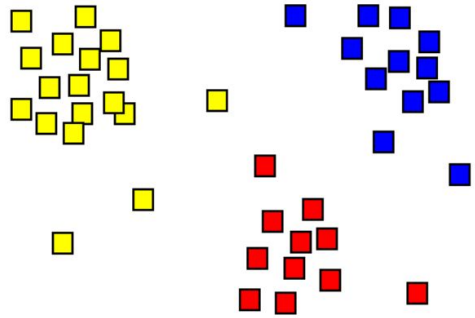


Algoritmi v bioinformatiki
Gručenje

Martin Milanič
martin.milanic@upr.si
UP FAMNIT

23. april 2025

Literatura za to poglavje:

- **Compeau-Pevzner**, Bioinformatics Algorithms: An Active Learning Approach, poglavje 8
  How Did Yeast Become a Wine Maker?

- **Jones-Pevzner**, An introduction to Bioinformatics Algorithms, poglavje 10
  Clustering and Trees

Problem gručenja

Pogost problem v biologiji je naslednji:

Kako razdeliti množico eksperimentalno pridobljenih podatkov na skupine (gruče), tako da bodo podatki znotraj iste gruče čim bolj podobni drug drugemu, podatki v različnih gručah pa čim bolj različni?

Ta problem **gručenja** (angl. clustering) nima enostavne rešitve.

Obravnavali bomo številne pristope k problemu.

Uporabe gručenja

- Identifikacija družin genov s podobnimi funkcijami, tudi ko ni povsem jasno, kakšna je vloga posameznih genov.

  Pri tem se uporabi **analiza izražanja genov**.

- Rekonstrukcija filogenetskih dreves.

- Poravnava več zaporedij.

Analiza izražanja genov

Ugotavljanje funkcij genov poteka dandanes s pomočjo **mikromrež** (DNA microarrays), ki analizirajo ravni izražanja genov (količino proizvedene mRNA) pod različnimi pogoji in ob različnih časih.

Na ta način ugotovimo, kateri geni so ob katerem času aktivni v celici.

Podatki takega eksperimenta so zbrani v t.i.

matriki izražanja, $\mathcal{I}$,

ki ima n vrstic, po eno za vsak gen, in m stolpcev, po enega za vsako meritev.

Element matrike

$$\mathcal{I}(i,j)$$

predstavlja raven izražanja i-tega gena v j-tem poskusu.

Celotna i-ta vrstica predstavlja t.i. **vzorec izražanja** i-tega gena.

Ideja: če imata dva gena podobna vzorca izražanja, potem sta zelo verjetno tudi biološko povezana (bodisi opravljata isto funkcijo ali sta prisotna pri istem biološkem procesu).

Algoritmi za gručenje tvorijo gruče paroma podobnih genov.

