UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Milena G. Filipović

KLASTEROVANJE GENA - ELEKTRONSKA LEKCIJA

master rad

Beograd, 2026.

Mentor:

prof. dr Jovana Kovačević
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mirjana Maljković Ružičić, docent
Univerzitet u Beogradu, Matematički fakultet

Nevena Ćirić, asistent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Zahvaljujem se mentorki na podršci, korisnim sugestijama i posvećenom vremenu. Miljanu na motivaciji i strpljenju. Najveću zahvalnost dugujem svojoj porodici i prijateljima koji su uvijek bili uz mene. Hvala vam na razumijevanju, podršci i strpljenju.

Ovaj rad posvećujem sestriću Nemanji Odžiću.

Naslov master rada: Klasterovanje gena - elektronska lekcija

Rezime: Klasterovanje je jedan od najvažnijih zadataka u savremenoj bioinformatici. Podaci o ekspresiji gena obrazuju velike matrice brojeva u kojima je cilj identifikovati funkcionalne grupe gena sa sličnim obrascima ponašanja. U ovom radu razvijena je elektronska lekcija posvećena klasterovanju gena, sa teorijskim objašnjenjem i vizualizacijom deset algoritama iz četiri klase: algoritmi zasnovani na centrima (*FarthestFirstTraversal*, *k-sredina / Lloyd*, *Soft k-sredina / EM*, *brute-force k-sredina*), algoritmi hijerarhijskog klasterovanja (*UPGMA*, aglomerativno, *DIANA*, divizivno), algoritmi zasnovani na gustini (*DBSCAN*) i grafovski algoritmi (*CAST*, *Louvain*, *Leiden*). Algoritmi se primenjuju na podatke o ekspresiji gena kvasca *Saccharomyces cerevisiae*, uz upoređivanje pristupa radi identifikovanja njihovih prednosti i ograničenja. Implementacija je urađena u programskom jeziku *Python*, isključivo uz standardnu biblioteku (moduli `math` i `random`), sa naglaskom na razumevanju matematičkih osnova: funkcije cilja, dokaze konvergencije, asimptotske složenosti i analizu mera bliskosti (euklidsko rastojanje, Pirsonov koeficijent korelacije). Korisnik može da prati izvršavanje algoritama korak po korak, čime lekcija služi kao edukativni alat u oblasti bioinformatike. Rad je realizovan kao interaktivna *Jupyter* sveska i javno dostupan kao projekat otvorenog koda.

Ključne reči: klasterovanje, ekspresija gena, bioinformatika, *k-sredina*, hijerarhijsko klasterovanje, *DBSCAN*, *Louvain*, elektronska lekcija, *Jupyter* sveska, *Saccharomyces cerevisiae*

Sadržaj

Glava 1

Uvod

Klasterovanje je postupak podele skupa entiteta na grupe (klastere) tako da entiteti unutar iste grupe budu što sličniji, a entiteti iz različitih grupa što različitiji. Jedna od njegovih primena u bioinformatici je analiza podataka o ekspresiji gena, gde je cilj prepoznati funkcionalno povezane grupe gena sa sličnim obrascima ponašanja.

Formalno, za skup tačaka $X = \{x_1, \ldots, x_n\}$, $x_i \in \mathbb{R}^d$, potrebno je pronaći skup particija $\mathcal{C} = \{C_1, \ldots, C_k\}$ takva da je $\bigcup_i C_i = X$ i $C_i \cap C_j = \emptyset$ za $i \neq j$. Problem je *NP-težak* (za $k \geq 2$), pa se u praksi koriste heuristike.

Cilj rada je da se kroz interaktivnu *elektronsku lekciju* prikaže deset algoritama klasterovanja iz četiri klase, algoritmi zasnovani na centrima (*FarthestFirstTraversal*, *k-sredina* (*Lloyd*), *Soft k-sredina*, *brute-force k-sredina*), algoritmi hijerarhijskog klasterovanja (*UPGMA*, *DIANA*), algoritmi zasnovani na gustini (*DBSCAN*) i grafovski algoritmi (*CAST*, *Louvain*, *Leiden*), primenjenih na podatke o ekspresiji gena kvasca *Saccharomyces cerevisiae*, sa naglaskom na razumevanje matematičkih osnova (funkcije cilja, asimptotske složenosti, mere bliskosti). Lekcija je realizovana kao interaktivna *Jupyter* sveska koja omogućava praćenje izvršavanja algoritama korak po korak, čime povezuje teoriju iz oblasti *bioinformatike* sa neposrednom primenom [10].

U glavi 2 opisuje se biološki kontekst i podaci, u glavi 3 mere bliskosti između tačaka, a u glavi 4 algoritmi klasterovanja iz četiri klase, sa funkcijom cilja, pseudokodom i analizom složenosti svakog. U glavi 5 daju se evaluacija klasterovanja, uporedni pregled i eksperimentalno poređenje algoritama, a u glavi 6 implementacija i korišćenje elektronske lekcije, uz prikaz rada algoritama korak po korak. U glavi 7 izvode se zaključci i razmatraju pravci daljeg rada.

Glava 2

Biološki kontekst

Da bi se razumeo zadatak klasterovanja gena, najpre treba objasniti šta su ulazni podaci (vrednosti genske ekspresije) i odakle potiču.

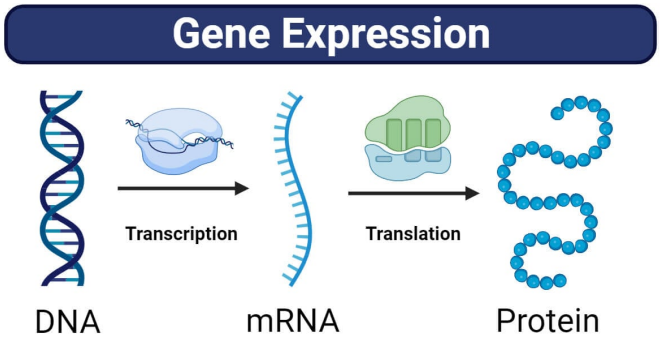
2.1 Od gena do proteina

Nasledna informacija svakog organizma zapisana je u molekulu dezoksiribonukleinske kiseline (DNK). DNK je podeljena na *gene* koji nose uputstvo za sintezu pojedinačnih proteina, glavnih gradivnih i funkcionalnih molekula ćelije. Put od gena do proteina opisuje *centralna dogma molekularne biologije* i odvija se u dva koraka:

$$\text{DNK} \xrightarrow{\text{transkripcija}} \text{mRNK} \xrightarrow{\text{translacija}} \text{protein.}$$

U *transkripciji* se sa gena prepisuje informaciona ribonukleinska kiselina, mRNK (engl. *messenger RNA , mRNA*), koja prenosi genetsku informaciju za sintezu proteina do mesta sinteze proteina. U *translaciji* se niz nukleotida mRNK čita u trojkama, *kodonima* prema genetskom kodu svaki kodon određuje jednu aminokiselinu[1], pa se nadovezivanjem aminokiselina gradi protein. Azbuka DNK ima samo četiri slova (nukleotidi A, C, G, T), a kako je $4^3 = 64 > 20$, kodoni mogu da kodiraju svih 20 standardnih aminokiselina koje izgrađuju proteine. Genetski kod je pri tome gotovo isti kod svih živih organizama [4, 6]. Sam genetski kod, pridruživanje kodona aminokiselinama - prikazan je u tabeli 2.1, a ceo tok informacija od DNK do proteina na slici 2.1.

[1] *Aminokiseline* su osnovne gradivne jedinice proteina.



Slika 2.1: Ekspresija gena: tok genetskih informacija od DNK, preko transkripcije u mRNK, do translacije u protein. Preuzeto sa [13].

2.2 Ekspresija gena

Sve ćelije jednog organizma sadrže istu DNK, ali se ne ponašaju isto: u svakom trenutku koriste samo deo svojih gena. *Ekspresija gena* je proces, opisan centralnom dogmom, u kome se informacija sadržana u genu prevodi u funkcionalni molekul, najčešće protein. Koliko je neki gen *aktivan*, koliko se jako eksprimira, razlikuje se od ćelije do ćelije i menja se tokom vremena.

Pošto se količina proteina teško meri direktno, kao mera aktivnosti gena koristi se količina informacione RNK (mRNK): što je gen aktivniji, to je u ćeliji prisutno više njegovih mRNK molekula, a količina mRNK je približno proporcionalna količini sintetisanog proteina.

2.3 Matrica genske ekspresije

Praćenjem nivoa ekspresije svih n gena u m kontrolnih tačaka sa obe strane diauksične promene[2] dobija se matrica koju definišemo na sledeći način.

Definicija 2.1 (Matrica genske ekspresije)**.** *Matrica genske ekspresije* E je matrica dimenzije $n \times m$ (gde je n broj gena, a m broj kontrolnih tačaka) čiji element $E_{i,j}$ predstavlja nivo ekspresije gena i u kontrolnoj tački j. i-ti red matrice E, vektor $(e_1, \ldots, e_m)$, naziva se *vektorom ekspresije* gena i.

Kontrolne tačke, odnosno kolone matrice, odgovaraju različitim uslovima pod kojima se ekspresija meri, pa zavise od pitanja koje se proučava. To mogu biti uzastopni

[2]U kontekstu analize ekspresije gena, *diauksična promena* je prelaz između dve faze rasta ćelije, kada se menja metabolički program. Detaljnije u odeljku 2.4.

Aminokiselina	**Kodoni**
Ala (A)	GCU, GCC, GCA, GCG
Arg (R)	CGU, CGC, CGA, CGG, AGA, AGG
Asn (N)	AAU, AAC
Asp (D)	GAU, GAC
Cys (C)	UGU, UGC
Gln (Q)	CAA, CAG
Glu (E)	GAA, GAG
Gly (G)	GGU, GGC, GGA, GGG
His (H)	CAU, CAC
Ile (I)	AUU, AUC, AUA
Leu (L)	UUA, UUG, CUU, CUC, CUA, CUG
Lys (K)	AAA, AAG
Met (M)	AUG
Phe (F)	UUU, UUC
Pro (P)	CCU, CCC, CCA, CCG
Ser (S)	UCU, UCC, UCA, UCG, AGU, AGC
Thr (T)	ACU, ACC, ACA, ACG
Trp (W)	UGG
Tyr (Y)	UAU, UAC
Val (V)	GUU, GUC, GUA, GUG
stop	UAA, UAG, UGA

Tabela 2.1: Standardni genetski kod: svaka od 20 aminokiselina (data troslovnom i jednoslovnom oznakom) kodirana je jednim ili sa više kodona, dok tri kodona označavaju kraj (*stop*). Odatle redundantnost koda: neke aminokiseline imaju i po šest kodona (npr. Leu, Ser, Arg), a neke samo jedan (Met, Trp). U DNK umesto baze U stoji T. Prilagođeno prema [4].

vremenski trenuci tokom nekog procesa (kao u ovom radu, gde se prati diauksična promena), različita tkiva ili tipovi ćelija, uzorci uzeti pre i posle nekog tretmana (na primer davanja leka) ili obolele naspram zdravih ćelija (recimo tumorske naspram zdravih). Upravo zbog te raznovrsnosti su matrice ekspresije sveprisutne u biološkim i biomedicinskim istraživanjima. Tako je, na primer, analizom ekspresije 70 humanih gena razvijen dijagnostički test *MammaPrint* za procenu rizika od povratka raka dojke [6]. U ovom radu kontrolne tačke su sedam vremenskih trenutaka oko diauksične promene.

Geni sa sličnim vektorima ekspresije najčešće su *koregulisani*, tj. aktiviraju se

i inhibiraju zajedno, što ukazuje na to da ih kontroliše isti transkripcioni faktor[3] i, posredno, na zajedničku biološku funkciju. Na ovoj pretpostavci počiva klasterovanje gena: ako je vektor ekspresije gena sa nepoznatom funkcijom sličan vektoru ekspresije gena sa poznatom funkcijom, može se pretpostaviti da ti geni obavljaju srodne funkcije.

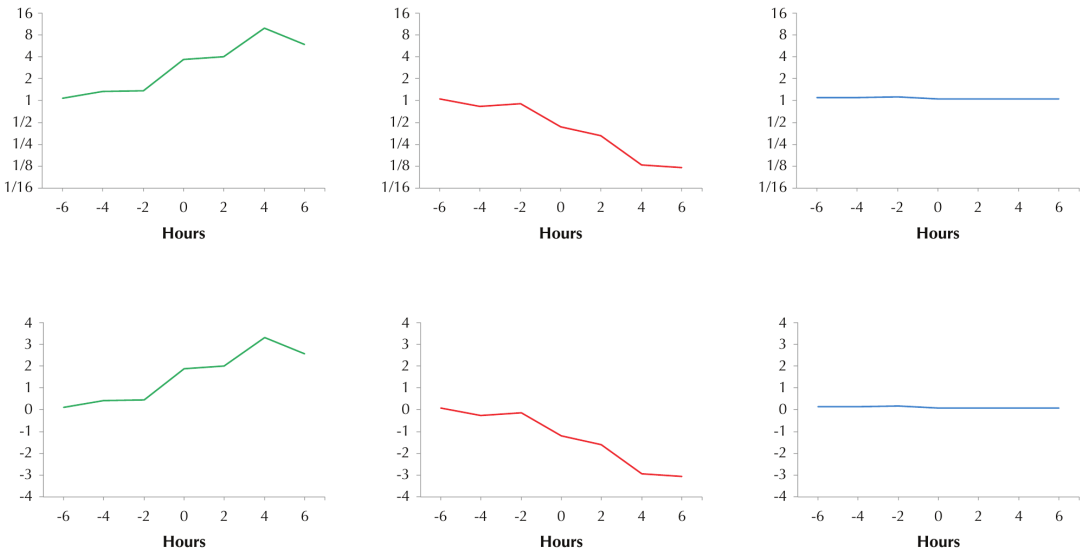
2.4 Diauksična promena kod kvasca

Diauksična promena (engl. *diauxic shift*) je prelazak kvasca sa jednog izvora energije na drugi tokom rasta. Dok u podlozi ima glukoze, kvasac je brzo razgrađuje *fermentacijom*, pri čemu nastaje etanol (alkohol). Kada se glukoza iscrpi, posle kratkog zastoja u rastu kvasac menja metabolizam i počinje da koristi nagomilani etanol putem *respiracije* (uz kiseonik). Taj nagli prelaz sa fermentacije na respiraciju (odatle prefiks *di-*, jer rast ima dve faze) prati drastična preraspodela genske ekspresije, zbog čega je pogodan za proučavanje. Kao referentni skup podataka u ovom radu koristi se klasičan eksperiment o diauksičnoj promeni kod kvasca *Saccharomyces cerevisiae*. Ovaj kvasac je, u zavisnosti od konteksta, poznat i pod nazivima pekarski kvasac, pivski (vinski) kvasac ili vinska kvasnica. Za ovu vrstu su DeRisi i saradnici [6] izmerili ekspresiju oko 6400 gena u sedam kontrolnih tačaka (u satima $-6, -4, -2, 0, +2, +4, +6$, gde sat 0 označava trenutak diauksične promene). Na osnovu biološke funkcije, geni se grubo dele u tri grupe: *regulisane na gore* (engl. *up-regulated*, čija ekspresija raste tokom vremena, u našem slučaju geni karakteristični za fazu respiracije), *regulisane na dole* (engl. *down-regulated*, čija ekspresija opada tokom vremena, u našem slučaju geni karakteristični za fazu fermentacije) i *nepromenjene* (stabilne, engl. *housekeeping*, čija ekspresija ostaje približno ista tokom vremena). Ova podela služi kao referenca pri proveri rezultata algoritama.

2.5 Logaritamski odnos promene ($\log_2$ *fold change*)

Svaka vrednost u vektoru ekspresije nije apsolutan broj mRNK molekula, nego *odnos* između izmerenog nivoa ekspresije gena i njegovog *baznog* nivoa, tj. nivoa na

[3] *Transkripcioni faktor* je protein koji reguliše proces transkripcije, tj. prepisivanje DNK u RNK, vezujući se za specifične sekvence DNK i tako pojačavajući (kao aktivator) ili prigušujući (kao represor) prepisivanje ciljnih gena. Jedan faktor obično deluje na više gena istovremeno [6].



Slika 2.2: Vektori ekspresije tri gena. Gore: sirovi odnos prema baznom nivou. Dole: posle uzimanja logaritma osnove 2 (vrednosti centrirane oko nule). Preuzeto iz [6].

prvoj kontrolnoj tački, pre diauksične promene. Zato je na prvoj tački taj odnos (približno) jednak 1: vrednost veća od 1 znači da je ekspresija u odnosu na početak porasla (na primer 2 znači udvostručenje), a vrednost manja od 1 da je opala (na primer 1/2 znači prepolovljenje) [6].

Ovakav odnos je nezgodan za poređenje jer mu je skala asimetrična: pojačanja se prostiru od 1 do ∞, a prigušenja od 1 do 0, pa udvostručenje (2) i prepolovljenje (1/2) nisu jednako udaljeni od 1. Zato se nad odnosom uzima logaritam osnove 2:

$$\text{log2FC}_{i,j} = \log_2 \frac{E_{i,j}}{E_{i,\text{ref}}},$$

gde je $E_{i,j}$ nivo ekspresije gena i u kontrolnoj tački j, a $E_{i,\text{ref}}$ njegova *ekspresija u referenci*, nivo istog gena u baznom (polaznom) uzorku, pre diauksične promene, sa kojim se porede sva ostala merenja.

