point point in points)
            {
                if (point.id == column2[i])
                {
                    contains = true;
                }
            }
            if (!contains)
            {
                Point p = new Point();
                p.id = column2[i];
                p.label = column2[i];
                points.Add(p);
            }
        }

        SortPoints(points);
         n = points.Count;
    }
```

```
static void SortPoints(List<Point> points)
{
    for (int i = 0; i < points.Count - 1; i++)
    {
        for (int j = i + 1; j < points.Count; j++)
        {
            if (points[i].id > points[j].id)
            {
                Point temp = points[i];
                points[i] = points[j];
                points[j] = temp;
            }
        }
    }
}

static void CalculateAdMat(ref int[,] AdMat, List<int>
    column1, List<int> column2)
{
    for (int k = 0; k < column1.Count; k++)
    {
        int i = column1[k] - 1;
        int j = column2[k] - 1;
        AdMat[i, j] = 1;
    }
}

static void CalculateDegs(int[,] AdMat, int n, ref List<Point
    > points)
{
    foreach (Point p in points)
    {
        int inDeg = 0;
```

```
            int outDeg = 0;
            for (int i = 0; i < n; i++)
            {
                if (AdMat[i, p.id - 1] == 1)
                {
                    inDeg++;
                }
            }
            p.inDeg = inDeg;
            for (int i = 0; i < n; i++)
            {
                if (AdMat[p.id - 1, i] == 1)
                {
                    outDeg++;
                }
            }
            p.outDeg = outDeg;
        }
    }

    static void CalculateModMat(ref decimal[,] ModMat, int[,]
        AdMat, List<Point> points, int n, int m)
    {
        decimal m_decimal = m;
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                ModMat[i, j] = AdMat[i, j] - ((points[j].inDeg *
                    points[i].outDeg) / m_decimal);
            }
        }
    }
```

```
static decimal CalculateModularity(decimal[,] ModMat, List<
    Point> points, int n, int m)
{
    decimal modularity = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (points[i].label == points[j].label)
            {
                modularity = modularity + ModMat[points[i].id
                    - 1, points[j].id - 1];
            }
        }
    }
    decimal m_decimal = m;
    modularity = modularity / m_decimal;

    return modularity;
}

static void PrintPoints(List<Point> points)
{
    foreach (Point pp in points)
    {
        System.Console.Out.Write(pp.label.ToString() + "\t");
    }
    System.Console.Out.Write("\n");
}

static void Wait()
{
```

```
                System.Console.ReadLine();
            }
        }
}
```

Životopis

Suzana Antunović rođena je 26. travnja 1987. godine u Splitu. Pohađala je IV. opću gimnaziju *"Marko Marulić"* u Splitu. Godine 2005. upisuje Preddiplomski studij Matematike i informatike na Prirodoslovno–matematičkom fakultetu u Splitu koji završava 19. rujna 2008. obranom završnog rada pod nazivom *"p-adski brojevi"*. Iste godine upisuje Diplomski studij Matematike i informatike na Prirodoslovno–matematičkom fakultetu u Splitu koji završava 15. listopada 2010. obranom diplomskog rada pod nazivom *"Fraktalna geometrija"*. U listopadu 2011. upisuje Poslijediplomski studij Matematike na Prirodoslovno–matematičkom fakultetu u Zagrebu. Zaposlena je kao znanstveni novak na projektu "Identifikacije rizika i planiranje koritenja zemljita za ublaavanje posljedica klizanja i poplava u Hrvatskoj" na Fakultetu građevine, arhitekture i geodezije u Splitu od lipnja 2011.

Sudjelovanje na znanstvenim skupovima i konferencijama

1. MATH/CHEM/COMP 2008 (MCC 2008), Dubrovnik, Lipanj 2011. – sudjelovanje
2. CompleNet, Bolognja, Ožujak 2014 – sudjelovanje
3. Adriatic Conference on Graph Theory and Complexity, Split, Travanj 2014. – izlaganje "Generalized network descriptors"
4. Mediterranean School of Complex Networks, Sicilija, Rujan 2015 – izlaganje "Curriculum networks"
5. 1st Croatian Combinatorial Days, Zagreb, Rujan 2016 – izlaganje "Kurikularne mreže"
6. CompleNet, Dubrovnik, Ožujak 2017. – sudjelovanje

Znanstveni radovi

1. Kovač, Žarko; Platt, Trevor; **Antunović, Suzana**; Sathyendranath, Shubha; Morović, Mira; Gallegos, Charles, Extended formulations and analytic solutions for watercolumn production integrals, *Frontiers in Marine Science* **4** (2017), str. 1–16. doi:10.3389/fmars.2017.00163

2. **Antunović, Suzana**; Kokan, Tonći; Vojković, Tanja; Vukičević, Damir Generalised network descriptors, textitGlasnik matematiki **48 (2)** (2013), str. 211–230

3. Knezić, Snježana; Andrić, Ivo; Vlastelica, Goran; Miščević, Predrag; Bonacci, Ognjen; **Antunović, Suzana**, Hazard assessment methodology for Split case study, *Landslide and Flood Hazard Assessment, Abstract proceedings of the 3rd Japanese-Croatian Project Workshop* / Mihalić Arbanas S.; Arbanas Ž. (ur.)., Zagreb: City of Zagreb, Emergency Management Office, 2013. str. 26-27