Iz matrike izražanja izračunajo **matriko razdalj**

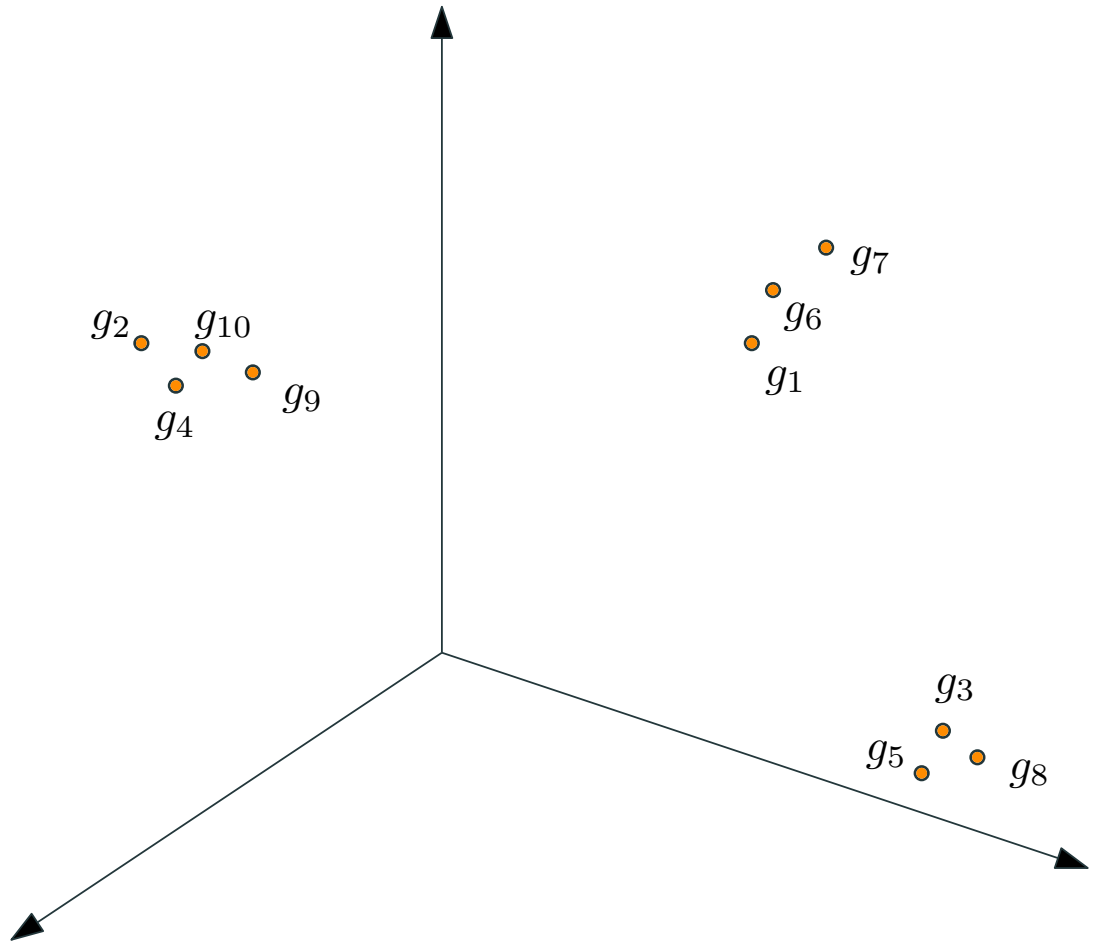
$$D = D_{ij} \, ,$$

kjer D_{ij} odraža, kako podobna sta vzorca izražanja genov i in j.

Zgled:

Matrika izražanja $\mathcal{I}$:

Čas	1h	2h	3h
g_1	10	8	10
g_2	10	0	9
g_3	4	9	3
g_4	9	1	9
g_5	5	8	3
g_6	11	9	12
g_7	5	9	11
g_8	3	9	2
g_9	10	2	9
g_{10}	10	1	9



Če za razdaljo vzamemo kar evklidsko razdaljo med točkami v $\mathbb{R}^3$, potem je

$$D_{12} = \sqrt{(10-10)^2 + (8-0)^2 + (10-9)^2} = \sqrt{0^2 + 8^2 + 1^2} \approx 8.1\,,$$

itd.

Matrika razdalj D

(približek; prikazana je le polovica nad glavno diagonalo):

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0	**8,1**	9,3	7,1	8,6	2,4	5,2	11	6,1	7,1
g_2		0	12	1,4	11	9,5	10	13	2	1
g_3			0	11	1,4	11	8,1	1,4	11	12
g_4				0	10	8,8	9,2	12	1,4	1
g_5					0	11	8,1	2,4	9,8	10
g_6						0	6,1	13	7,7	8,6
g_7							0	9,2	8,8	9,6
g_8								0	12	13
g_9									0	1
g_{10}										0