Logaritmovanjem se dobija veličina centrirana oko nule i sa simetričnom skalom, kao što je prikazano na slici 2.2: udvostručenje ekspresije daje $+1$, prepolovljenje -1, a nepromenjena ekspresija 0. Pozitivne vrednosti tako označavaju pojačanu, a negativne smanjenu ekspresiju u odnosu na bazni nivo. Svaki gen je na kraju predstavljen vektorom $\log_2$ vrednosti kroz sedam kontrolnih tačaka, i upravo se nad tim vrednostima sprovodi klasterovanje. Ovako logaritmovana matrica ekspresije

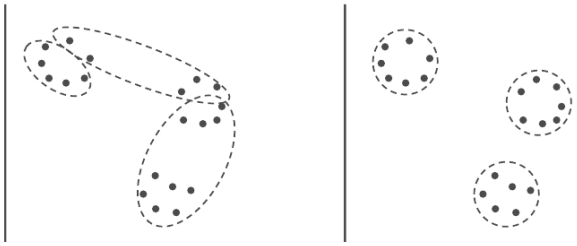
Gene	Expression Vector						
YLR361C	**0.14**	0.03	-0.06	0.07	-0.01	-0.06	-0.01
YMR290C	0.12	-0.23	-0.24	-1.16	-1.40	-2.67	**-3.00**
YNR065C	-0.10	**-0.14**	-0.03	-0.06	-0.07	-0.14	-0.04
YGR043C	-0.43	-0.73	-0.06	-0.11	-0.16	**3.47**	2.64
YLR258W	0.11	0.43	0.45	1.89	2.00	**3.32**	2.56
YPL012W	0.09	-0.28	-0.15	-1.18	-1.59	-2.96	**-3.08**
YNL141W	-0.16	-0.04	-0.07	-1.26	-1.20	-2.82	**-3.13**
YJL028W	-0.28	-0.23	-0.19	-0.19	**-0.32**	-0.18	-0.18
YKL026C	-0.19	-0.15	0.03	0.27	0.54	**3.64**	2.74
YPR055W	0.15	0.15	**0.17**	0.09	0.07	0.09	0.07

Slika 2.3: Matrica genske ekspresije posle logaritmovanja ($\log_2$): deset gena kvasca kroz sedam kontrolnih tačaka. Boja gena označava grupu prema ponašanju ekspresije: zeleno su regulisani na gore (vrednosti rastu ka kraju), crveno regulisani na dole (vrednosti opadaju), a plavo nepromenjeni (vrednosti ostaju blizu nule). Tri gena koja se u nastavku prate kroz ceo rad su YLR258W (zeleno), YPL012W (crveno) i YPR055W (plavo). Preuzeto iz [6].

prikazana je na slici 2.3, gde je u svakom redu podebljana ona vrednost sa najvećim apsolutnim odstupanjem od nule, tj. trenutak u kome se ekspresija tog gena najjače promenila u odnosu na bazni nivo (najviše porasla ili opala).

2.6 Klasterovanje gena

Cilj klasterovanja je da se skup gena podeli na grupe tako da geni unutar iste grupe imaju što sličnije vektore ekspresije, a geni iz različitih grupa što različitije. Time se velika matrica ekspresije svodi na nekoliko karakterističnih obrazaca ponašanja, što olakšava biološku interpretaciju i otkrivanje funkcionalno povezanih gena. Princip dobrog klasterovanja ilustrovan je na slici 2.4.



Slika 2.4: Princip dobrog klasterovanja. Levo: podela koja ga ne zadovoljava. Desno: podela u kojoj je svaka instanca grupisana sa njoj najbližim instancama. Preuzeto iz [6].

Glava 3

Mere bliskosti

Pre opisa algoritama treba definisati kako se meri blizina dve *tačke* (u opštem slučaju to su entiteti koje klasterujemo, a u analizi ekspresije gena vektori ekspresije pojedinačnih gena), tj. koliko su dve tačke međusobno slične ili udaljene.

3.1 Euklidsko rastojanje i Pirsonov koeficijent korelacije

Definicija 3.1 (Euklidsko rastojanje)**.** Za tačke $a = (a_1, \ldots, a_d)$ i $b = (b_1, \ldots, b_d)$ iz $\mathbb{R}^d$, *euklidsko rastojanje* između a i b definiše se kao

$$d(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}.$$

Euklidsko rastojanje meri geometrijsku udaljenost dve tačke u prostoru i osetljivo je i na skaliranje i na translaciju podataka. Izračunavanje jednog rastojanja je vremenske složenosti $O(d)$.

Definicija 3.2 (Pirsonov koeficijent korelacije)**.** Za vektore $x, y \in \mathbb{R}^n$ sa srednjim vrednostima $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ i $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$, *Pirsonov koeficijent korelacije* je

$$r(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}}.$$

Vrednost $r(x, y)$ pripada intervalu $[-1, 1]$ i meri *linearnu zavisnost* između dva vektora. Za razliku od euklidskog rastojanja, ova mera nije osetljiva ni na skaliranje ni na translaciju, pa je prirodna za koregulisane gene. Korelacija se u rastojanje

pretvara transformacijom $d = 1 - r$. Na taj način dobijamo veličinu koju zovemo *Pirsonovo rastojanje* [6]. Sličnost dva vektora ekspresije može se meriti i *skalarnim proizvodom* $\langle x, y \rangle = \sum_{i=1}^{n} x_i \, y_i$. On je veći kada vektori istovremeno imaju velike vrednosti na istim mestima, ali za razliku od Pirsonovog koeficijenta jeste osetljiv na skaliranje [6].

Mnogi algoritmi klasterovanja ne rade nad samim tačkama (pojedinačnim entitetima, u našem slučaju genima), nego nad izvedenom matricom među svim parovima tačaka: matricom *rastojanja* (hijerarhijski algoritmi *UPGMA* i *DIANA*) ili matricom *sličnosti* (grafovski algoritmi *CAST*, *Louvain* i *Leiden*).

Definicija 3.3 (Matrica rastojanja i matrica sličnosti)**.** Za skup od n tačaka, *matrica rastojanja* D je matrica dimenzije $n \times n$ čiji je element $D_{i,j} = d(x_i, x_j)$ jednak rastojanju između tačaka i i j. Simetrična je i ima nule na dijagonali. *Matrica sličnosti* S je matrica dimenzije $n \times n$ čiji element $S_{i,j}$ izražava koliko su tačke i i j slične (veće vrednosti znače veću sličnost). Najčešće se dobija iz Pirsonovog koeficijenta korelacije, $S_{i,j} = r(x_i, x_j)$, ili iz rastojanja, na primer $S_{i,j} = 1 - d_{i,j}/d_{\max}$.

Algoritmi hijerarhijskog klasterovanja (*UPGMA*, *DIANA*) rade nad matricom rastojanja D, a *CAST* i grafovski algoritmi nad matricom sličnosti S.

Glava 4

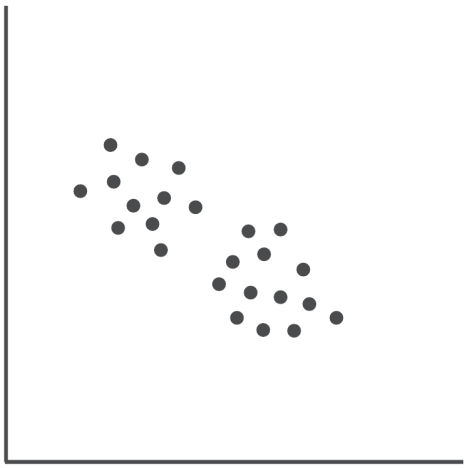
Algoritmi klasterovanja

Ne postoji jedinstveni i najbolji algoritam klasterovanja, svaki polazi od drugačije definicije klastera. Kod jedne vrste algoritama klaster je skup tačaka koje imaju zajedničkog centra, tako da je svaka tačka bliža centru svog klastera nego centrima ostalih klastera (pri čemu se bliskost meri izabranom merom, najčešće euklidskim rastojanjem), kod druge niz grupa koje se postepeno objedinjuju ili razdvajaju, kod treće gusto naseljena oblast oko koje se ne nalaze druge tačke, a kod četvrte čvrsto povezana zajednica u grafu sličnosti. Neki algoritmi zahtevaju da se broj klastera k zna unapred, dok ga drugi sami otkrivaju iz strukture podataka. U ovoj glavi opisani su algoritmi grupisani u četiri klase, kako bi se uporedile njihove prednosti i ograničenja, jer isti skup gena različiti algoritmi mogu podeliti na različite načine. Uz svaki algoritam navode se funkcija cilja, pseudokod i ponašanje na podacima o ekspresiji gena.

4.1 Algoritmi zasnovani na centrima

Idealno bi bilo da svaka podela zadovoljava sledeće prirodno pravilo, koje nazivamo *pravilom bliskosti*: svake dve tačke iz istog klastera treba da budu bliže jedna drugoj nego bilo koje dve tačke iz različitih klastera. Za mnoge skupove podataka, međutim, takva podela uopšte ne postoji, pa se do dobre podele ne može doći direktnim traženjem grupa [6]. Primer je skup tačaka na slici 4.1: oko ga bez oklevanja deli na dve grupe, ali unutar jedne izdužene grupe njeni krajevi su međusobno udaljeniji nego od pojedinih tačaka susedne grupe, pa nijedna podela na dva klastera ne zadovoljava pravilo bliskosti. Isto se sreće i kod samih podataka o ekspresiji gena: profili ekspresije često čine kontinuum u kome geni postepeno prelaze iz jednog ob-

rasca ponašanja u drugi, bez oštre granice na kojoj bi svaki par gena unutar grupe bio sličniji nego bilo koji par gena iz različitih grupa [6].



Slika 4.1: Skup tačaka koji se vizuelnim posmatranjem lako deli na dve grupe, ali za koji nijedna podela na dva klastera ne zadovoljava pravilo bliskosti: neke tačke iz iste grupe udaljenije su od pojedinih tačaka iz druge grupe. Preuzeto iz [6].

Zato algoritmi zasnovani na centrima menjaju postavku problema klasterovanja: umesto da odmah traže grupe, oni biraju k reprezentativnih tačaka, *centara*, od kojih svaka predstavlja po jedan klaster. *Centar* je tačka koja apstrahuje svoj klaster. Najčešće je to prosek (centar mase) njegovih tačaka i ne mora biti nijedna od datih tačaka, a klaster čine sve tačke kojima je upravo taj centar najbliži. Time se klasterovanje svodi na optimizacioni problem: treba izabrati centre tako da tačke budu što bliže svojim centrima. Funkcija cilja koju ovi algoritmi nastoje da minimizuju jeste *kvadratna greška distorzije*, koja meri koliko su tačke u proseku udaljene od najbližeg centra.

Definicija 4.1 (Kvadratna greška distorzije)**.** Za skup tačaka $X = \{x_1, \ldots, x_n\}$ i skup centara C, *kvadratna greška distorzije* je prosečno kvadratno rastojanje svake tačke do njoj najbližeg centra:

$$\text{Distortion}(X, C) = \frac{1}{n} \sum_{j=1}^{n} \min_{c \in C} \|x_j - c\|^2.$$

Kvadriranje se vrši zato što se time jače kažnjava grupisanje tačaka koje su daleko od svog centra: tačka dvostruko udaljenija doprinosi greški četiri puta više, što vodi

tome da se u sledećoj iteraciji centri pomere ka sredini grupe. Za razliku od najvećeg rastojanja (*MaxDistance*), koje gleda samo jednu, najudaljeniju tačku, kvadratna distorzija sabira doprinose svih tačaka, pa je znatno manje osetljiva na atipične tačke (izolovane tačke, engl. *outliers*), što je poželjno kod bioloških podataka, gde atipične tačke često potiču od grešaka u merenju [6]. Upravo zbog kvadrata najbolji centar postaje obična aritmetička sredina (prosek) tačaka u klasteru (princip najmanjih kvadrata), koju je lako odrediti. Ovo tvrđenje je poznato kao teorema o centru mase i biće formulisano u odeljku o *k-sredina* algoritmu [6]. Pošto je nalaženje najboljeg mogućeg rešenja NP-težak problem (za $k \geq 2$), u praksi se koriste heuristike koje nalaze dovoljno dobro rešenje.

4.1.1 *FarthestFirstTraversal*

Kod ovog pristupa kvalitet podele meri se *najvećim* rastojanjem neke tačke do njoj najbližeg centra. Za skup centara C neka je $d(x, C) = \min_{c \in C} \|x - c\|$ rastojanje tačke x do najbližeg centra. Tada je

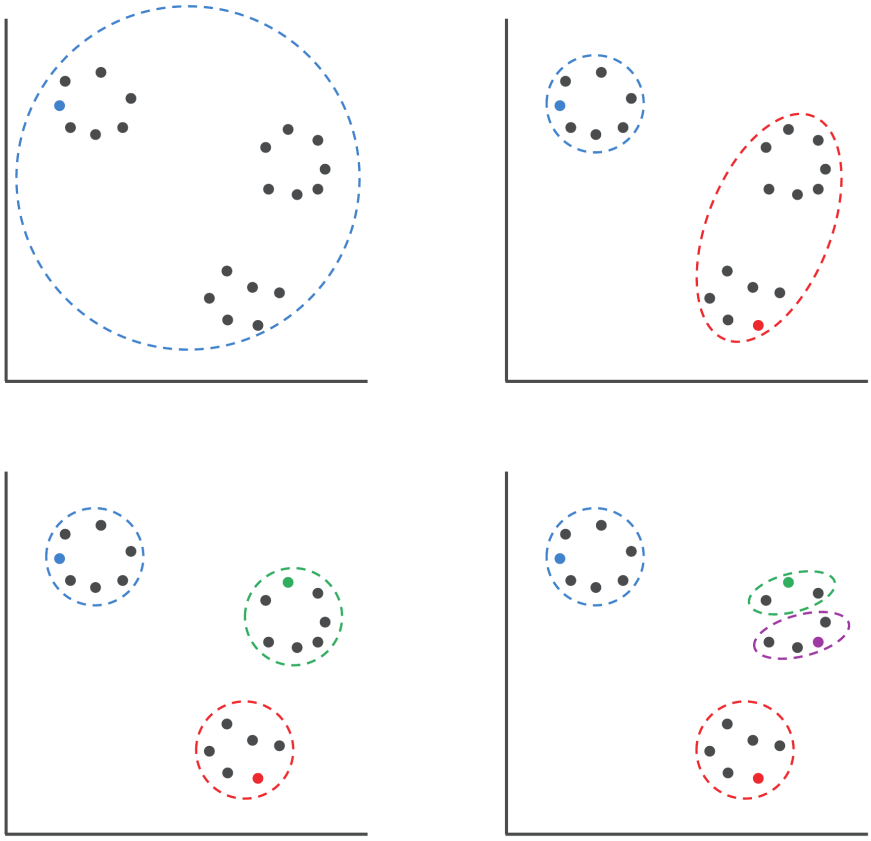
$$\text{MaxDistance}(X, C) = \max_{x \in X} d(x, C).$$

Definicija 4.2 (Problem k-centara)**.** Za skup tačaka X i ceo broj k, problem k-centara podrazumeva pronalaženje skupa od k centara C koji minimizuje $\mathit{MaxDistance}(X, C)$.

Ovaj problem je NP-težak, pa se u praksi koristi heuristika *FarthestFirstTraversal*: prvi centar bira se proizvoljno među tačkama, a zatim se u svakom koraku za novi centar uzima ona tačka iz X koja je najudaljenija od već izabranih centara, sve dok se ne izabere k centara [6]. Izbor centara korak po korak prikazan je na slici 4.2. Ceo postupak prikazan je u Algoritmu 1.

Algoritam 1 $\mathit{FarthestFirstTraversal}(X, k)$

$C \leftarrow \{$ proizvoljno izabrana tačka iz $X \}$
while $|C| < k$ **do**
    izaberi tačku $x \in X$ koja maksimizuje $d(x, C)$
    dodaj x u skup C
end while
return C



Slika 4.2: Izbor centara heuristikom *FarthestFirstTraversal* za $k = 4$. Gore levo: prva tačka (plava) bira se proizvoljno i sve tačke čine jedan klaster. Gore desno: za drugi centar (crveni) uzima se tačka najudaljenija od prvog. Dole levo: treći centar (zeleni) je tačka sa najvećim rastojanjem do prva dva centra. Dole desno: četvrti centar (ljubičasti). Preuzeto iz [6].

Ova heuristika je računski efikasna. Ako sa n označimo broj tačaka, sa k broj centara, a sa d dimenziju prostora (broj koordinata svake tačke, pri čemu je izračunavanje jednog euklidskog rastojanja složenosti $O(d)$), tada - uz čuvanje najmanjeg rastojanja svake tačke do tekućih centara, dodavanje svakog novog centra zahteva samo po jedno računanje rastojanja po tački, tj. $O(nd)$ po koraku, pa je kroz k koraka ukupna složenost $O(nkd)$. Uz to, garantuje rešenje koje nije više od dvostruko lošije od optimalnog, $MaxDistance(X, C) \leq 2\, MaxDistance(X, C_{\text{opt}})$.

Teorema 4.1. *Neka je C skup centara koji vrati* FarthestFirstTraversal(X, k)*, a C_{opt} optimalno rešenje problema k-centara. Tada je*

$$MaxDistance(X, C) \leq 2\, MaxDistance(X, C_{\text{opt}}).$$

Dokaz. Označimo $r = \mathit{MaxDistance}(X, C)$ i $r^* = \mathit{MaxDistance}(X, C_{\text{opt}})$. Cilj je $r \leq 2r^*$. Ako je $r = 0$, tvrđenje trivijalno važi, pa pretpostavimo $r > 0$.

Neka su $c_1, \ldots, c_k$ centri redom kako ih algoritam bira i neka je $C_{j-1} = \{c_1, \ldots, c_{j-1}\}$ skup centara pre izbora c_j. Po definiciji koraka je

$$c_j = \arg\max_{x \in X} d(x, C_{j-1}), \qquad r_j := d(c_j, C_{j-1}) = \max_{x \in X} d(x, C_{j-1}).$$

Dodavanjem centara skup C_{j-1} raste, pa je za svaku fiksiranu tačku x vrednost $d(x, C_{j-1})$ nerastuća po j. Samim tim je i niz maksimuma nerastući:

$$r_2 \geq r_3 \geq \cdots \geq r_k \geq r_{k+1}, \qquad r_{k+1} := \max_{x \in X} d(x, C_k) = r.$$

Neka je $z \in X$ tačka koja dostiže maksimum, $d(z, C) = r$. Tvrdimo da je $k + 1$ tačaka $\{c_1, \ldots, c_k, z\}$ *u parovima* na rastojanju bar r:

- za $i < j$ je $c_i \in C_{j-1}$, pa je $\|c_j - c_i\| \geq d(c_j, C_{j-1}) = r_j \geq r_{k+1} = r$.
- za svako i je $\|z - c_i\| \geq d(z, C) = r$.

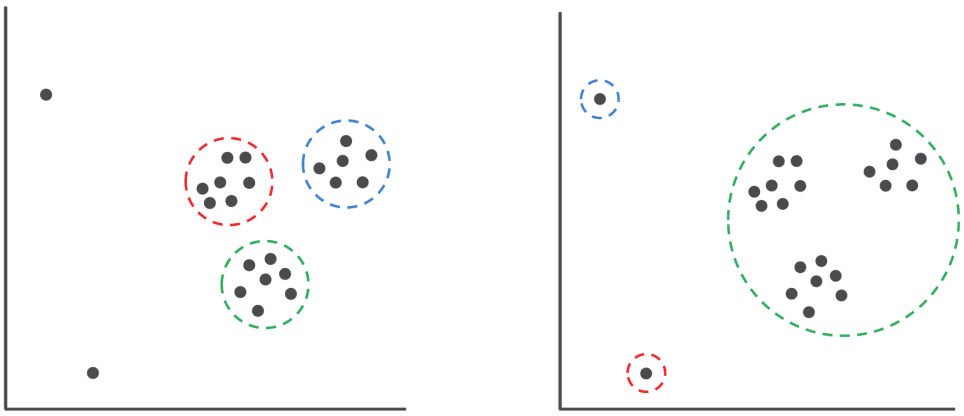
Optimalno rešenje deli X na k klastera, po jedan za svaki centar iz C_{opt}, pri čemu je svaka tačka na rastojanju najviše r^* od svog optimalnog centra. Kako imamo $k+1$ tačaka, a samo k optimalnih centara, po Dirihleovom principu[1] dve od njih, recimo p i q, pripadaju istom optimalnom klasteru, tj. imaju isti najbliži optimalni centar $c^* \in C_{\text{opt}}$. Tada je, po nejednakosti trougla,

$$r \leq \|p - q\| \leq \|p - c^*\| + \|c^* - q\| \leq r^* + r^* = 2r^*,$$

čime je $r \leq 2r^*$. □

Ipak, u analizi genske ekspresije ovaj algoritam se retko koristi: pošto zavisi isključivo od najvećeg rastojanja, na nju presudno utiču atipične tačke, koje često potiču od eksperimentalnih grešaka. Takva tačka postaje centar sopstvenog jednočlanog klastera, pa se sve ostale tačke sabijaju u preostale klastere, kao što je prikazano na slici 4.3.

[1] Dirihleov princip (engl. *pigeonhole principle*): ako se $m + 1$ objekata rasporedi u m kutija, onda bar jedna kutija sadrži bar dva objekta. Ovde se $k + 1$ tačaka raspoređuje u k optimalnih klastera [11].

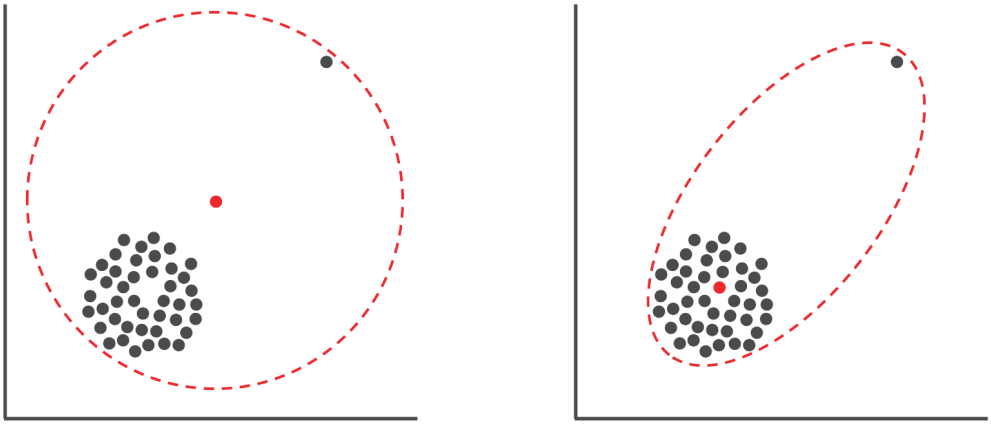


Slika 4.3: Slučaj u kome *FarthestFirstTraversal* zakaže. Levo: tri jasna klastera i dve atipične tačke (gore levo i dole levo). Desno: pošto se svaki novi centar bira kao tačka najudaljenija od već izabranih, za $k = 3$ obe atipične tačke postaju centri sopstvenih jednočlanih klastera, pa se sva tri stvarna klastera stope u jedan. Preuzeto iz [6].

Zbog toga se prelazi na meru koja uzima u obzir tipično, a ne najveće odstupanje, kvadratnu grešku distorzije.

4.1.2 *K-sredina* (*Lloyd*)

Algoritam *k-sredina* (engl. *k-means*) minimizuje kvadratnu grešku distorzije, koja uzima u obzir sva rastojanja, a ne samo najveće, pa je znatno manje osetljiva na atipične tačke. Razlika u odnosu na problem k-centara najbolje se vidi na položaju samog centra, što ilustruje slika 4.4: kod k-centara centar se bira tako da se najveće rastojanje do bilo koje tačke klastera smanji, pa ga jedna jedina atipična tačka može znatno pomeriti. Kod *k-sredina* atipična tačka učestvuje ravnopravno sa svim ostalim tačkama, pa je njen uticaj na centar mnogo manji, što je poželjno kod bioloških podataka: kao što je već bilo reči u potpoglavlju 4.1.1, njihove atipične tačke često potiču od grešaka u merenju.



Slika 4.4: Uticaj atipične tačke na izbor centra. Levo (problem k-centara): pošto se gleda najveće rastojanje, centar (crveni) se pomera ka usamljenoj tački, daleko od gustog dela klastera. Desno (problem *k-sredina*): centar ostaje u gustom delu klastera, jer atipična tačka doprinosi jednako kao i svaka druga tačka. Preuzeto iz [6].

Definicija 4.3 (Problem k-sredina)**.** Za skup tačaka X i ceo broj k, tražimo skup od k centara C koji minimizuje Distortion(X, C).

Problem je NP-težak za $k > 1$. Za $k = 1$, međutim, rešenje je jednostavno: jedini centar koji minimizuje distorziju je *centar mase* (težište) skupa, tačka čija je i-ta koordinata jednaka proseku i-tih koordinata svih tačaka.

Teorema 4.2 (Teorema o centru mase)**.** *Centar mase skupa tačaka X jedinstvena je tačka koja rešava problem k-sredina za $k = 1$.*

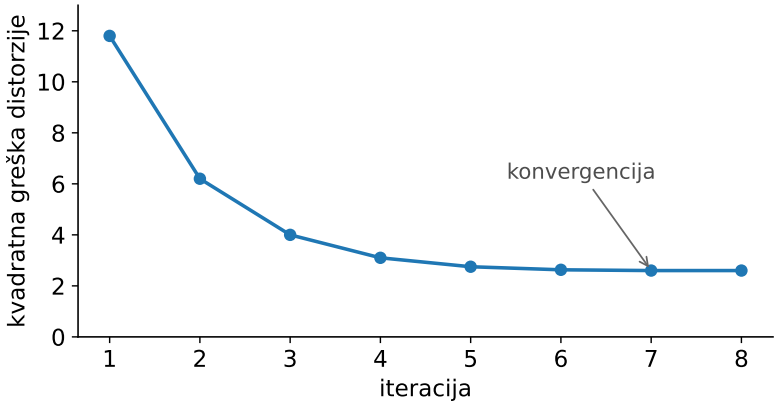
Na ovom zapažanju počiva *Lojdov algoritam* (engl. *Lloyd*), najpoznatija heuristika za problem k-sredina [6]. On najpre izabere k početnih centara, a zatim naizmenično ponavlja dva koraka:

- *od centara do klastera*: svaka tačka se dodeljuje klasteru čiji joj je centar najbliži.
- *od klastera do centara*: novi centar svakog klastera postaje centar mase (prosek) tačaka tog klastera.

Ceo postupak prikazan je u Algoritmu 2.

Algoritam 2 *Lloyd*(X, k)

izaberi k početnih centara C
repeat
    od centara do klastera: dodeli svaku tačku najbližem centru
    od klastera do centara: pomeri svaki centar u centar mase svog klastera
until se centri menjaju
return C



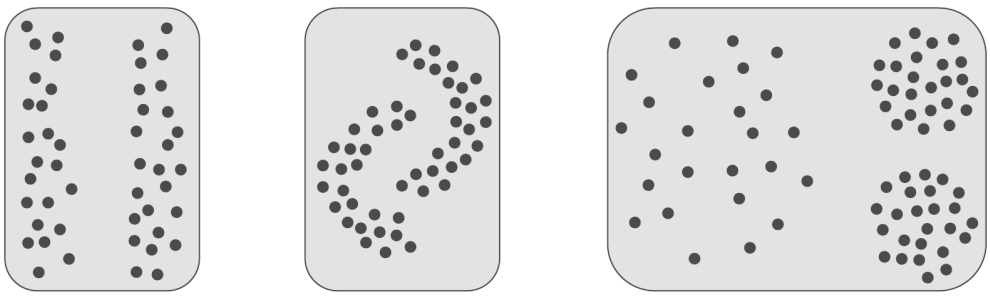
Slika 4.5: Šematski prikaz konvergencije Lojdovog algoritma: kvadratna greška distorzije opada iz iteracije u iteraciju i ubrzo se ustali, kada se centri (a time i klasteri) više ne menjaju.

Postupak se ponavlja dok se centri (a time i klasteri) više ne menjaju, tada kažemo da je algoritam *konvergirao*. Tok konvergencija algoritma je prikazan na slici 4.5. U svakom koraku distorzija se smanjuje ili ostaje ista: u koraku „od centara do klastera" tačka menja klaster samo ako joj je novi centar bliži, a u koraku „od klastera do centara" novi centar po teoremi o centru mase minimizuje distorziju svog klastera. Pošto ima samo konačno mnogo podela tačaka u k klastera, algoritam mora da konvergira, ali obično ka lokalnom, a ne globalnom minimumu. Jedna iteracija ima vremensku složenost $O(nkd)$, jer se za svaku od n tačaka računa rastojanje do k centara u d dimenzija.

Rezultat presudno zavisi od izbora početnih centara: ako neki gust deo podataka ostane bez ijednog centra, algoritam brzo konvergira ka pogrešnoj podeli. Zato se centri ne biraju sasvim slučajno, već tako da budu međusobno razmaknuti. Ovakav postupak, poznat kao *k-sredina++*, bira svaki naredni početni centar nasumično, ali sa verovatnoćom srazmernom kvadratu njegovog rastojanja od već izabranih centa-

ra, čime se izbegava gomilanje početnih centara i u proseku ubrzava konvergencija ka boljem rešenju.

Iako Lojdov algoritam dobro radi kada su klasteri približno globularni i približno jednake veličine, postoje rasporedi tačaka na kojima ne daje ispravnu podelu, iako je ona vizuelno očigledna, kao što je prikazano na slici 4.6. Kod izduženih ili isprepletanih klastera i klastera različite gustine, dodela svake tačke najbližem centru razdvaja grupe koje bismo prirodno videli kao celinu [6]. Osim toga, tvrda dodela[2] je proizvoljna za tačke koje se nalaze približno na sredini između dva centra. Ova ograničenja motivišu naredne pristupe: *meku* dodelu (*Soft k-sredina*), koja ublažava problem graničnih tačaka, i algoritme zasnovane na gustini (*DBSCAN*), koji prepoznaju klastere proizvoljnog oblika.

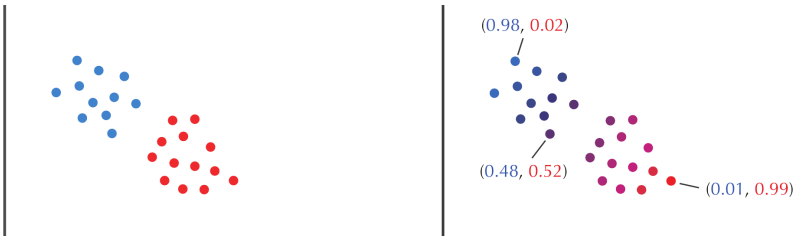


Slika 4.6: Teški rasporedi tačaka za *k-sredina*. Levo i u sredini: dva klastera ($k = 2$) izduženog, odnosno isprepletanog oblika. Desno: tri klastera ($k = 3$) različite gustine. Oko ih lako razdvaja, ali ih Lojdov algoritam, koji svaku tačku pripisuje najbližem centru, često pogrešno podeli. Preuzeto iz [6].

4.1.3 *Soft k-sredina* (*EM*)

Lojdov algoritam svaku tačku pripisuje tačno jednom klasteru (*tvrda* dodela). Za tačke koje se nalaze otprilike na sredini između dva centra takva odluka je proizvoljna i često pogrešna. Zato *soft k-sredina* uvodi *meku* dodelu: umesto da svaku tačku svrsta u jedan klaster, svakoj tački pridružuje niz brojeva - *pripadnosti* (engl. *responsibility*) pojedinih centara za tu tačku, koje se sabiraju na 1. Tačka koja se nalazi u unutrašnjosti jedne grupe ima pripadnost blisku jedinici za taj klaster, dok se kod tačke na sredini pripadnost ravnomerno raspodeljuje među susednim klasterima, kao što je prikazano na slici 4.7.

[2]Kod *tvrde dodele* svaka tačka se svrstava u tačno jedan klaster, za razliku od drugih pristupa gde jedna tačka može istovremeno pripadati većem broju klastera.



Slika 4.7: Tvrda naspram meke dodele klastera tačkama. Levo: Lojdov algoritam svaku tačku boji isključivo plavo ili crveno, prema klasteru kome pripada. Desno: kod meke dodele svaka tačka nosi par brojeva, udeo „plavog" i „crvenog" klastera (njihove pripadnosti), pa se boje mešaju duž spektra. Preuzeto iz [6].

Pripadnosti se računaju *EM* (engl. *expectation maximization*) algoritmom, koji, kao i Lojdov algoritam, polazi od nasumično izabranih centara i naizmenično ponavlja dva koraka [6]:

- *E-korak* (od centara do mekih klastera): za date centre svakoj tački se dodeljuje pripadnost svakom od klastera, srazmerna blizini tačke tom centru.
- *M-korak* (od mekih klastera do centara): svaki centar se pomera u *ponderisani centar mase* (engl. *weighted center of gravity*) svih tačaka, gde su težine upravo pripadnosti.

U E-koraku pripadnost i-tog centra c_i za tačku x_j najčešće se računa pomoću particione funkcije iz statističke fizike

$$HiddenMatrix_{i,j} = \frac{e^{-\beta\, d(x_j, c_i)}}{\sum_l e^{-\beta\, d(x_j, c_l)}},$$

gde je $\beta \geq 0$ *parametar krutosti* (engl. *stiffness*) koji određuje koliko je dodela „meka": za veliko β dodela postaje gotovo tvrda, tj. svaka tačka gotovo u potpunosti pripada jednom klasteru (kao kod Lojdovog algoritma), a za β blisko nuli svi centri dele pripadnost gotovo podjednako [6]. U M-koraku novi centar je

$$c_i = \frac{\sum_j HiddenMatrix_{i,j}\, x_j}{\sum_j HiddenMatrix_{i,j}}.$$

Ceo postupak prikazan je u Algoritmu 3.

Algoritam 3 *SoftKMeans*(X, k, β)

izaberi k početnih centara C
repeat
    E-korak: izračunaj matricu pripadnosti *HiddenMatrix* iz X i C
    M-korak: pomeri svaki centar u ponderisani centar mase (engl. *weighted center of gravity*) tačaka
until se centri menjaju
return matricu pripadnosti *HiddenMatrix* i centre C

Meka dodela ublažava glavni nedostatak Lojdovog algoritma (proizvoljno svrstavanje graničnih tačaka) i daje stabilnije klastere kada grupe nisu jasno razdvojene. Kod podataka o ekspresiji gena to znači da se gen koji se ponaša na prelazu između dva obrasca ne mora obavezno pripisati samo jednom klasteru, već može delimično pripadati i jednom i drugom. Po vremenskoj složenosti jedne iteracije *soft k-sredina* se ne razlikuje od Lojdovog algoritma: i ovde se za svaku od n tačaka računa odnos prema svakom od k centara u d dimenzija, pa jedna iteracija ima vremensku složenost $O(nkd)$. Razlika je samo u tome što se umesto tvrde dodele računaju pripadnosti.

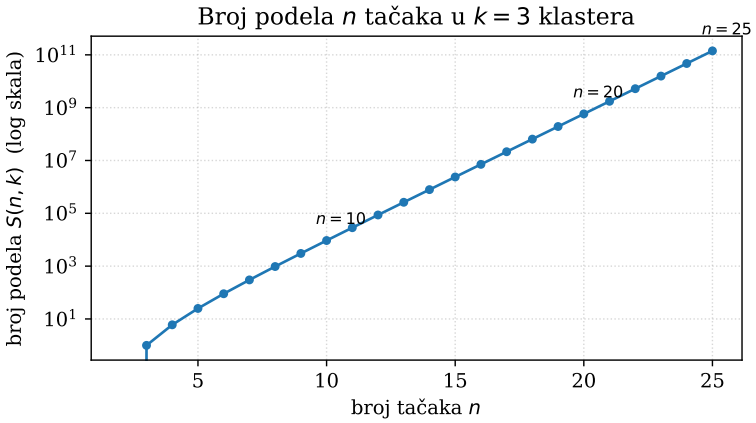
4.1.4 *Brute-force k-sredina*

Pošto je problem *k-sredina* NP-težak za $k > 1$, prirodno je zapitati se zašto se ne reši egzaktnim metodama (iscrpnom pretragom). *Brute-force* (grubi) pristup upravo to radi: razmatra *svaku* moguću podelu n tačaka u k klastera, za svaku izračuna centre mase i odgovarajuću kvadratnu grešku distorzije, i zadrži podelu sa najmanjom greškom. Time se garantovano nalazi globalni optimum, za razliku od heuristika poput Lojdovog algoritma, koje se mogu zaustaviti u lokalnom minimumu. Ceo postupak prikazan je u Algoritmu 4.

Algoritam 4 *BruteForceKMeans*(X, k)

$best \leftarrow +\infty$
for each podelu P skupa X na k nepraznih klastera **do**
    izračunaj centar mase svakog klastera i distorziju Distortion(P)
    if Distortion$(P) <$ *best* **then**
        zapamti P i ažuriraj *best*
    end if
end for
return zapamćenu podelu

Problem je što broj podela n tačaka u k nepraznih klastera, označimo ga $S(n,k)$[3], raste izuzetno brzo, eksponencijalno po n, kao što pokazuje slika 4.8 [6]. Već za $n = 25$ i $k = 3$ postoji preko $1{,}4 \cdot 10^{11}$ podela, a kako se za svaku još računaju centri i distorzija u $O(nkd)$ operacija, ukupna složenost $O\big(S(n,k) \cdot nkd\big)$ čini pretragu izvodljivom samo za sasvim male skupove. Zato se *brute-force k-sredina* koristi jedino kao pojmovni orijentir, tačno rešenje sa kojim se porede vremenski efikasnije heuristike, dok se u praksi na stvarnim podacima primenjuju algoritmi poput Lojdovog.