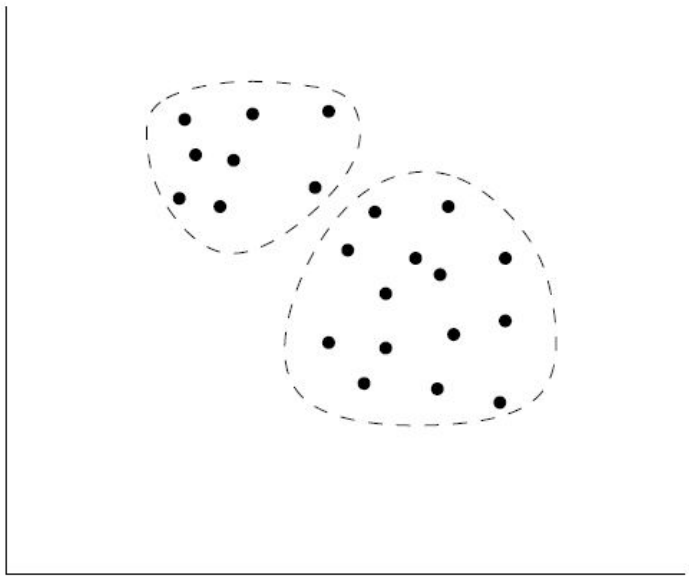
Kdaj je gručenje dobro?

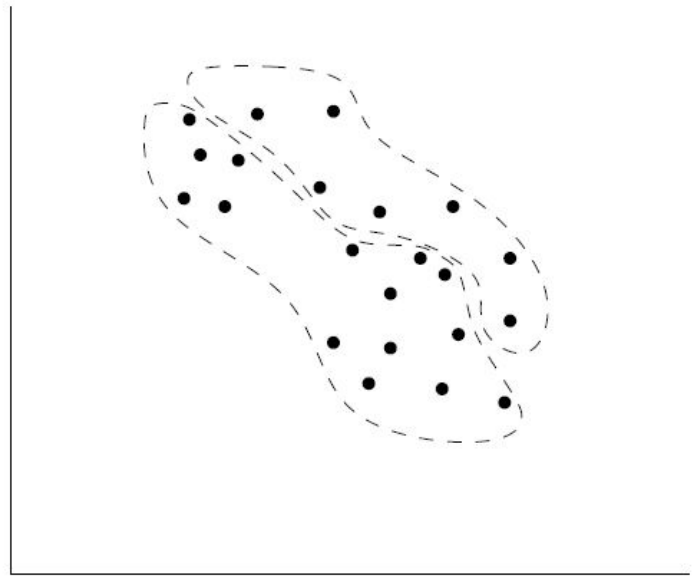
Dobro gručenje mora zadoščati naslednjima pogojema:

- **Homogenost.**
  Vzorci izražanja znotraj iste gruče morajo biti paroma podobni,
  tj. vrednosti D_{ij} morajo biti **majhne** za vse pare (i, j), kjer sta i in j iz iste gruče.

- **Separacija.**
  Vzorci izražanja iz različnih gruč morajo biti čim bolj različni, tj. vrednosti D_{ij} morajo biti **velike** za vse pare (i, j), kjer sta i in j iz različnih gruč.

Zgled dobrega (levo) in slabega gručenja (desno):





Hierarhično gručenje

Hierarhično gručenje je tehnika, ki organizira elemente v drevo.

- Ne prikazuje le ene razdelitve dane množice na gruče, temveč predstavlja družino takih razdelitev.
- Gruče imajo podgruče, te imajo spet svoje podgruče itd.