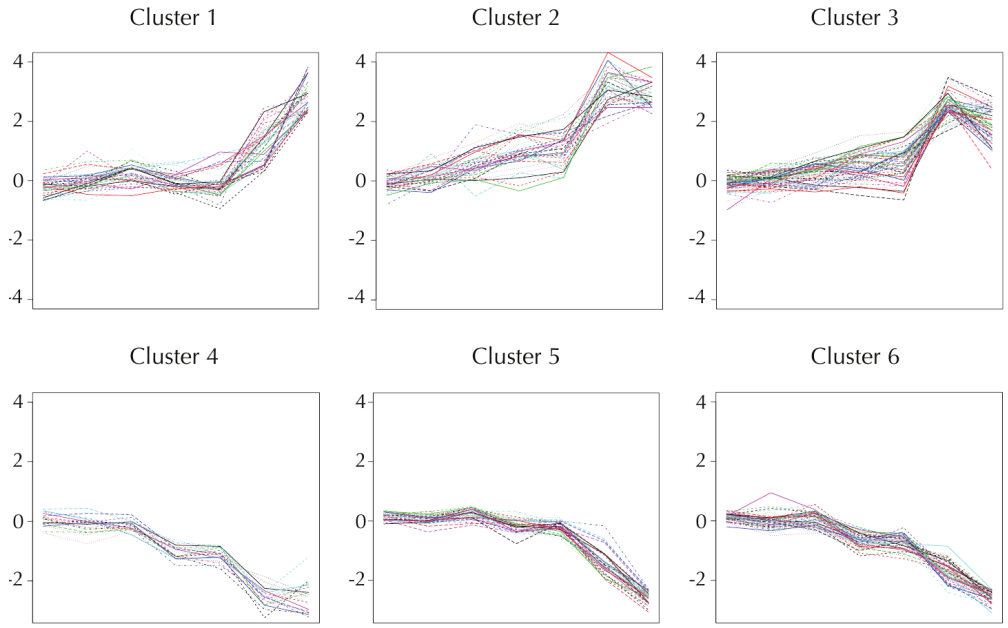


Slika 4.8: Broj različitih podela n tačaka u $k = 3$ klastera ($S(n,k)$, logaritamska skala) raste eksponencijalno, pa iscrpna pretraga brzo postaje neizvodljiva. Činjenica o broju podela preuzeta iz [6].

4.1.5 Primena: geni diauksične promene kod kvasca

Da bismo videli kako se prethodno objašnjeni algoritmi ponašaju na stvarnim podacima, vratimo se skupu iz uvoda, ekspresiji gena kvasca oko diauksične promene. Primenom Lojdovog algoritma na skraćeni skup od 230 gena, uz izbor $k = 6$, dobija se šest klastera koji otkrivaju šest različitih obrazaca ponašanja gena tokom diauksične promene, što je prikazano na slici 4.9 [6].

[3] $S(n,k)$ je *Stirlingov broj druge vrste*: broj načina da se skup od n elemenata podeli na k nepraznih, nerazlikovanih (redosled podskupova nije bitan) podskupova. Računa se zatvorenom formulom $S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^j\binom{k}{j}(k-j)^n$ ili rekurzivno $S(n,k) = k\,S(n-1,k)+S(n-1,k-1)$, uz $S(0,0) = 1$ [9].



Slika 4.9: Rezultat Lojdovog algoritma ($k = 6$) na skupu od 230 gena kvasca. Svaki panel prikazuje vektore ekspresije gena jednog klastera kroz vreme. Šest klastera odgovara šest tipičnih obrazaca regulacije. Preuzeto iz [6].

Neki klasteri okupljaju gene čija ekspresija naglo raste posle diauksične promene (gornji paneli), dok drugi sadrže gene čija ekspresija opada (donji paneli). Ovakva podela ima jasno biološko značenje: geni iz istog klastera verovatno su koregulisani i učestvuju u istim procesima, pa klasterovanje postaje polazna tačka za hipoteze o tome koji regulatorni mehanizmi pokreću prelazak kvasca sa faze fementacija na fazu respiracije [6].

4.2 Algoritmi hijerarhijskog klasterovanja

Algoritmi zasnovani na centrima zahtevaju da se broj klastera k zada unapred. U praksi, međutim, taj broj često nije poznat: kod podataka o ekspresiji gena reč je o hiljadama gena u prostoru mnogo dimenzija, koji se ne može jednostavno nacrtati ni vizuelno prikazati, pa je teško odrediti broj klastera. Povrh toga, unutar globano uočenih grupa (klastera) često se mogu uočiti podgrupe (podklasteri), i tako dalje. Algoritmi hijerarhijskog klasterovanja ne prave jednu podelu, već čitavu *hijerarhiju* ugnežđenih klastera, prikazanu stablom (dendrogramom). Horizontalni rez kroz

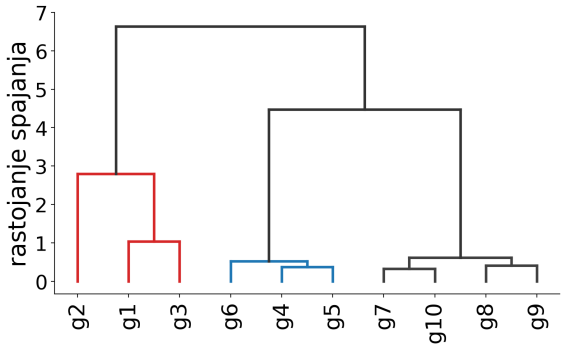
stablo na i mesta deli podatke na i klastera, pa se broj klastera ne mora unapred znati, bira se naknadno, prema tome gde se stablo preseca [6]. Umesto same matrice ekspresije, ovi algoritmi najčešće polaze od $n \times n$ matrice rastojanja D, gde je $D_{i,j}$ rastojanje između vektora ekspresije gena i i j. Postoje dva pristupa za izgradnju hijerarhije (stabla) klastera: *aglomerativni* (odozdo naviše, spajanjem klastera) i *divizivni* (odozgo naniže, deljenjem klastera).

4.2.1 *UPGMA* (aglomerativni pristup)

UPGMA (engl. *Unweighted Pair Group Method with Arithmetic Mean*) je aglomerativni algoritam: kreće od n jednočlanih klastera (listova stabla) i u svakom koraku spaja dva *najbliža* klastera u jedan, dodajući novi čvor stabla iznad njih, sve dok ne ostane jedan klaster koji obuhvata sve tačke [6]. Rastojanje između dva klastera C_1 i C_2 *UPGMA* računa kao prosečno rastojanje svih parova njihovih elemenata,

$$D_{\text{avg}}(C_1, C_2) = \frac{\sum_{i \in C_1} \sum_{j \in C_2} D_{i,j}}{|C_1| \cdot |C_2|}.$$

Drugi izbori (najmanje rastojanje $D_{\min}$, tzv. jednostruko povezivanje, engl. *single linkage*; najveće rastojanje $D_{\max}$, tzv. potpuno povezivanje, engl. *complete linkage*; ili rastojanje između centara, tj. težišta klastera, tzv. centroidno povezivanje, engl. *centroid linkage*) daju drugačija stabla. Rezultat je stablo u kome listovi odgovaraju genima, a unutrašnji čvorovi klasterima nastalim spajanjem, kao što je prikazano na slici 4.10.



Slika 4.10: UPGMA dendrogram za deset gena iz tekućeg primera. Visina spoja odgovara rastojanju na kom su se klasteri spojili. Tri grane, regulisani na gore (crveno), regulisani na dole (plavo) i nepromenjeni (sivo), spajaju se tek na velikoj visini, pa presek stabla na toj visini daje tačno tri grupe. Pomeranjem preseka naniže dobija se više, a naviše manje klastera.

Ceo postupak prikazan je u Algoritmu 5.

Algoritam 5 *UPGMA*(D)

1: $n \leftarrow$ broj vrsta matrice D
2: $Clusters \leftarrow n$ jednočlanih klastera; stablo T sa n izolovanih čvorova
3: **while** ima više od jednog klastera **do**
4: nađi dva najbliža klastera C_i i C_j
5: spoji ih u novi klaster C_{new} i dodaj čvor C_{new} u T povezan sa C_i i C_j
6: ukloni C_i i C_j, pa izračunaj rastojanja $D(C_{new}, C)$ do svih preostalih klastera
7: **end while**
8: **return** stablo T

UPGMA je istorijski jedan od prvih algoritama primenjenih na analizu ekspresije gena i danas se koristi za grupisanje koregulisanih gena. Njegov nedostatak je što su odluke o spajanju *pohlepne* i konačne: jednom spojeni klasteri se kasnije ne mogu razdvojiti, pa rana pogrešna spajanja ostaju u stablu. Što se složenosti tiče, neka je n broj tačaka (gena). Već sama matrica rastojanja zauzima $O(n^2)$ memorije, a algoritam ima $n-1$ koraka spajanja. U svakom koraku traži se najmanji element matrice i ažuriraju rastojanja. Ako se taj minimum u svakom koraku pronalazi prostim pregledom cele matrice (bez pomoćne strukture podataka), jedan korak je složenosti $O(n^2)$, pa je ukupno $(n-1) \cdot O(n^2) = O(n^3)$. Upotrebom pogodnih struktura podataka (npr. prioritetnog reda[4]) vremenska složenost se može spustiti na $O(n^2 \log n)$. Zato je *UPGMA* pogodan za nekoliko stotina do nekoliko hiljada gena, ali ne i za cele genome bez dodatnih ubrzanja.

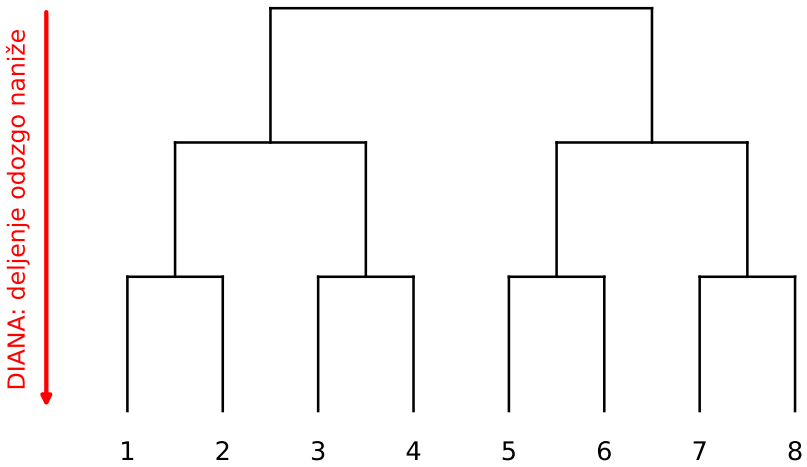
4.2.2 *DIANA* (divizivno)

DIANA (engl. *DIvisive ANAlysis*) ide u suprotnom smeru: polazi od jednog klastera koji sadrži sve tačke i taj klaster postepeno *deli*, sve dok svaka tačka ne postane sopstveni klaster [14]. U svakom koraku bira se klaster sa najvećim *prečnikom* (najvećim rastojanjem između dve njegove tačke), a zatim se deli pomoću principa tzv. *otcepljene grupe* (engl. *splinter group*): za početak se izdvoji tačka sa najvećim prosečnim rastojanjem od ostalih, a onda se u tu grupu redom premeštaju tačke koje su u proseku bliže otcepljenoj grupi nego ostatku klastera. Postupak je prikazan na slici 4.11. Ceo postupak prikazan je u Algoritmu 6.

[4] *Prioritetni red* je struktura podataka koja u svakom trenutku omogućava brz pristup najmanjem (ili najvećem) elementu i njegovo uklanjanje, obično u $O(\log n)$, pa se najmanje rastojanje ne mora tražiti pregledom cele matrice.

Algoritam 6 *DIANA*(D)

$n \leftarrow$ broj vrsta matrice D
$Clusters \leftarrow \{\{1, \ldots, n\}\}$ ▷ jedan klaster sa svih n tačaka
while neki klaster ima više od jedne tačke **do**
    izaberi klaster C sa najvećim prečnikom
    u novu grupu C' izdvoji tačku sa najvećim prosečnim rastojanjem od ostalih u C
    while u $C \setminus C'$ ima tačaka koje su u proseku bliže grupi C' nego ostatku **do**
        premesti ih u C'
    end while
    zameni C klasterima C' i $C \setminus C'$
end while
return stablo podela



Slika 4.11: Divizivno hijerarhijsko klasterovanje (*DIANA*): za razliku od aglomerativnog pristupa koji spaja odozdo naviše, *DIANA* hijerarhiju gradi odozgo naniže: kreće od celog skupa i u svakom koraku razdvaja klaster najvećeg prečnika. Postupak opisan u [14].

Divizivni pristup u prvom koraku sagledava globalnu strukturu podataka (celinu koju deli), dok aglomerativni počinje od lokalnih spajanja. Zbog toga *DIANA* ume da izbegne neka rana pogrešna spajanja *UPGMA*, ali je računski zahtevnija zbog složenosti koraka određivanja optimalne podele klastera [14]. Svaki korak podele klastera ima složenost $O(n^2)$ (računanje rastojanja i prosečnih sličnosti), a kako broj podela klastera može biti do $n - 1$, ukupna složenost dostiže $O(n^3)$, pa je

DIANA u praksi vremenski sporiji algoritam od *UPGMA* i primenjuje se na manje skupove podataka.

4.3 Algoritmi zasnovani na gustini

Algoritmi zasnovani na centrima i hijerarhijski algoritmi dobro rade kada su klasteri približno globularni i približno jednake veličine. Kada klasteri imaju izdužen ili nepravilan oblik, ili kada u podacima ima mnogo šuma, ovi pristupi nisu adekvatan izbor. Algoritmi zasnovani na gustini polaze od drugačije definicije klastera: klaster je *gusto naseljena* oblast prostora, odvojena od drugih klastera oblastima male gustine. Ova klasa algoritama može da otkrije klastere proizvoljnog oblika, ne zahtevaju zadavanje broja klastera i eksplicitno izdvajaju atipične tačke kao šum.

4.3.1 *DBSCAN*

DBSCAN (engl. *Density-Based Spatial Clustering of Applications with Noise*) koristi dva parametra: poluprečnik ε i najmanji broj tačaka $MinPts$ [7]. Za tačku p, njena ε-*okolina* je skup svih tačaka na rastojanju najviše ε od p. Prema gustini svoje okoline, tačke se dele na tri vrste, kao što je prikazano na slici 4.12:

- *tačke u jezgru* (engl. *core*): u ε-okolini imaju bar $MinPts$ tačaka.
- *granične tačke* (engl. *border*): nisu u jezgru, ali se nalaze u okolini neke tačke iz jezgra.
- *šum* (engl. *noise*): sve ostale, tačke u oblastima male gustine.

Slika 4.12: Podela tačaka kod *DBSCAN* algoritma za $MinPts = 4$: tačke u jezgru (crvene) u svojoj ε-okolini imaju bar $MinPts$ tačaka, granične (narandžaste) se nalaze u okolini neke tačke iz jezgra, a tačke šuma (sive) su u oblastima male gustine.

Klaster se definiše kao maksimalan skup međusobno *gusto povezanih* tačaka: dve tačke su gusto povezane ako se do obe može stići lancem tačaka iz jezgra u kome je svaka sledeća u ε-okolini prethodne. Algoritam redom obrađuje još neobrađene tačke: kada naiđe na tačku iz jezgra, formira novi klaster i proširuje ga svim tačkama koje su sa njom gusto povezane. Tačke koje ne pripadaju nijednom klasteru ostaju označene kao šum. Ceo postupak prikazan je u Algoritmu 7.

Algoritam 7 $DBSCAN(X, \varepsilon, MinPts)$

1: **for each** neobrađenu tačku p iz X **do**
2: označi p obrađenom i nađi njenu ε-okolinu
3: **if** p u svojoj ε-okolini ima manje od $MinPts$ tačaka **then**
4: privremeno je označi kao šum
5: **else**
6: formiraj novi klaster i proširi ga svim tačkama koje su gusto povezane sa p
7: **end if**
8: **end for**
9: **return** klastere (tačke označene kao šum ne pripadaju nijednom klasteru)

Glavne prednosti *DBSCAN*-a su što sam otkriva broj klastera, prepoznaje klastere proizvoljnog oblika i otporan je na atipične tačke, koje odvaja kao šum. Nedostatak je osetljivost na izbor parametara ε i $MinPts$ i slabije ponašanje kada klasteri imaju bitno različite gustine. Složenost zavisi od načina traženja ε-okoline: u jednostavnoj izvedbi, gde se za svaku tačku prolazi kroz sve ostale, ona iznosi $O(n^2)$, dok se uz prostorni indeks (npr. *R^*-stablo*[5]) prosečna složenost spušta na $O(n \log n)$, što *DBSCAN* čini primenljivim i na velikim skupovima [7].

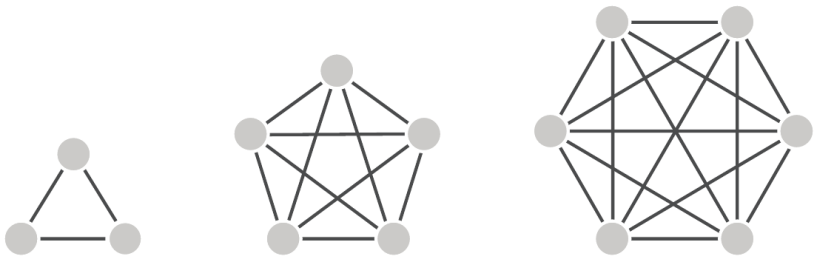
4.4 Grafovski algoritmi

Podaci o ekspresiji gena mogu se predstaviti i *grafom sličnosti*: čvorovi su geni, a dva čvora se povezuju granom ako je sličnost odgovarajućih vektora ekspresije dovoljno velika. Klasterovanje tada postaje *detekcija zajednica*, traženje grupa čvorova koji su gusto povezani unutar sebe, a retko sa ostatkom grafa.

[5] *R^*-stablo* je struktura podataka za prostorno indeksiranje: uravnoteženo stablo koje tačke grupiše u ugnežđene minimalne granične pravougaonike (engl. *minimum bounding rectangles*), pa upit poput „koje tačke se nalaze u ε-okolini date tačke" ne mora da pretražuje ceo skup, već samo relevantne grane, u proseku u $O(\log n)$.

4.4.1 *CAST*

CAST (engl. *Cluster Affinity Search Technique*) polazi od grafa sličnosti $G(R, \theta)$, gde je R matrica sličnosti gena (u glavi 3 označena sa S): $R_{i,j}$ je sličnost vektora ekspresije gena i i j, računata kao što je opisano u Definiciji 3.3 (najčešće Pirsonovim koeficijentom korelacije, $R_{i,j} = r(x_i, x_j)$). Parametar θ je prag povezivanja: dva gena se povezuju granom ako im sličnost $R_{i,j}$ prelazi prag θ [6]. U idealnom slučaju, dobra podela odgovarala bi grafu čije su komponente potpuni podgrafovi (*klike*), takav graf zove se *klik-graf*, kao što je prikazano na slici 4.13. Tada bi svaki klaster bio klika, a između različitih klastera ne bi bilo grana.

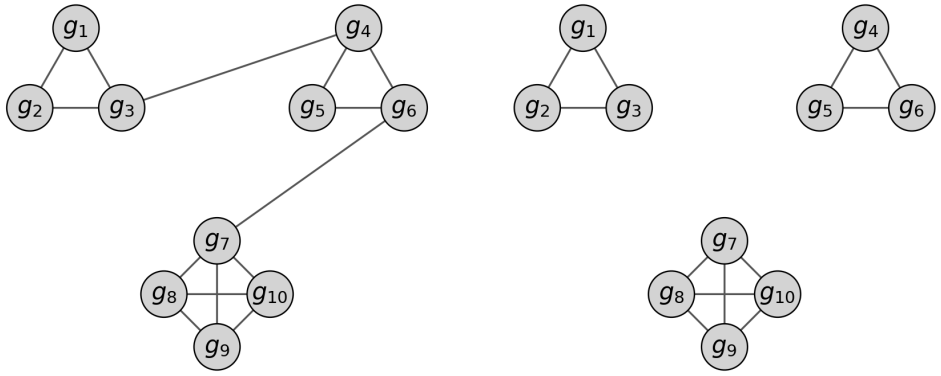


Slika 4.13: Klik-graf sastavljen od tri klike (potpuna podgrafa). U idealnom slučaju, dobra podela gena odgovara upravo ovakvom grafu, svaka klika je jedan klaster, a među klasterima nema grana. Preuzeto iz [6].

U stvarnim podacima graf sličnosti odstupa od idealnog klik-grafa: neke grane nedostaju, a neke su suvišne, kao što je prikazano na slici 4.14.

Definicija 4.4 (Problem iskvarenih klika)**.** Za dati graf, naći najmanji broj grana koje treba dodati i ukloniti da bi se on transformisao u klik-graf.

Ovaj problem je NP-težak, pa se rešava heuristikama. *CAST* je jedna od najuspešnijih metoda na podacima o ekspresiji gena. Afinitet gena i prema klasteru C definiše se kao prosečna sličnost gena i sa svim genima iz C. Gen je θ-*blizak* klasteru ako mu je afinitet veći od θ, inače je θ-*dalek*.



Slika 4.14: Levo: jedan mogući graf sličnosti za deset gena iz tabele 5.2. Desno: isti graf sveden na klik-graf (uklanjanjem suvišnih grana), čije su tri komponente (klike) upravo očekivane grupe: na gore $\{g_1, g_2, g_3\}$, na dole $\{g_4, g_5, g_6\}$ i nepromenjeni $\{g_7, g_8, g_9, g_{10}\}$. *CAST* radi upravo nad ovakvim grafom sličnosti. Prilagođeno prema [6].

Klasteri se grade jedan po jedan: kreće se od čvora najvećeg stepena, pa se naizmenično dodaje najbliži θ-blizak gen koji nije u C i uklanja najdalji θ-dalek gen iz C, dok se klaster ne iskonvergira. Stabilizovani klaster se ukloni iz grafa i postupak se ponavlja nad ostatkom. Ceo postupak prikazan je u Algoritmu 8.

Algoritam 8 $CAST(R, \theta)$

$Graph \leftarrow G(R, \theta)$; $Clusters \leftarrow \emptyset$
while $Graph$ neprazan **do**
    $C \leftarrow$ jednočlani klaster od čvora najvećeg stepena u $Graph$
    while postoji θ-blizak gen van C ili θ-dalek gen u C **do**
        dodaj najbližeg θ-bliskog, ukloni najdaljeg θ-dalekog
    end while
    dodaj C u $Clusters$ i ukloni čvorove iz C iz $Graph$
end while
return $Clusters$

Za razliku od algoritama zasnovanih na centrima, *CAST* ne zahteva unapred zadat broj klastera i pokazao se veoma uspešnim baš na podacima o ekspresiji gena [6]. Konstrukcija grafa sličnosti zahteva poređenje svih parova gena, pa već njegovo računanje ima složenost $O(n^2)$. Kako se pri dodavanju i uklanjanju gena afiniteti mogu inkrementalno ažurirati, ukupna slozenost je reda $O(n^2)$, uz konstantu koja zavisi od broja izmena unutar klastera.

4.4.2 *Louvain* i *Leiden*

Kada je graf sličnosti velik, sa desetinama hiljada čvorova pa i više, potrebni su algoritmi koji zajednice pronalaze brzo i bez unapred zadatog broja klastera. Dva takva, danas među najpopularnijima, jesu *Louvain* i *Leiden*, nazvani po univerzitetima na kojima su nastali, u belgijskom Luvenu i holandskom Lajdenu.[6] Iako su razvijeni za analizu društvenih i drugih složenih mreža, podjednako se prirodno primenjuju u bioinformatici, na grafu sličnosti gena ili u mreži interakcija proteina, gde jedna zajednica odgovara grupi koregulisanih gena ili funkcionalno povezanih proteina. *Leiden* je novija verzija *Louvain*-a: uklanja mu glavnu manu, sklonost da stvori slabo povezane, pa i nepovezane zajednice, a uz to je i vremenski efikasniji [15]. Kao i kod *CAST*-a, graf sličnosti nad kojim oba algoritma rade gradi se iz izabrane mere bliskosti prema Definiciji 3.3, dok se izbor konkretne mere za posmatrani skup gena obrazlaže u odeljku 5.3.

Kvalitet podele grafa na zajednice meri se *modularnošću* Q, razlikom između udela grana koje pripadaju nekoj zajednici i udela koji bi se očekivao u slučajnom grafu sa istim stepenima čvorova [12]. Visoka modularnost znači da su zajednice gušće povezane iznutra nego što bi bile slučajno. *Louvain* algoritam pohlepno maksimizuje modularnost u dve faze koje se ponavljaju, kao što je prikazano na slici 4.15 [5]:

- *lokalno premeštanje*: svaki čvor se premešta u onu susednu zajednicu koja daje najveći porast modularnosti, sve dok su takva premeštanja moguća.
- *agregacija*: svaka zajednica se sažima u jedan super-čvor, čime nastaje manji težinski graf nad kojim se postupak ponavlja.

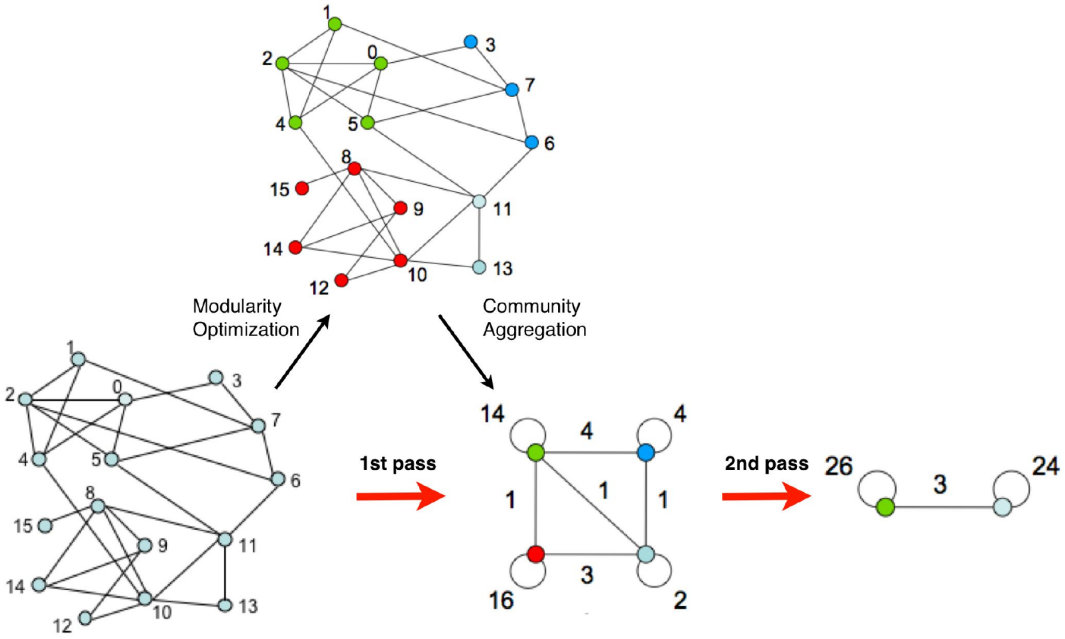
Ponavljanjem ove dve faze zajednice se spajaju u sve veće, pa nastaje hijerarhija zajednica u kojoj svaki naredni nivo objedinjuje one sa prethodnog. Algoritam je računski veoma efikasan i primenjiv na mreže sa milionima čvorova. Ceo postupak prikazan je u Algoritmu 9.

[6] *Louvain* metod uveli su 2008. godine Blondel i saradnici na Univerzitetu u Luvenu, prvobitno za brzo otkrivanje zajednica u ogromnim mrežama koje dotadašnje metode nisu mogle da obrade, poput mreže mobilne telefonije i veb-grafova sa milionima čvorova [5]. *Leiden* su 2019. godine predložili Traag i saradnici na Univerzitetu u Lajdenu, kao ispravku mane po kojoj *Louvain* ume da vrati loše povezane zajednice [15].

Algoritam 9 *Louvain*(G)

1: svaki čvor stavi u sopstvenu zajednicu
2: **repeat**
3: *lokalno premeštanje*:
4: **while** raste modularnost **do**
5: premesti svaki čvor u susednu zajednicu koja daje najveći porast Q
6: **end while**
7: *agregacija*: sažmi svaku zajednicu u super-čvor i formiraj novi težinski graf G
8: **until** se modularnost povećava
9: **return** pronađenu hijerarhiju zajednica

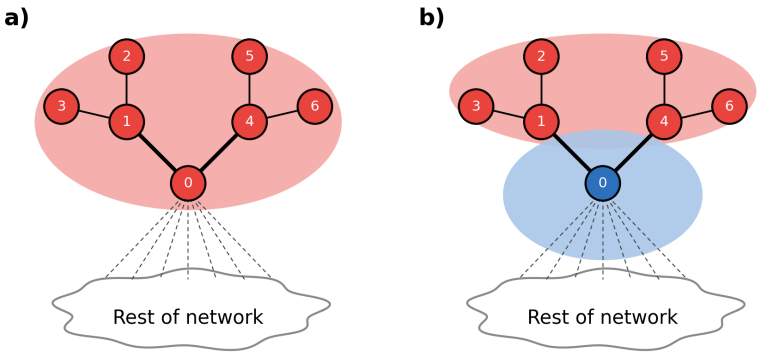
Najzahtevniji deo ovog algoritma je lokalno premeštanje, koje prolazi kroz sve grane grafa, pa je jedan prolaz složenosti $O(m)$, gde je m broj grana. Pošto agregacija smanjuje graf, algoritam se u praksi ponaša skoro linearno po veličini mreže, što je i razlog njegove popularnosti [5].



Slika 4.15: Dve faze *Louvain* algoritma. Prvo se lokalnim premeštanjem čvorova optimizuje modularnost, a zatim se pronađene zajednice agregiraju u super-čvorove novog, manjeg grafa. Postupak se ponavlja. Preuzeto iz [5].

Louvain ima jedan nedostatak: zajednice koje pronađe mogu biti loše povezane, pa čak i *iznutra nepovezane*, premeštanjem jednog čvora zajednica se može raspasti na dva razdvojena dela, kao što je prikazano na slici 4.16 [15]. *Leiden* algoritam

otklanja taj problem uvodeći dodatnu fazu *prečišćavanja* između lokalnog premeštanja i agregacije, koja garantuje da su sve zajednice dobro povezane. Pritom je i računski efikasniji od *Louvain*-a [15].



Slika 4.16: Nedostatak *Louvain* algoritma koji *Leiden* otklanja. (a) Crvena zajednica je povezana samo preko čvora 0. (b) Kada se čvor 0 premesti u drugu zajednicu, preostali deo crvene zajednice postaje iznutra nepovezan, iako i dalje čini jednu zajednicu. Prilagođeno prema [15].

Ceo postupak prikazan je u Algoritmu 10.

Algoritam 10 *Leiden*(G)

1: svaki čvor stavi u sopstvenu zajednicu
2: **repeat**
3: *lokalno premeštanje* čvorova radi porasta modularnosti
4: *prečišćavanje*: unutar svake zajednice dozvoli razdvajanje na dobro povezane podzajednice
5: *agregacija* prečišćenih zajednica u novi graf G
6: **until** se modularnost povećava
7: **return** pronađenu hijerarhiju zajednica

Dodatna faza prečišćavanja garantuje da nijedna zajednica nije iznutra nepovezana, a *Leiden* je pritom u praksi i računski efikasniji od *Louvain*-a, uz istu približno linearnu složenost po broju grana [15].

Glava 5

Evaluacija i poređenje algoritama

Pošto različiti algoritmi isti skup gena mogu podeliti na različite načine, prirodno je zapitati se koliko je neka podela dobra i kako izabrati broj klastera. U ovoj glavi najpre se opisuju mere kvaliteta klasterovanja i način izbora broja klastera, zatim se svi obrađeni algoritmi porede po osnovnim svojstvima, i na kraju se eksperimentalno porede primenom na istom skupu podataka o ekspresiji gena.

5.1 Evaluacija klasterovanja i izbor broja klastera

Kvalitet podele može se meriti *unutrašnjim* merama, koje koriste samo geometriju podataka. U potpoglavlju 4.1 već je uvedena kvadratna greška distorzije, koja meri koliko su tačke u proseku blizu svojim centrima. Druga česta mera je *koeficijent siluete* (engl. *silhouette coefficient*) [14]. Za tačku x_i neka je $a(x_i)$ njeno prosečno rastojanje do ostalih tačaka istog klastera, a $b(x_i)$ najmanje prosečno rastojanje do tačaka nekog drugog klastera. Silueta tačke je

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max\{a(x_i),\, b(x_i)\}} \in [-1, 1],$$

a silueta cele podele je prosek $s(x_i)$ po svim tačkama. Vrednosti $s(x_i)$ blizu 1 znači da je tačka znatno bliža svom nego susednom klasteru (dobro razdvajanje), a negativna vrednost da je verovatno pogrešno svrstana [14].

Kod algoritama koji zahtevaju broj klastera k unapred, samo k se najčešće bira jednom od dve metode. *Metoda lakta* (engl. *elbow method*) posmatra distorziju kao funkciju broja klastera k: ta funkcija monotono opada sa porastom k (svaki dodatni klaster smanjuje distorziju), ali se brzina njenog opadanja u jednom trenutku naglo smanjuje, pa dalje povećanje k donosi sve manje smanjenje distorzije. Tačka u

kojoj nastupa ta nagla promena brzine opadanja, tzv. *lakat* („koleno") krive, uzima se kao dobar izbor broja klastera k. Alternativno, bira se k koje daje najveću prosečnu siluetu. U analizi ekspresije gena konačan izbor često se dodatno usklađuje sa biološkom interpetacijom, tako da dobijeni klasteri odgovaraju prepoznatljivim grupama gena [6]. Kada je tačna podela unapred poznata, kvalitet se može proceniti i *spoljašnjom* merom, poređenjem dobijenih klastera sa poznatim grupama, što je pristup korišćen u eksperimentu koji sledi.

5.2 Uporedni pregled algoritama

Tabela 5.1 sažima osnovna svojstva svih obrađenih algoritama: kojoj klasi pripadaju, da li zahtevaju broj klastera unapred i da li ga sami otkrivaju, kakav oblik klastera prepoznaju, koliku imaju složenost i kako se ponašaju u prisustvu atipičnih tačaka.

		Broj klastera			
Algoritam	**Klasa**	k	**Oblik klastera**	**Atipične tačke**	**Složenost**
FarthestFirstTraversal	centri	zadaje se	globularni	vrlo osetljiv	$O(nkd)$
k-sredina (Lloyd)	centri	zadaje se	globularni	osetljiv	$O(nkd)$/it.
Soft k-sredina (EM)	centri	zadaje se	globularni (meko)	osetljiv	$O(nkd)$/it.
Brute-force k-sredina	centri	zadaje se	globularni	osetljiv	$O(S(n,k)\,nkd)$
UPGMA	hijerarhijski	otkriva se*	zavisi od povezivanja	osetljiv	$O(n^3)$
DIANA	hijerarhijski	otkriva se*	zavisi od povezivanja	osetljiv	$O(n^3)$
DBSCAN	gustina	otkriva se	proizvoljan	robustan (šum)	$O(n \log n)$
CAST	grafovski	otkriva se	guste grupe (klike)	umeren	$O(n^2)$
Louvain	grafovski	otkriva se	zajednice	umeren	$\approx O(m)$
Leiden	grafovski	otkriva se	zajednice	umeren	$\approx O(m)$

Tabela 5.1: Uporedni pregled algoritama klasterovanja. Oznaka n je broj tačaka, k broj klastera, d dimenzija, m broj grana grafa, a $S(n,k)$ Stirlingov broj druge vrste (broj podela n elemenata u k nepraznih grupa). Oznaka „/it." znači *po jednoj iteraciji*. * Hijerarhijski algoritmi ne traže k unapred, broj klastera bira se naknadno, presekom stabla na željenoj visini.

5.3 Eksperimentalno poređenje na podacima o kvascu

Da bi se algoritmi uporedili u praksi, svi implementirani algoritmi u okviru ovog rada primenjeni su na isti mali demonstrativni skup podataka: deset gena merenih u sedam vremenskih tačaka, sa unapred poznatom podelom na tri grupe, geni regulisani na gore (pojačana ekspresija posle diauksične promene), geni regulisani na dole

(smanjena ekspresija) i geni nepromenjene ekspresije. Skup je prikazan u tabeli 5.2. U nastavku se prati kroz svaki algoritam redom, uz odgovarajuću transformaciju ulaznih podataka koju svaki zahteva, a pregled rezultata daje tabela 5.4.