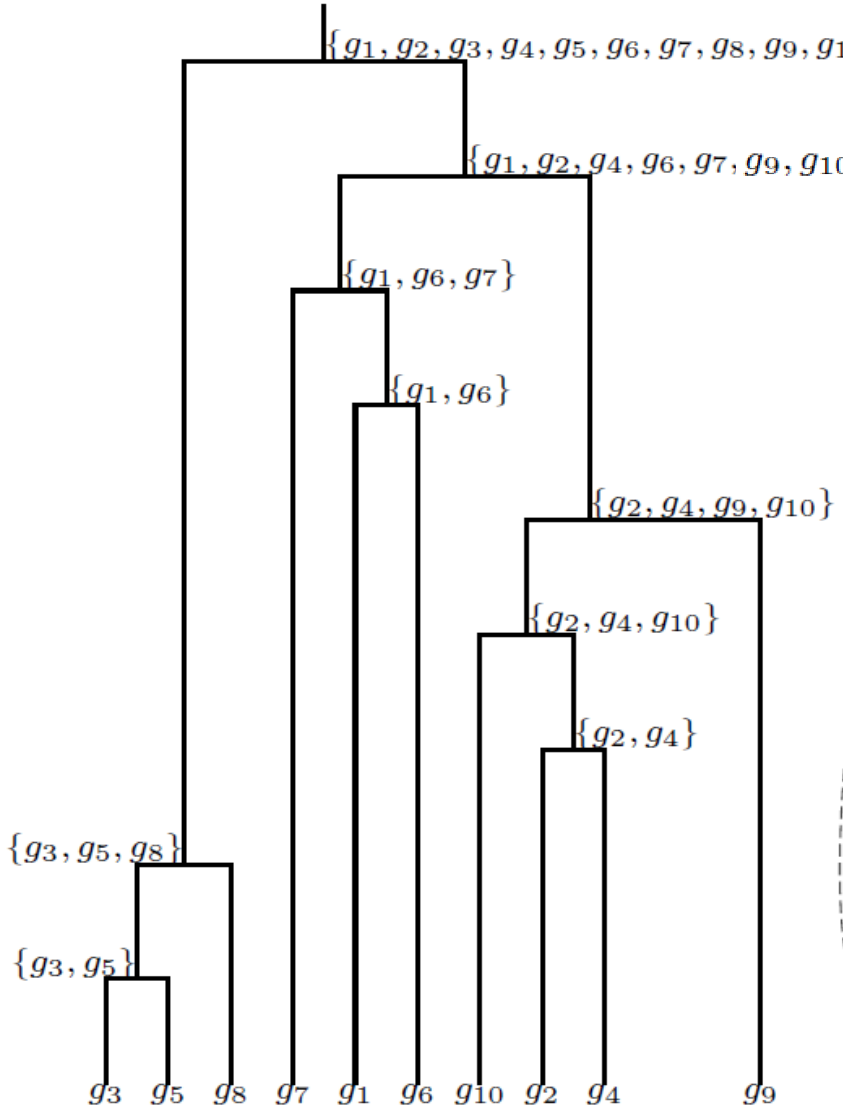
Zgled:

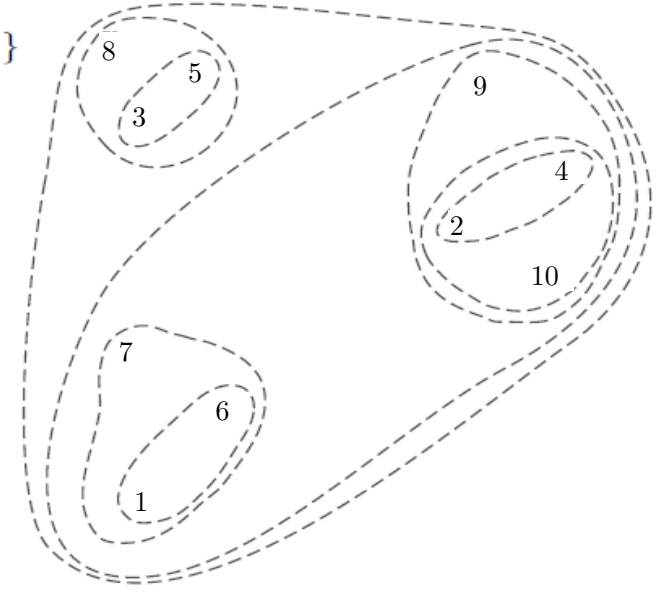
Sesalci se delijo na primate, zveri, netopirje, vrečarje, itd.

Red **Zveri** se nadalje deli na mačke, hijene, medvede, tjulnje, itd.

Mačke pa se še nadalje delijo na 37 vrst. ▲

Zgled hierarhičnega gručenja in pripadajočega **dendrograma** (drevesa):





Prirejeno po: Jones-Pevzner, Introduction to Bioinformatics Algorithms, str. 344

Če nadaljujemo prejšnji zgled o izražanju genov, lahko drevo hierarhičnega gručenja opišemo takole:

- Geni so na listih.
- Vsaka povezava na drevesu ima neko dolžino.
- Razdalje med listi so sorazmerne z elementi matrike razdalj.