Gen	**Gen kvasca**	**Grupa**	t_1	t_2	t_3	t_4	t_5	t_6	t_7
g_1	YGR043C	na gore	−0,43	−0,73	−0,06	−0,11	−0,16	3,47	2,64
g_2	YLR258W	na gore	0,11	0,43	0,45	1,89	2,00	3,32	2,56
g_3	YKL026C	na gore	−0,19	−0,15	0,03	0,27	0,54	3,64	2,74
g_4	YMR290C	na dole	0,12	−0,23	−0,24	−1,16	−1,40	−2,67	−3,00
g_5	YPL012W	na dole	0,09	−0,28	−0,15	−1,18	−1,59	−2,96	−3,08
g_6	YNL141W	na dole	−0,16	−0,04	−0,07	−1,26	−1,20	−2,82	−3,13
g_7	YLR361C	nepromenjen	0,14	0,03	−0,06	0,07	−0,01	−0,06	−0,01
g_8	YNR065C	nepromenjen	−0,10	−0,14	−0,03	−0,06	−0,07	−0,14	−0,04
g_9	YJL028W	nepromenjen	−0,28	−0,23	−0,19	−0,19	−0,32	−0,18	−0,18
g_{10}	YPR055W	nepromenjen	0,15	0,15	0,17	0,09	0,07	0,09	0,07

Tabela 5.2: Stvarni skup od deset gena kvasca: 10×7 podmatrica DeRisijeve matrice ekspresije, sa vrednostima $\log_2$ *fold change* u sedam vremenskih tačaka oko diauksične promene; preuzeto iz [6]. Kolona *Gen kvasca* daje oznaku (*ORF*) svakog gena, a kolona *Grupa* očekivanu grupu (regulisani na gore, regulisani na dole ili nepromenjeni). Radi preglednosti, u nastavku se geni označavaju sa $g_1, \ldots, g_{10}$.

Različiti algoritmi zahtevaju različit oblik ulaznih podataka. Algoritmi zasnovani na centrima rade direktno nad sirovim vektorima ekspresije u $\mathbb{R}^7$. Hijerarhijski algoritmi ulaz najpre transformišu u 10×10 matricu rastojanja D, a grafovski algoritmi (*CAST*, *Louvain*, *Leiden*) grade graf sličnosti $G(S, \theta)$ iz matrice sličnosti S, koja se može dobiti iz Pirsonove korelacije ili iz rastojanja (Definicija 3.3). U primerima (glave 5 i 6) koristićemo sličnost izvedenu iz euklidskog rastojanja ($S_{i,j} = 1 - d_{i,j}/d_{\max}$). Očekivana podela ima tri grupe: regulisane na gore $\{g_1, g_2, g_3\}$, regulisane na dole $\{g_4, g_5, g_6\}$ i nepromenjene $\{g_7, g_8, g_9, g_{10}\}$.

Sve tri reprezentacije podataka prikazane su na slici 5.1 i ističu suštinsku razliku između dve mere bliskosti. U matrici rastojanja D sve tri grupe obrazuju jasne blokove malih međusobnih udaljenosti: regulisani na gore i na dole zato što dele izražen obrazac promene, a nepromenjeni zato što se njihove vrednosti nalaze u okolini nule, te su i međusobno bliski. U matrici Pirsonove korelacije R blokove visoke sličnosti ($R \approx 1$) obrazuju, međutim, samo regulisani na gore i na dole, dok nepromenjeni geni, čija je ekspresija niska i pretežno šumovita, međusobno slabo koreliraju i ne formiraju blok.

Ova razlika je načelne prirode: euklidsko rastojanje izražava bliskost po *nivou* ekspresije, dok Pirsonova korelacija izražava podudarnost *obrasca* promene, odnosno

koregulisanost. Nepromenjeni geni bliski su po nivou, budući da im vrednosti leže u okolini nule, ali nisu koregulisani, jer nemaju zajednički obrazac promene. Izbor mere bliskosti stoga određuje da li će nepromenjeni geni uopšte biti prepoznati kao zasebna grupa. Kako u ovom skupu nepromenjene gene posmatramo kao jednu od tri grupe, graf sličnosti za grafovske algoritme (*CAST*, *Louvain*, *Leiden*) konstruišemo koristeći euklidsko rastojanje kao osnovu sličnosti ($S_{i,j} = 1 - d_{i,j}/d_{\max}$, Definicija 3.3); tada i ti algoritmi prepoznaju sve tri grupe. Da smo graf konstruisali iz korelacije, nepromenjeni geni bi se, kao nekoregulisani, priključili zajednicama regulisanih gena umesto da obrazuju sopstvenu; korelaciona sličnost bila bi primerenija ukoliko bi cilj bio isključivo izdvajanje koregulisanih grupa.

U nastavku se prikazuje ponašanje svakog algoritma na ovom skupu podataka, redom po klasama: za svaki se navodi format ulaznih podataka koji zahteva i grupisanje gena koju vraća na ovom primeru.

FarthestFirstTraversal. Nad sirovim vektorima ekspresije gena, za $k = 3$, vraća tačno tri očekivane grupe.

k-sredina (Lloyd). Nad sirovim vektorima ekspresije gena, za $k = 3$, brzo konvergira (distorzija $\approx 0{,}64$) i u svih pet nasumičnih inicijalizacija vraća iste tri grupe gena. Dobijeni centri odgovaraju prosečnim obrascima grupa, rastućem, opadajućem i neizmenjenom.

Soft k-sredina (EM). Isti sirovi format ulaznih podataka, $k = 3$, $\beta = 2$. Matrica pripadnosti je izrazito izoštrena (svaki gen ima pripadnost blisku jedinici za svoju grupu), pa se meka dodela poklapa sa tvrdom dodelom. Sa manjim β dodela postaje mekša.

Brute-force k-sredina. Iscrpna pretraga potvrđuje istu podelu na tri grupe.

***UPGMA* i jednostruko povezivanje.** Ulaz se najpre transformiše u matricu rastojanja D. Prva spajanja su unutar grupa (na malim rastojanjima), a tek poslednja dva spajanja povezuju različite grupe, uz nagli skok rastojanja (na $\approx 4{,}5$ i $\approx 6{,}6$). Rez na tri klastera daje tačno ciljno grupisanje gena (dendrogram dobijene hijerarhije klastera prikazan je na slici 4.10). Jednostrukim povezivanjem ($D_{\min}$) dobija se ista podela gena.

DIANA. Kao kod prethodnog algoritma, polazi se od matrice rastojanja D, ali se hijerarhija klastera gradi odozgo naniže, uzastopnim podelama klastera najvećeg prečnika: prva podela izdvaja gene regulisane na dole, a naredni korak razdvaja nepromenjene gene od regulisanih na gore. Rez na tri klastera daje isto ciljno grupisanje gena kao *UPGMA*.

DBSCAN. Radi nad sirovim vektorskim reprezentacija ekspresije gena i sam otkriva broj klastera. Za $\varepsilon \in [2{,}5;\ 3{,}0]$, $MinPts \in \{2, 3\}$ vraća tri klastera bez ijedne tačke šuma - opet tri očekivane grupe (pri manjim ε deo gena ostaje označen kao šum).

CAST. Ulaz se pretvara u graf sličnosti izveden iz rastojanja ($S = 1 - d/d_{\max}$) uz prag $\theta = 0{,}7$. *CAST* automatski nalazi tri klastera koji odgovaraju očekivanim grupama gena.

Louvain* i *Leiden. Nad istim grafom sličnosti (prag $\theta = 0{,}6$) oba algoritma maksimizuju modularnost i daju $Q \approx 0{,}61$, uz podelu na iste tri očekivane grupe gena.

Vrsta ulaza koju svaki algoritam zahteva i rezultat koji vraća na ovom primeru sažeti su u tabeli 5.3.

Algoritam	**Ulaz**	**Vraća (na ovom primeru)**
FarthestFirstTraversal	sirovi vektori u $\mathbb{R}^7$, $k = 3$	tri centra i podelu, tri očekivane grupe
k-sredina (*Lloyd*)	sirovi vektori, $k = 3$	tri centra i podelu (tri grupe), distorzija $\approx 0{,}64$
Soft k-sredina (*EM*)	sirovi vektori, $k = 3$, β	matricu pripadnosti 3×10, tri grupe
Brute-force k-sredina	sirovi vektori, $k = 3$	podelu sa najmanjom distorzijom, tri grupe
UPGMA	matrica rastojanja D (10×10)	dendrogram; rez na $k = 3$ daje tri grupe
DIANA	matrica rastojanja D	dendrogram (odozgo), tri grupe
DBSCAN	sirovi vektori, ε, $MinPts$	tri klastera, nula šuma, tri grupe. Broj bira sam
CAST	graf sličnosti $(S = 1 - d/d_{\max})$, $\theta = 0{,}7$	tri klastera, tri grupe
Louvain	graf sličnosti	zajednice, $Q \approx 0{,}61$, tri grupe
Leiden	graf sličnosti	zajednice, $Q \approx 0{,}61$, tri grupe

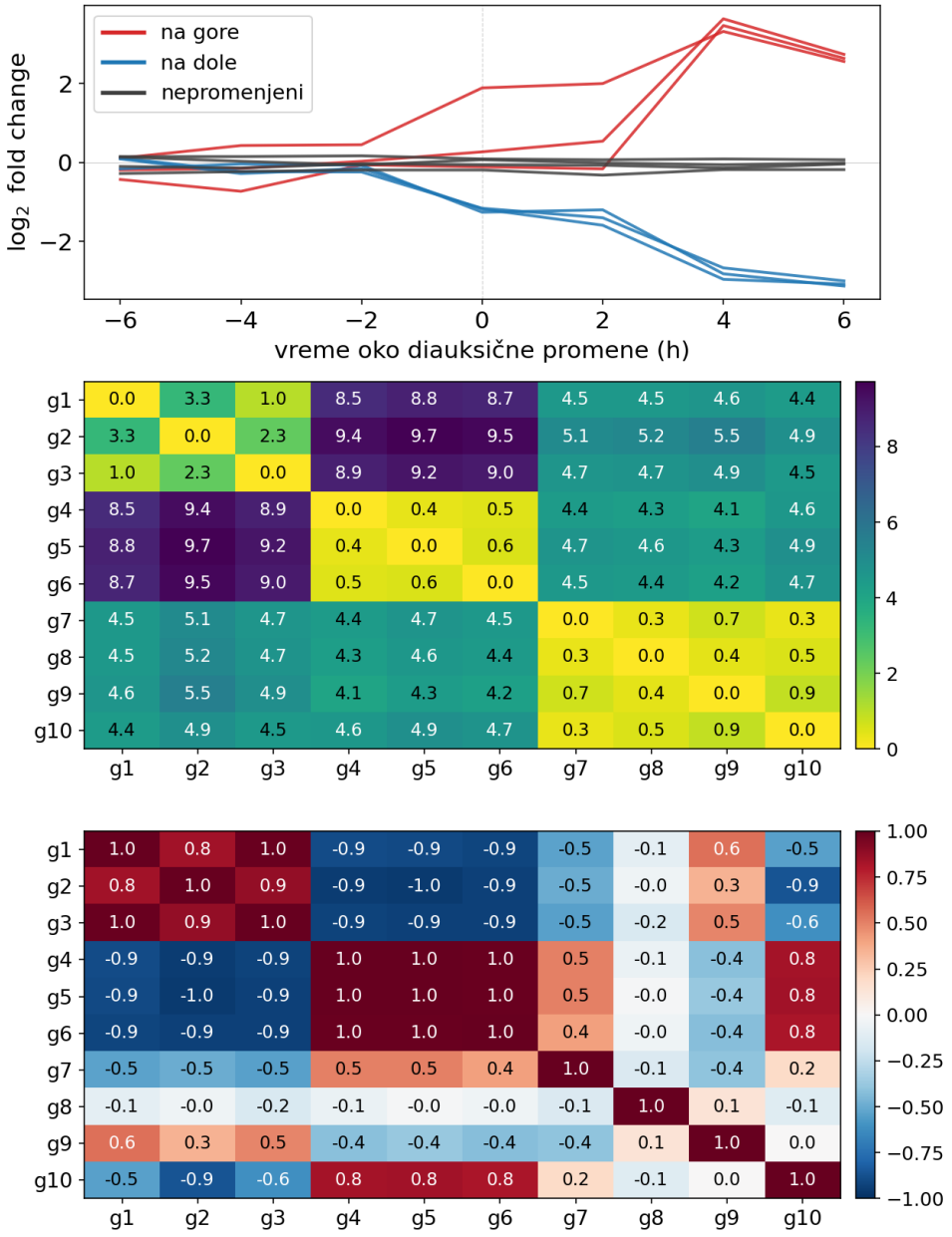
Tabela 5.3: Ulaz i izlaz svakog algoritma na demonstrativnom skupu od deset gena. Algoritmi zasnovani na centrima rade nad sirovim vektorima, hijerarhijski nad matricom rastojanja D, a grafovski nad grafom sličnosti (sličnost $S = 1 - d/d_{\max}$ izvedena iz rastojanja).

Algoritam	**Klasa**	**Vraća očekivane grupe?**
FarthestFirstTraversal	centri	da
k-sredina (*Lloyd*)	centri	da (za svih 5 inicijalizacija)
Soft k-sredina (*EM*)	centri	da
Brute-force k-sredina	centri	da
UPGMA	hijerarhijski	da
DIANA	hijerarhijski	da
DBSCAN	gustina	da
CAST	grafovski	da
Louvain	grafovski	da
Leiden	grafovski	da

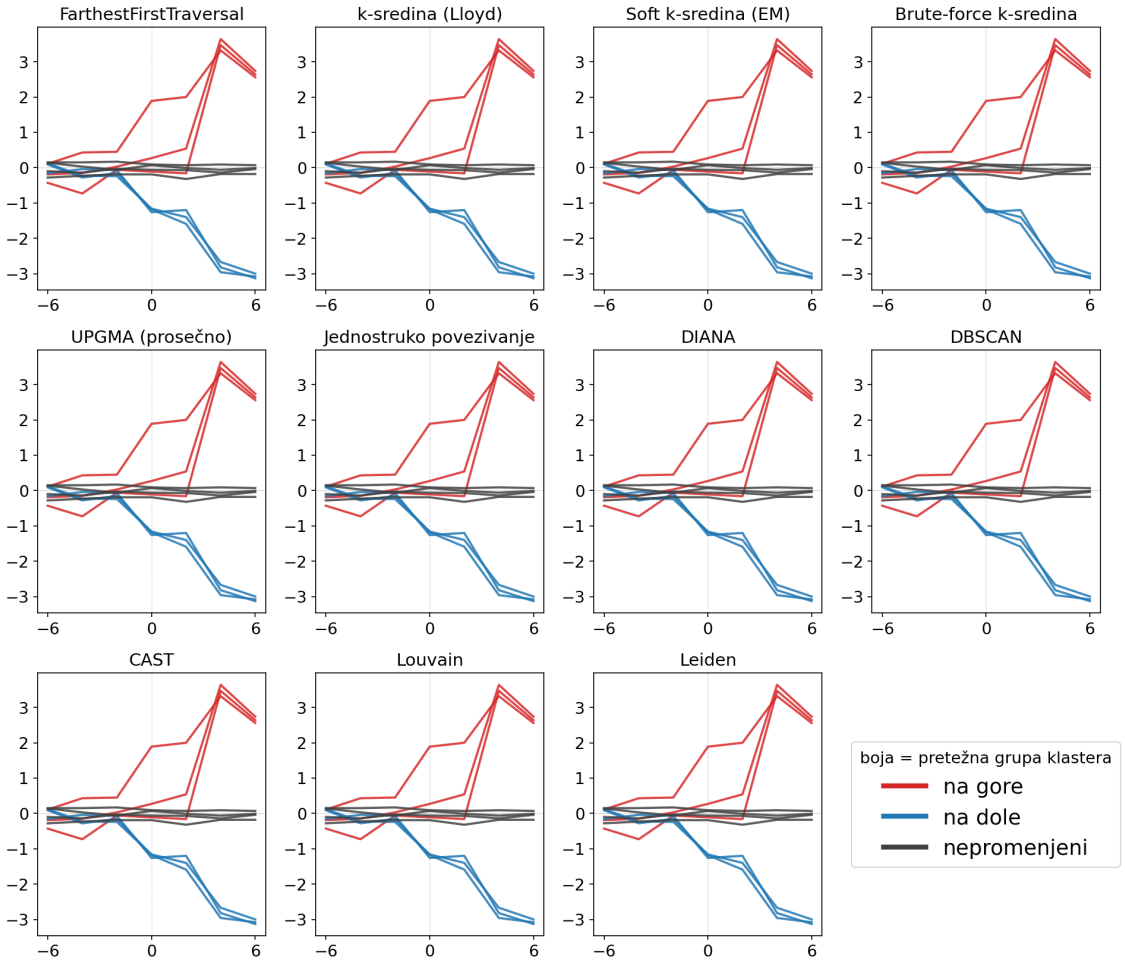
Tabela 5.4: Rezultat primene algoritama na skup od deset gena sa poznatom podelom na tri grupe. Prikazano je svih deset algoritama; *UPGMA* daje istu podelu za obe varijante povezivanja (prosečno i jednostruko), pa je naveden u jednom redu. Svi algoritmi tačno rekonstruišu sve tri grupe (grafovski algoritmi uz sličnost izvedenu iz rastojanja).

Na posmatranom skupu podataka svih deset razmatranih algoritama uspešno je rekonstruisalo tri očekivane grupe, pri čemu *UPGMA* daje istu podelu za obe varijante kriterijuma povezivanja. Rezultati grupisanja dobijeni primenom pojedinačnih algoritama prikazani su na slici 5.2. Lojdov algoritam je u svih pet nasumičnih inicijalizacija pronašao identičnu i tačnu podelu skupa gena. Ovakva stabilnost rezultata

ukazuje na to da, u slučaju jasno razdvojenih grupa sa izraženim obrascima ekspresije, izbor početnih centara nema značajan uticaj na konačno rešenje. Grafovski algoritmi (*CAST*, *Louvain* i *Leiden*) takođe su uspešno identifikovali tri očekivane grupe kada je graf sličnosti konstruisan na osnovu rastojanja. Međutim, izbor mere sličnosti pokazao se kao značajan faktor koji direktno utiče na strukturu dobijenog grafa, a samim tim i na rezultat detekcije zajednica. U slučaju konstrukcije grafa na osnovu Pirsonove korelacije, nepromenjeni geni ne bi ostvarili dovoljan nivo povezanosti sa ostalim genima da bi formirali zasebnu zajednicu, već bi bili raspoređeni među grupe regulisanih gena. Ovaj rezultat dodatno ukazuje na značaj odgovarajućeg izbora mere bliskosti pri primeni grafovskih metoda. Na osnovu dobijenih rezultata može se zaključiti da, na manjim skupovima gena sa jasno izraženim obrascima ekspresije, sve razmatrane klase algoritama mogu pouzdano rekonstruisati očekivanu strukturu grupa, pod uslovom da su mera bliskosti i relevantni parametri algoritma adekvatno odabrani. Istovremeno, rezultati ukazuju na to da se prednosti grafovskih metoda za detekciju zajednica mogu očekivati pre svega pri analizi većih i složenijih skupova podataka.