Za izračun drevesa iz matrike razdalj obstaja več metod, na primer:

- **UPGMA** (Unweighted Pair Group Method with Arithmetic Averages)
- **NJ** (Neighbor Joining) – to metodo uporabljajo programi *Clustal* (vrsta programov za poravnavo več zaporedij)
- **Rekonstrukcija dreves iz aditivnih matrik**

  (ta metoda je direktno uporabna za konstrukcijo filogenetskih dreves, ob t.i. predpostavki *molekularne ure*)

Metoda **UPGMA** je poseben primer naslednjega splošnega algoritma.

Algoritem Hierarhično Gručenje

Vhod: razdaljna matrika D velikosti $n \times n$

Izhod: Drevo T, ki ponazarja hierarhično gručenje n elementov, predstavljenih z D.

Ustvari n gruč s po enim elementom.

Konstruiraj graf T, tako da vsaki gruči priredíš izolirano točko.

while obstaja več kot ena gruča

Poišči dve najbližji si gruči C_1 in C_2.

Združi C_1 in C_2 v novo gručo C s $|C_1| + |C_2|$ elementi.

Izračunaj razdaljo od C do vseh ostalih gruč. $(*)$

Dodaj novo točko C grafu T in jo poveži s točkama C_1 in C_2.

Odstrani iz D vrstici in stolpca, ki ustrezata C_1 in C_2.

Dodaj v D vrstico in stolpec za novo gručo C.

end while

return T

Izračun razdalje v $(*)$ je mogoč na več različnih načinov, npr.:

- najmanjša razdalja med poljubnima elementoma:

$$d_{min}(C^*, C) = \min_{x \in C^*, y \in C} d(x, y)\,,$$

- povprečna razdalja:

$$d_{avg}(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y)\,.$$

- razdalja na osnovi separacije gruč C_1 in C_2:

$$d(C^*, C) = \frac{d(C^*, C_1) + d(C^*, C_2) - d(C_1, C_2)}{2}\,.$$

Različni izračuni razdalj v splošnem vodijo do različnih dreves T.

Hierarhično gručenje: metoda UPGMA

Algoritem UPGMA (Sokal-Michener 1958)

Vhod: Matrika razdalj D velikosti $n \times n$

Izhod: Drevo T, ki ponazarja hierarhično gručenje n elementov, predstavljenih z D.

Ustvari n gruč s po enim elementom.

Ustvari graf T, tako da vsaki gruči prirediš po eno (izolirano) točko C, za katero velja $h(C) = 0$.
while obstaja več kot ena gruča

1. Poišči dve najbližji si gruči C_1 in C_2.
2. Združi C_1 in C_2 v novo gručo C s $|C_1| + |C_2|$ elementi.
3. Za vse gruče $C^* \neq C$ izračunaj $D(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y)$
4. Grafu T dodaj novo točko C in jo poveži s točkama C_1 in C_2.
5. $h(C) \leftarrow \frac{D(C_1, C_2)}{2}$.
6. Dodeli dolžino $h(C) - h(C_1)$ povezavi (C_1, C)
7. Dodeli dolžino $h(C) - h(C_2)$ povezavi (C_2, C)
8. Odstrani iz D vrstici in stolpca, ki ustrezata C_1 in C_2.
9. Dodaj v D vrstico in stolpec za novo gručo C.

end while
return T

Zgled

Dana je naslednja matrika razdalj:

	g_1	g_2	g_3	g_4
g_1	0	8	7	12
g_2		0	9	8
g_3			0	11
g_4				0

Začnemo z gručami s po enim elementom:

$C_1 = \{g_1\}$,

$C_2 = \{g_2\}$,

$C_3 = \{g_3\}$,

$C_4 = \{g_4\}$.

	C_1	C_2	C_3	C_4
C_1	0	8	7	12
C_2		0	9	8
C_3			0	11
C_4				0

Poiščemo najbližji si gruči. To sta C_1 in C_3.