Slika 5.1: Isti skup podataka (deset gena iz tabele 5.2) u tri reprezentacije koje traže različite klase algoritama. (a) Sirovi vektori ekspresije (Definicija 2.1), ulaz za algoritme zasnovane na centrima i *DBSCAN*. (b) Matrica euklidskih rastojanja D (Definicije 3.1 i 3.3), ulaz za hijerarhijske algoritme. Tamni blokovi na dijagonali pokazuju da su geni iste grupe međusobno bliski, dok su rastojanja između grupa velika. (c) Matrica Pirsonove korelacije R (Definicije 3.2 i 3.3), koja meri koregulisanost. Kod regulisanih na gore i na dole tamnocrveni blokovi označavaju visoku korelaciju ($R \approx 1$) unutar grupe, dok nepromenjeni geni međusobno slabo koreliraju i ne formiraju blok; zato grafovski algoritmi u eksperimentu koriste sličnost izvedenu iz rastojanja (matrica D), koja nepromenjene gene grupiše. Matrice D i R izračunate su iz sirovih vektora sa panela (a).



Slika 5.2: Podela koju vraća *svaki pojedinačni* algoritam na skupu od deset gena. U svakom panelu krive ekspresije obojene su prema klasteru kome je gen dodeljen. Svi prikazani postupci (zasnovani na centrima, hijerarhijski sa obe varijante povezivanja, *DBSCAN* i grafovski) vraćaju identičnu podelu veličina 3-3-4; grafovski algoritmi (*CAST*, *Louvain*, *Leiden*) pri tome koriste sličnost izvedenu iz rastojanja. Boja krive označava grupu kojoj klaster pripada (na gore crveno, na dole plavo, nepromenjeni tamnosivo).

Glava 6

Implementacija i uputstvo za korišćenje elektronske lekcije

Elektronska lekcija realizovana je kao interaktivna *Jupyter sveska* [1] i sastoji se od teorijskog i interaktivnog dela. Teorijski deo sadrži objašnjenja i matematičke formulacije algoritama, dok interaktivni deo omogućava izvršavanje algoritama korak po korak: pritiskom na *Shift+Enter* (ili klikom na dugme *Run* u *Toolbar* liniji) u odgovarajućoj ćeliji prikazuje se sledeći korak izvršavanja algoritma na konkretnom skupu podataka. Sav kod algoritama je pisan u programskom jeziku *Python* [3] uz upotrebu samo standardne biblioteke, dok je za vizuelizaciju rezultata korišćena biblioteka *matplotlib* [2].

Svaki korak istovremeno se prikazuje na dva komplementarna načina. Grafički prikaz sadrži raspored tačaka obojenih prema pripadnosti klaster u tekućoj podeli, položaje centara (oznaka $\times$) i, gde je to primereno algoritmu, dodatne elemente poput grana grafa sličnosti (*Louvain*) ili istaknute ε-okoline razmatrane tačke (*DBSCAN*). Uz svaku ćeliju se ispisuje i tekstualni opis koraka algoritma: redni broj koraka i ukupan broj koraka, traka napretka i rečenica koja sažima promene nastale u tom koraku, na primer koliko je tačaka promenilo klaster, koliki je pomak centara ili koja je tačka pripojena klasteru i kolika je pri tome njena sličnost sa tim klasterom. Kod algoritma *soft k-sredina* dodatno se prikazuje i matrica pripadnosti (*HiddenMatrix*) sa verovatnoćama pripadnosti pojedinačnih gena klasterima. Na taj način interaktivni deo pruža ne samo vizuelni uvid u rad algoritma, već i njegove kvantitativne međurezultate.

Parametri algoritama. Za svaki algoritam korisnik može da menja parametre i da neposredno posmatra njihov uticaj na dobijeno klasterovanje. Skup parametara dostupnih u interaktivnom delu dat je u tabeli 6.1. Promenom vrednosti (na primer radijusa ε i praga *MinPts* kod algoritma *DBSCAN*, ili parametra krutosti β kod algoritma *soft k-sredina*) i ponovnim pokretanjem ćelije klasterovanje se iznova izvršava za nove vrednosti parametara, čime se ilustruje osetljivost svakog algoritma na izbor parametara. Kod algoritama zasnovanih na centrima, početna vrednost generatora slučajnih brojeva (engl. *random seed*) određuje slučajan izbor početnih centara: ista vrednost daje isti izbor, pa i istu, ponovljivu podelu, dok drugačija početna vrednost može dati drugačiju inicijalizaciju i konačnu podelu.

Algoritam	**Podesivi parametri algoritma**
FarthestFirstTraversal	broj klastera k; izbor prvog centra
k-sredina (*Lloyd*)	broj klastera k; najveći broj iteracija; tolerancija[1]; slučajan izbor početnih centara (engl. *random seed*)
Soft k-sredina (*EM*)	broj klastera k; parametar krutosti β; najveći broj iteracija; slučajan izbor početnih centara
UPGMA	ciljni broj klastera; pravilo povezivanja (prosečno, jednostruko, potpuno, centroidno)
DIANA	ciljni broj klastera
DBSCAN	radijus okoline ε; najmanji broj suseda *MinPts*
CAST	prag sličnosti θ
Louvain / *Leiden*	prag ε za građenje grafa sličnosti
Brute-force k-sredina	broj klastera k

Tabela 6.1: Parametri koje korisnik može da zadaje u interaktivnom delu lekcije, po algoritmu.

Projekcija gena u dve dimenzije. Da bi se koraci mogli pratiti u ravni, svaki gen, izvorno vektor ekspresije u sedam vremenskih tačaka, preslikava se u jednu tačku dvodimenzionog prostora određenu dvema izvedenim veličinama: na x-osi je prosečna ekspresija gena nakon diauksične promene, a na y-osi raspon (amplituda) njegove ekspresije, definisan kao razlika između najveće i najmanje izmerene vrednosti. Ova projekcija je odabrana zato što upravo te dve veličine, prosečni nivo i amplituda promene, razdvajaju tri očekivane grupe gena (regulisane na gore, regulisane na dole i nepromenjene), pa se struktura klastera jasno uočava i u ravni. Sami algoritmi pri tome rade nad punim sedmodimenzionim vektorima (odnosno

[1]Tolerancija je prag konvergencije: najmanji pomak centara između dve iteracije ispod kojeg se algoritam zaustavlja (centri se praktično više ne pomeraju).

nad izvedenim matricama rastojanja ili sličnosti); dvodimenziona projekcija služi isključivo za vizualizaciju rezultata.

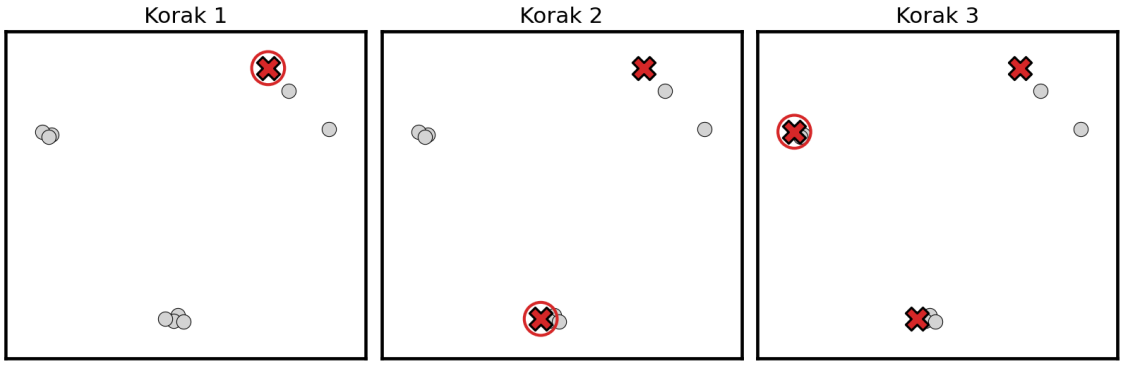
6.1 Primer korišćenja elektronske lekcije

Da bi se videlo *kako* svaki algoritam dolazi do podele, isti geni su projektovani u dve dimenzije (na x-osu prosečna ekspresija, na y-osu raspon) i svaki algoritam je prikazan korak po korak. Boje označavaju tekuće klastere, a oznaka $\times$ centar klastera (gde ga algoritam koristi). Numerisani koraci ispod svakog algoritma odgovaraju panelima (Korak 1, 2, ...) na slici.

Korak po korak prikazano je devet algoritama: *FarthestFirstTraversal*, *k-sredina* (*Lloyd*), *soft k-sredina* (*EM*), *brute-force k-sredina*, *UPGMA*, *DIANA*, *DBSCAN*, *CAST* i *Louvain*. Algoritam *Leiden* nije prikazan zasebno jer u fazi lokalnog premeštanja čvorova ponavlja korake algoritma *Louvain*, uz dodatnu proveru povezanosti zajednica koja na ovako malom grafu ne menja rezultat; iz istog razloga kod algoritma *UPGMA* varijanta sa jednostrukim povezivanjem nije prikazana zasebno: na ovom skupu daje isti redosled spajanja klastera kao *UPGMA* sa prosečnim povezivanjem.

FarthestFirstTraversal. Algoritam bira centre tako da je svaki novi najudaljeniji od prethodno izabranih. Izbor centara korak po korak prikazan je na slici 6.1:

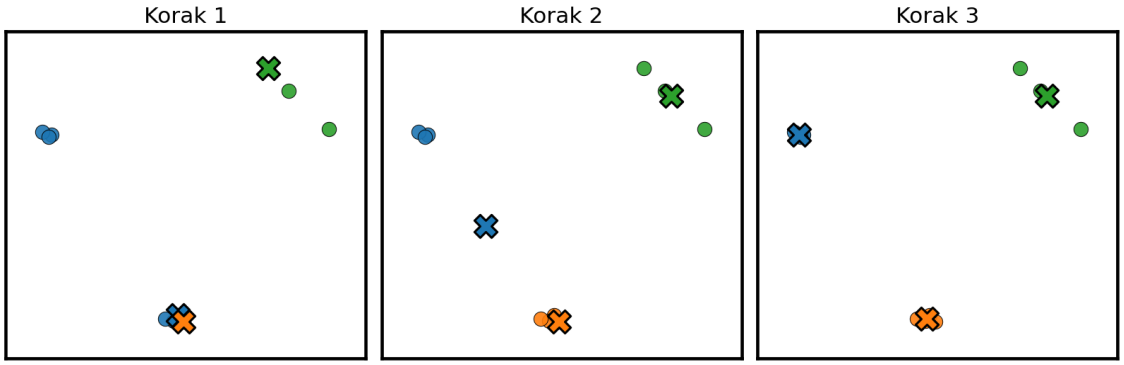
1. Prvi centar biran proizvoljno (g_1).
2. Drugi centar je tačka najudaljenija od izabranih (g_9).
3. Treći centar je opet najudaljeniji od izabranih (g_5).



Slika 6.1: *FarthestFirstTraversal* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

k-sredina (Lloyd). Algoritam naizmenično dodeljuje tačke najbližem centru i pomera centre u centar mase svakog klastera. Iteracije algoritma prikazane su na slici 6.2:

1. Slučajno izabrani početni centri.
2. Prva iteracija: centri se pomeraju u centre mase klastera i tri tačke menjaju klaster.
3. Druga iteracija: centri se ponovo pomeraju, ali nijedna tačka više ne menja klaster, pa je postignuta konvergencija na tri očekivane grupe.



Slika 6.2: *k-sredina (Lloyd)* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

Radi ilustracije načina na koji je interaktivni deo lekcije realizovan, u nastavku je prikazan kôd kojim se generišu koraci algoritma *k-sredina*, uz pomoćnu funkciju `assign` koja svaku tačku dodeljuje najbližem centru. Funkcija `kmeans_steps` ne vraća samo konačnu podelu, već čitav niz međukoraka: za svaki korak pamti položaje centara, dodelu tačaka klasterima i tekstualni opis promene, čime se omogućava zaustavljanje i pregled izvršavanja korak po korak. Sve slike sa 2D prikazom gena u ovom radu (slike 5.1, 5.2 i prikazi korak po korak u ovoj glavi) generisane su iz prateće *Jupyter sveske*, elektronske lekcije [8], pokretanjem njenog vizualizacionog koda.

```
def assign(data, centers): # dodela najblizem centru
    labels = []
    for p in data:
        d = [euclidean(p, c) for c in centers]
        labels.append(d.index(min(d)))
    return labels

def kmeans_steps(data, k, seed=0, max_iter=20):
    """Lloyd k-sredina; vraca listu koraka [(centri, oznake, opis), ...]."""
    rnd = random.Random(seed)
    centers = [list(p) for p in rnd.sample(data, k)] # slucajni pocetni centri
    history = []
    labels = assign(data, centers)
    history.append((centers, labels, "Korak 0: inicijalni centri"))
    for it in range(1, max_iter + 1):
        new_centers = [] # pomeranje u tezista
        for j in range(k):
            grp = [data[i] for i, l in enumerate(labels) if l == j]
            new_centers.append(centroid_of(grp) if grp else centers[j])
        new_labels = assign(data, new_centers)
        changed = sum(1 for a, b in zip(labels, new_labels) if a != b)
        history.append((new_centers, new_labels,
            f"Korak {it}: {changed} tacaka promenilo klaster"))
        if changed == 0: # konvergencija
            break
        centers, labels = new_centers, new_labels
    return history
```

Listing 6.1: Generisanje koraka algoritma *k-sredina* u lekciji.

Napomena. Prikazana funkcija `kmeans_steps` namenjena je vizualizaciji, pa vraća ceo niz međukoraka. Pored nje, sveska sadrži i standardnu implementaciju svakog

algoritma (klase poput `KMeans`, sa metodom `fit`) koja ne prati izvršavanje korak po korak, već vraća samo konačan rezultat, spisak klastera odnosno oznake kojima su tačke dodeljene.

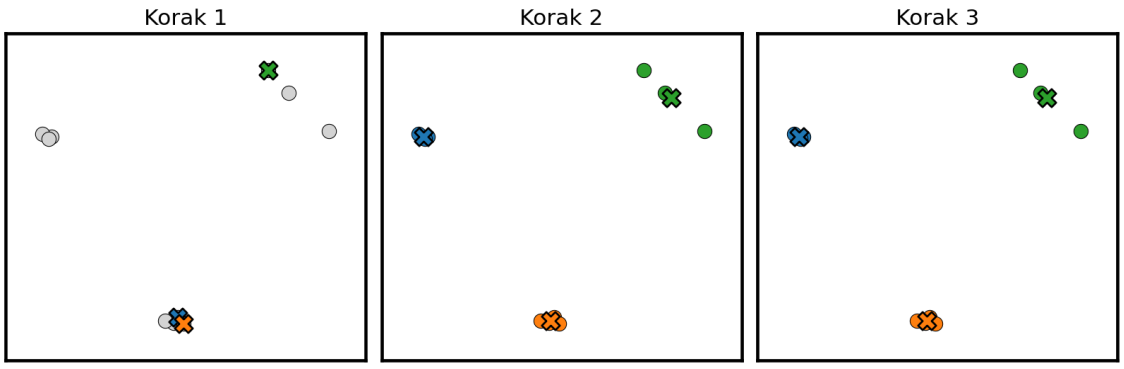
Niz koraka izračunava se jednom i čuva (na primer `km_history = kmeans_steps(data, k)`). Funkcija `next_kmeans(data, km_history)` služi za interaktivni prikaz: pri svakom pozivu (pritiskom na *Shift+Enter*) prikazuje naredni korak iz niza `km_history`, uz grafički prikaz i prateći tekstualni ispis, na primer:

```
============================================================
  K-means (Lloyd) korak 2 / 3
------------------------------------------------------------
  [#################--------] 67%
  Korak 1: centri pomereni; 2 tacaka promenilo klaster
------------------------------------------------------------
  Shift+Enter -> sledeci korak
============================================================
```

Listing 6.2: Primer tekstualnog ispisa jednog koraka.

Soft k-sredina (EM). Algoritam umesto tvrde dodele računa pripadnosti svake tačke prema svim centrima. Koraci algoritma su prikazani na slici 6.3:

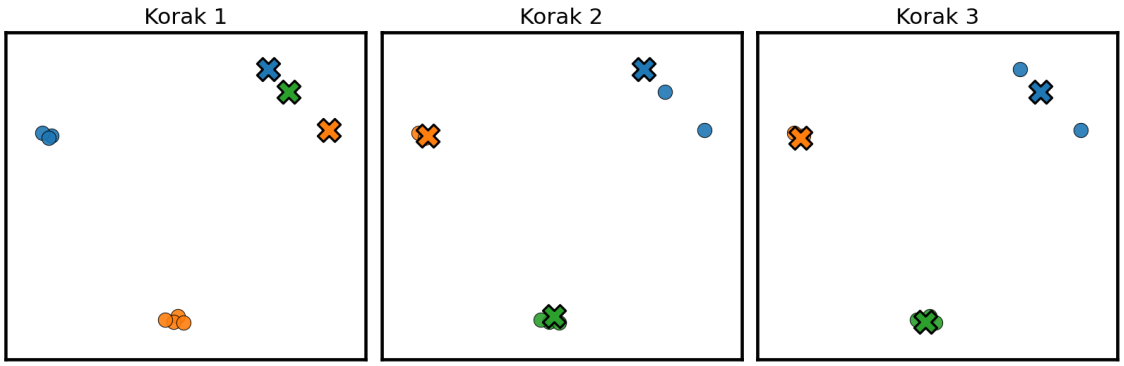
1. Slučajno izabrani početni centri.
2. Međukorak EM iteracija (pomak centara $\approx 2{,}69$).
3. Konvergencija (pomak $\approx 0{,}00$), pripadnosti postaju gotovo binarne.