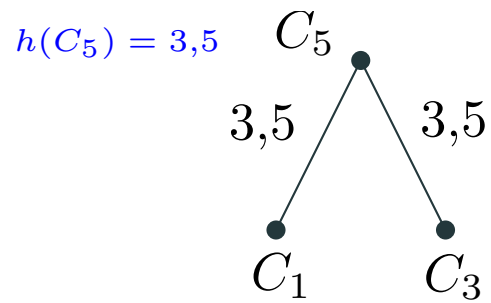
Združimo $C_1 = \{g_1\}$ in $C_3 = \{g_3\}$ v novo gručo $C_5 = \{g_1, g_3\}$.

Povežemo novo gručo C_5 z gručama C_1 in C_3.

Določimo višino nove točke: $h(C_5) = 7/2 = 3,5$.

Izračunamo dolžini novih povezav na drevesu.

	C_1	C_2	C_3	C_4
C_1	0	8	**7**	12
C_2		0	9	8
C_3			0	11
C_4				0



Izračunamo razdaljo nove gruče C_5 do vseh ostalih gruč (razen C_1 in C_3) z uporabo formule

$$d_{avg}(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y)\,.$$

	C_1	C_2	C_3	C_4
C_1	0	8	7	12
C_2		0	9	8
C_3			0	11
C_4				0

Razdalja od $C_5 = \{g_1, g_3\}$ do $C_2 = \{g_2\}$:

$$d(C_5, C_2) = \frac{1}{2 \cdot 1} \left(d(g_1, g_2) + d(g_3, g_2)\right) = \frac{1}{2}\,(8 + 9) = 8,5\,.$$

Izračunamo razdaljo nove gruče C_5 do vseh ostalih gruč (razen C_1 in C_3) z uporabo formule

$$d_{avg}(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y)\,.$$

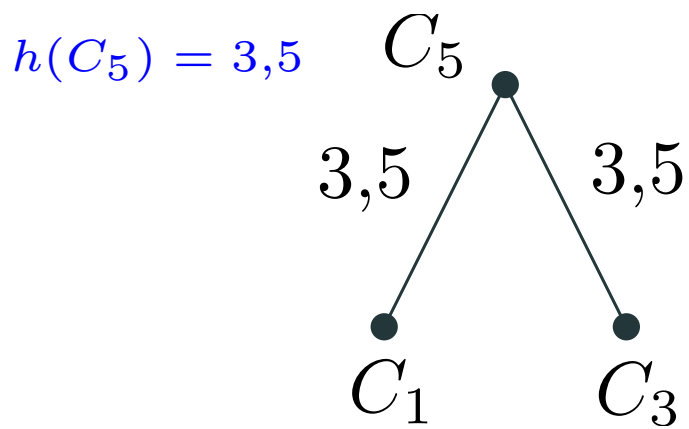
	C_1	C_2	C_3	C_4
C_1	0	8	7	12
C_2		0	9	8
C_3			0	11
C_4				0

Razdalja od $C_5 = \{g_1, g_3\}$ do $C_4 = \{g_4\}$:

$$d(C_5, C_4) = \frac{1}{2 \cdot 1} \left(d(g_1, g_4) + d(g_3, g_4)\right) = \frac{1}{2}\left(12 + 11\right) = 11,5\,.$$

Nova matrika razdalj:

	$C_5 = \{g_1, g_3\}$	$C_2 = \{g_2\}$	$C_4 = \{g_4\}$
$C_5 = \{g_1, g_3\}$	0	8,5	11,5
$C_2 = \{g_2\}$		0	8
$C_4 = \{g_4\}$			0



Postopek ponovimo.

Najbližji si gruči sta C_2 in C_4.

Združimo C_2 in C_4 v novo gručo $C_6 = \{g_2, g_4\}$.

Povežemo gručo C_6 z gručama C_2 in C_4.

Določimo višino nove točke: $h(C_6) = 8/2 = 4$.

Izračunamo dolžini novih povezav na drevesu.

	$C_5 = \{g_1, g_3\}$	$C_2 = \{g_2\}$	$C_4 = \{g_4\}$
$C_5 = \{g_1, g_3\}$	0	$8,5$	$11,5$
$C_2 = \{g_2\}$		0	**8**
$C_4 = \{g_4\}$			0

Postopek ponovimo.