Slika 6.3: *Soft k-sredina (EM)* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

Brute-force k-sredina. Prolazi kroz sve moguće podele i pamti onu sa najmanjom distorzijom. Razmatrane podele prikazane su na slici 6.4:

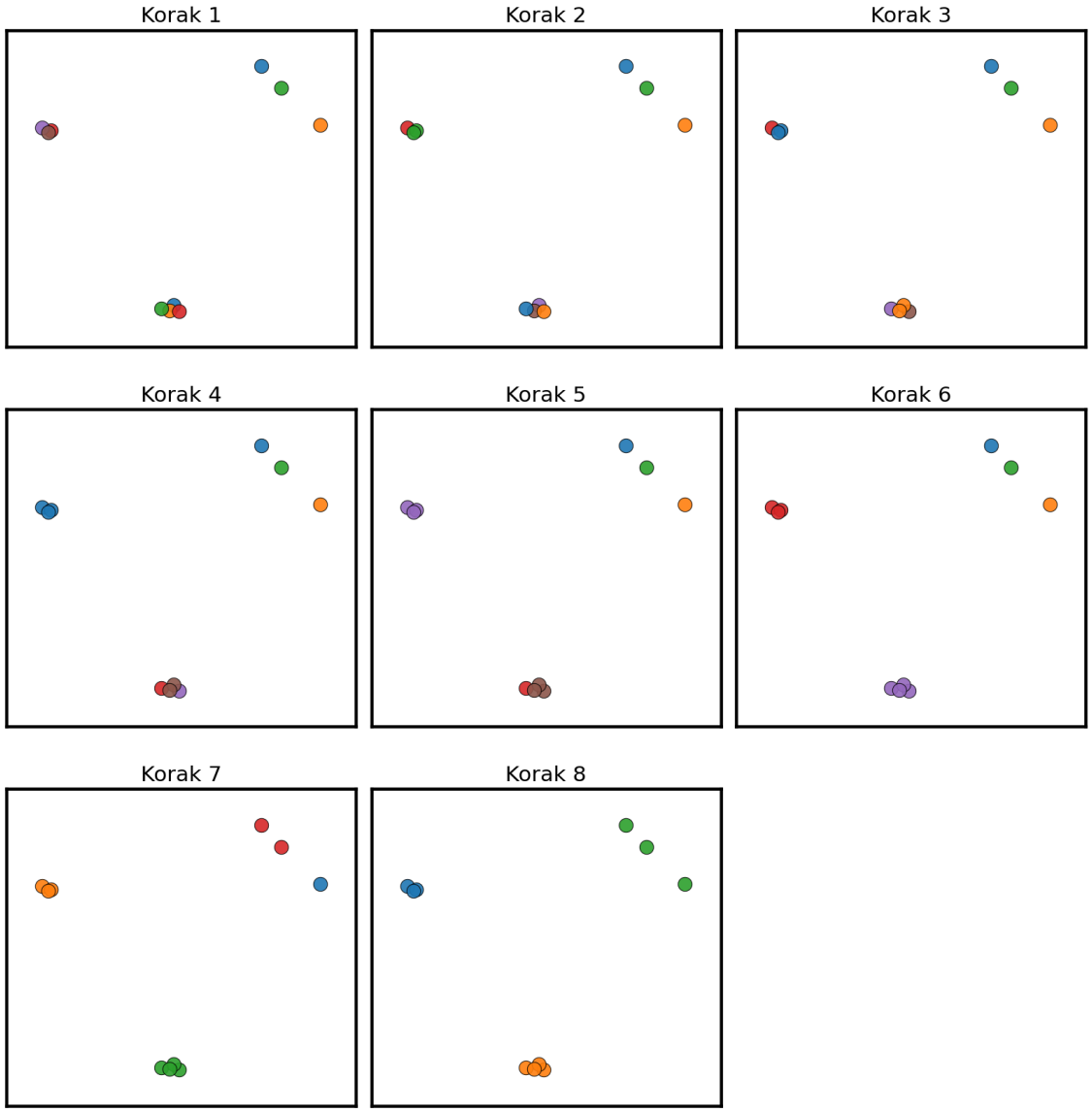
1. Prva razmatrana podela (distorzija ≈ 27).
2. Pronađena znatno bolja podela (distorzija $\approx 2{,}6$).
3. Najbolja pronađena podela (distorzija $\approx 2{,}0$) daje tri grupe.



Slika 6.4: *Brute-force k-sredina* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

***UPGMA* (aglomerativno).** Algoritam spaja redom dva najbliža klastera, počev od jednočlanih. Redosled spajanja klastera prikazan je na slici 6.5:

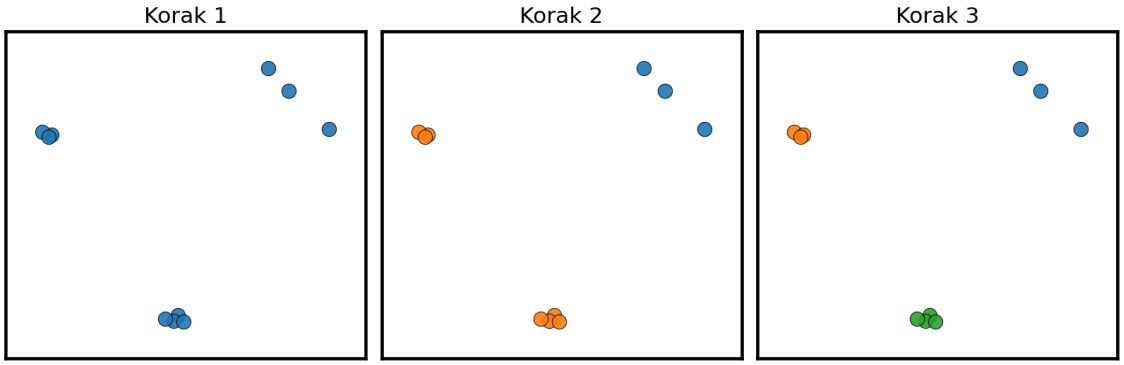
1. Svaka tačka je svoj klaster (10 klastera).
2. Spajaju se g_4 i g_6 ($D_{\text{avg}} \approx 0{,}05$) $\rightarrow$ 9 klastera.
3. Spajaju se g_7 i g_8 ($D_{\text{avg}} \approx 0{,}12$) $\rightarrow$ 8 klastera.
4. g_5 se pridružuje klasteru $\{g_4, g_6\}$ ($D_{\text{avg}} \approx 0{,}14$) $\rightarrow$ 7 klastera.
5. g_{10} se pridružuje klasteru $\{g_7, g_8\}$ ($D_{\text{avg}} \approx 0{,}14$) $\rightarrow$ 6 klastera.
6. g_9 se pridružuje klasteru $\{g_7, g_8, g_{10}\}$ ($D_{\text{avg}} \approx 0{,}22$) $\rightarrow$ 5 klastera.
7. Spajaju se g_1 i g_3 ($D_{\text{avg}} \approx 0{,}50$) $\rightarrow$ 4 klastera.
8. g_2 se pridružuje klasteru $\{g_1, g_3\}$ ($D_{\text{avg}} \approx 1{,}14$) $\rightarrow$ 3 klastera (tri očekivane grupe).



Slika 6.5: *UPGMA (aglomerativno)* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

***DIANA* (divizivno).** Algoritam gradi hijerarhiju klastera odozgo naniže: kreće od jednog klastera koji zatim uzastopno deli. Redosled podela prikazan je na slici 6.6:

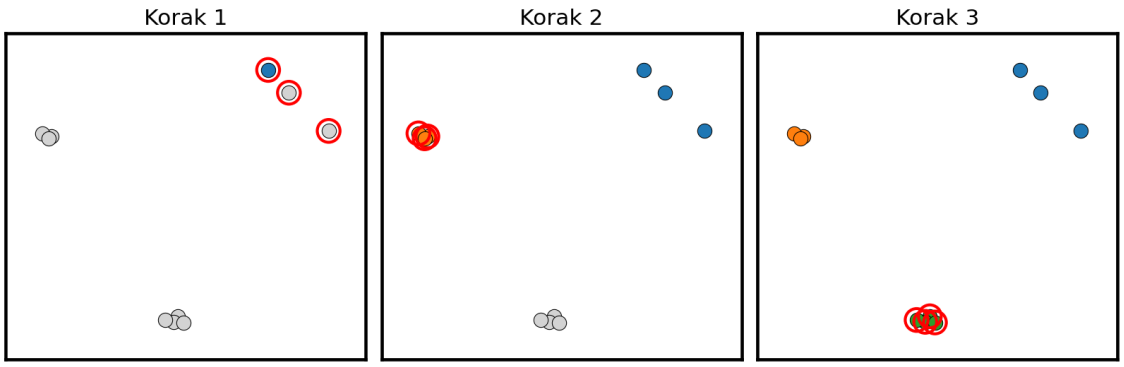
1. Svih deset gena u jednom klasteru.
2. Prva podela (prečnik $\approx 4{,}65$) daje 2 klastera.
3. Druga podela (prečnik $\approx 3{,}83$) daje 3 klastera.



Slika 6.6: *DIANA (divizivno)* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

DBSCAN. Algoritam proširuje klastere polazeći od jezgrenih (*core*) tačaka preko ε-okoline. Proširivanje klastera prikazano je na slici 6.7:

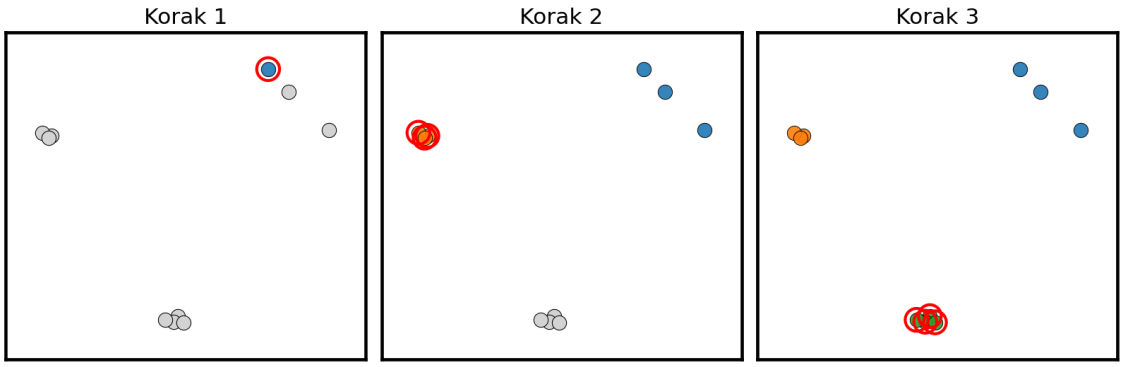
1. Tačka iz jezgra otvara prvi klaster (regulisani na gore).
2. Širi se drugi klaster (regulisani na dole).
3. Otvara se i treći klaster (nepromenjeni); ukupno tri klastera, bez šuma.



Slika 6.7: *DBSCAN* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

CAST. *CAST* gradi klastere jedan po jedan; svaki klaster počinje od jednog početnog gena i raste dodavanjem θ-bliskih gena, dok se ne ustali. Rast klastera prikazan je na slici 6.8:

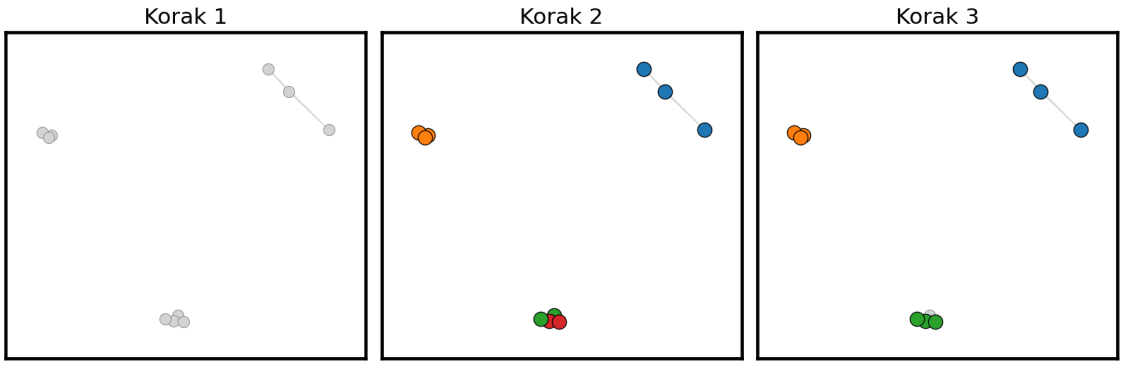
1. Prvi klaster počinje od gena g_1.
2. Raste drugi klaster (dodaju se θ-bliski geni).
3. Formira se treći klaster.



Slika 6.8: *CAST* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

Louvain. Lokalnim premeštanjem čvorova maksimizuje modularnost, pa agregira zajednice. Faze algoritma prikazane su na slici 6.9:

1. Svaka tačka je zasebna zajednica.
2. Lokalna premeštanja spajaju čvorove u zajednice.
3. Posle svih premeštanja ostaju tri zajednice.



Slika 6.9: *Louvain* korak po korak na 2D projekciji deset gena (numeracija koraka odgovara prethodno opisanom redosledu koraka).

6.2 Korišćenje alata veštačke inteligencije

Tokom izrade ovog rada korišćen je alat veštačke inteligencije *Claude* (kompanije Anthropic). Alat je korišćen za generisanje slika, ispravke teksta, pronalaženje referenci i grafički prikaz rezultata pri implementaciji algoritama. Idejni koncept rada, matematičke formulacije i implementacije algoritama predstavljaju samostalan rad autorke.

Glava 7

Zaključak

Ovaj rad daje pregled algoritama klasterovanja koji se koriste u analizi podataka o ekspresiji gena. Kroz teorijsko izlaganje, vizualizacije i primere prikazane su prednosti i nedostaci svakog algoritma, kao i to kako različiti algoritmi isti skup gena mogu podeliti na različite načine.

U prvom delu uvedeni su osnovni biološki pojmovi (centralna dogma, ekspresija gena, matrica genske ekspresije i diauksična promena kod kvasca) i objašnjeno je kako se klasterovanjem traže grupe koregulisanih gena. Zatim su definisane mere bliskosti (euklidsko rastojanje i Pirsonov koeficijent korelacije) i funkcija cilja klasterovanja. Zatim je opisano deset algoritama iz četiri klase algoritama klasterovanja, uglavnom prema knjizi *Bioinformatics Algorithms: An Active Learning Approach* autora Filipa Kompoa (*Phillip Compeau*) i Pavela Pevznera (*Pavel Pevzner*) [6]: algoritmi zasnovani na centrima (*FarthestFirstTraversal*, *k-sredina / Lloyd*, *Soft k-sredina / EM* i *brute-force k-sredina*), algoritmi hijerarhijskog klasterovanja (*UPGMA*, aglomerativno i *DIANA*, divizivno), algoritmi zasnovani na gustini (*DBSCAN*) i grafovski algoritmi za detekciju zajednica (*CAST*, *Louvain* i *Leiden*). Za svaki algoritam navedena je funkcija cilja koju minimizuje (ili, kod algoritama bez eksplicitne funkcije cilja, kriterijum prema kome se formiraju klasteri, poput gustinske povezanosti ili praga sličnosti), pseudokod algoritma i računska složenost, uz sliku koja ilustruje njegov rad ili ograničenje.

Glavni cilj rada bila je izrada elektronske lekcije. Lekcija je realizovana kao interaktivna *Jupyter sveska* koja se sastoji od teorijskog i interaktivnog dela. Teorijski deo prati tekst pisanog rada, dok interaktivni deo omogućava pokretanje algoritama korak po korak, pri čemu se svaki korak vizualizuje na konkretnom skupu podataka. Kompletan kod je pisan u programskom jeziku *Python*, uz upotrebu samo stan-

dardne biblioteke za implementaciju samih algoritama i biblioteke *matplotlib* [2] za vizualizaciju. Ovako zamišljena lekcija može poslužiti predavačima kao nastavno sredstvo, a studentima za samostalno učenje i proučavanje algoritama klasterovanja u interaktivnom okruženju.

Postoji dosta prostora za dalja poboljšanja. Fokus je bio na algoritmima koji se obrađuju na kursu *Uvod u bioinformatiku*, pa mnoge modifikacije i ubrzanja pomenutih algoritama nisu obrađeni - na primer, korišćenje prostornih indeksa za *DBSCAN*, drugačija pravila povezivanja kod hijerarhijskih algoritama ili automatski izbor parametara algoritma. Kao nastavak rada, lekcija bi se mogla proširiti novim algoritmima i primenom na veće, stvarne skupove podataka o ekspresiji gena.

Bibliografija

[1] Jupyter dokumentacija, 2026. zvanični sajt: `https://docs.jupyter.org/` (pristupljeno: 26. 8. 2026.).

[2] Matplotlib dokumentacija, 2026. zvanični sajt: `https://matplotlib.org/` (pristupljeno: 26. 8. 2026.).

[3] Python dokumentacija, 2026. zvanični sajt: `https://docs.python.org/3/` (pristupljeno: 26. 8. 2026.).

[4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell*. Garland Science, New York, 4 edition, 2002.

[5] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[6] Phillip Compeau and Pavel Pevzner. *Bioinformatics Algorithms: An Active Learning Approach*. Active Learning Publishers, 2018. zvanični sajt: `https://www.bioinformaticsalgorithms.org/` (pristupljeno: 26. 8. 2026.).

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[8] Milena Filipović. Klasterovanje u bioinformatici – elektronska lekcija (jupyter sveska), 2026. izvorni kod i sveska: `https://github.com/Milena-99/bioinformatics_clustering` (pristupljeno: 26. 8. 2026.).

[9] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 2 edition, 1994.

[10] Jovana Kovačević. Uvod u bioinformatiku, prezentacije sa predavanja. Univerzitet u Beogradu, Matematički fakultet, 2026. zvanični sajt: `http://www.bioinformatika.matf.bg.ac.rs/` (pristupljeno: 26. 8. 2026.).

[11] Jiří Matoušek and Jaroslav Nešetřil. *Invitation to Discrete Mathematics*. Oxford University Press, 2 edition, 2008.

[12] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[13] Microbe Notes. Gene expression, 2023. zvanični sajt: `https://microbenotes.com/gene-expression/` (pristupljeno: 26. 8. 2026.).

[14] Maurice Roux. A comparative study of divisive and agglomerative hierarchical clustering algorithms. *Journal of Classification*, 35:345–366, 2018.

[15] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9:5233, 2019.

Biografija autora

Milena Filipović rođena je 27. oktobra 1999. godine. Završila je 2014. osnovnu školu „Rajko Mihailović" u Banjanima, a potom 2018. godine gimnaziju „Branislav Petronijević" u Ubu. Iste godine upisala je Matematički fakultet Univerziteta u Beogradu, na kome je 2022. završila osnovne akademske studije. Tokom školovanja učestvovala je na mnogobrojnim takmičenjima iz matematike, srpskog jezika, istorije i geografije. Od 2022. godine zaposlena je u kompaniji Procescom d.o.o. na poziciji programera, pretežno na razvoju mobilnih mreža kao *network core* programer, na projektima 4G, 5G i VoLTE.