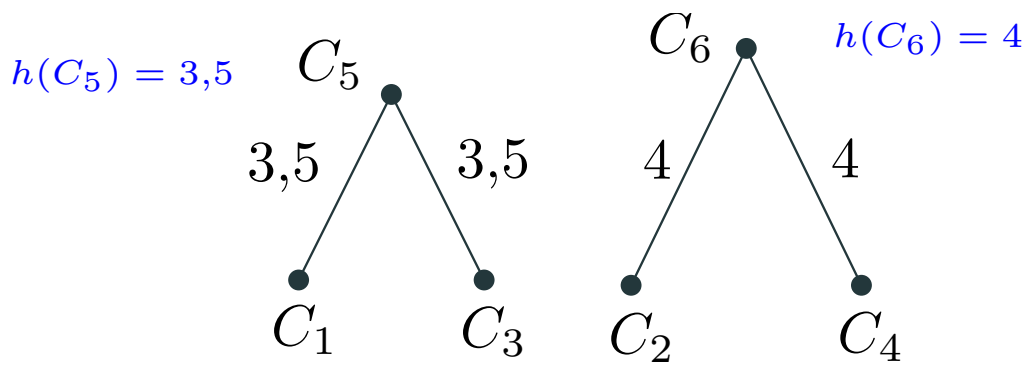
Najbližji si gruči sta C_2 in C_4.

Združimo C_2 in C_4 v novo gručo $C_6 = \{g_2, g_4\}$.

Povežemo gručo C_6 z gručama C_2 in C_4.

Določimo višino nove točke: $h(C_6) = 8/2 = 4$.

Izračunamo dolžini novih povezav na drevesu.



Izračunamo razdaljo nove gruče C_6 do vseh ostalih gruč (razen C_2 in C_4) z uporabo formule

$$d_{avg}(C^*, C) = \frac{1}{|C^*||C|} \sum_{x \in C^*, y \in C} d(x, y)\,.$$

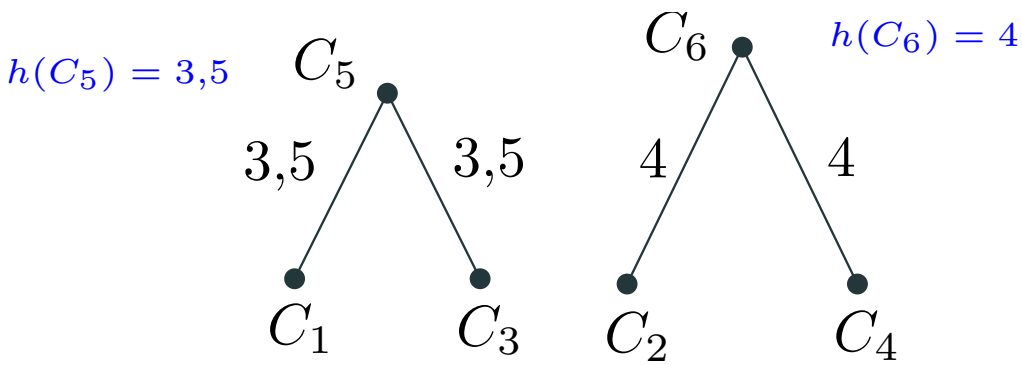
	C_1	C_2	C_3	C_4
C_1	0	8	7	12
C_2		0	9	8
C_3			0	11
C_4				0

Razdalja od $C_6 = \{g_2, g_4\}$ do $C_5 = \{g_1, g_3\}$:

$$\begin{aligned} d(C_6, C_5) &= \frac{1}{2 \cdot 2} \left(d(g_2, g_1) + d(g_2, g_3) + d(g_4, g_1) + d(g_4, g_3)\right) \\ &= \frac{1}{4} \left(8 + 9 + 12 + 11\right) = 10\,. \end{aligned}$$

Nova matrika razdalj:

	$C_5 = \{g_1, g_3\}$	$C_6 = \{g_2, g_4\}$
$C_5 = \{g_1, g_3\}$	0	10
$C_6 = \{g_2, g_4\}$		0



Zadnji korak:

Združimo gruči C_5 in C_6 v novo gručo $C_7 = \{g_1, g_2, g_3, g_4\}$.

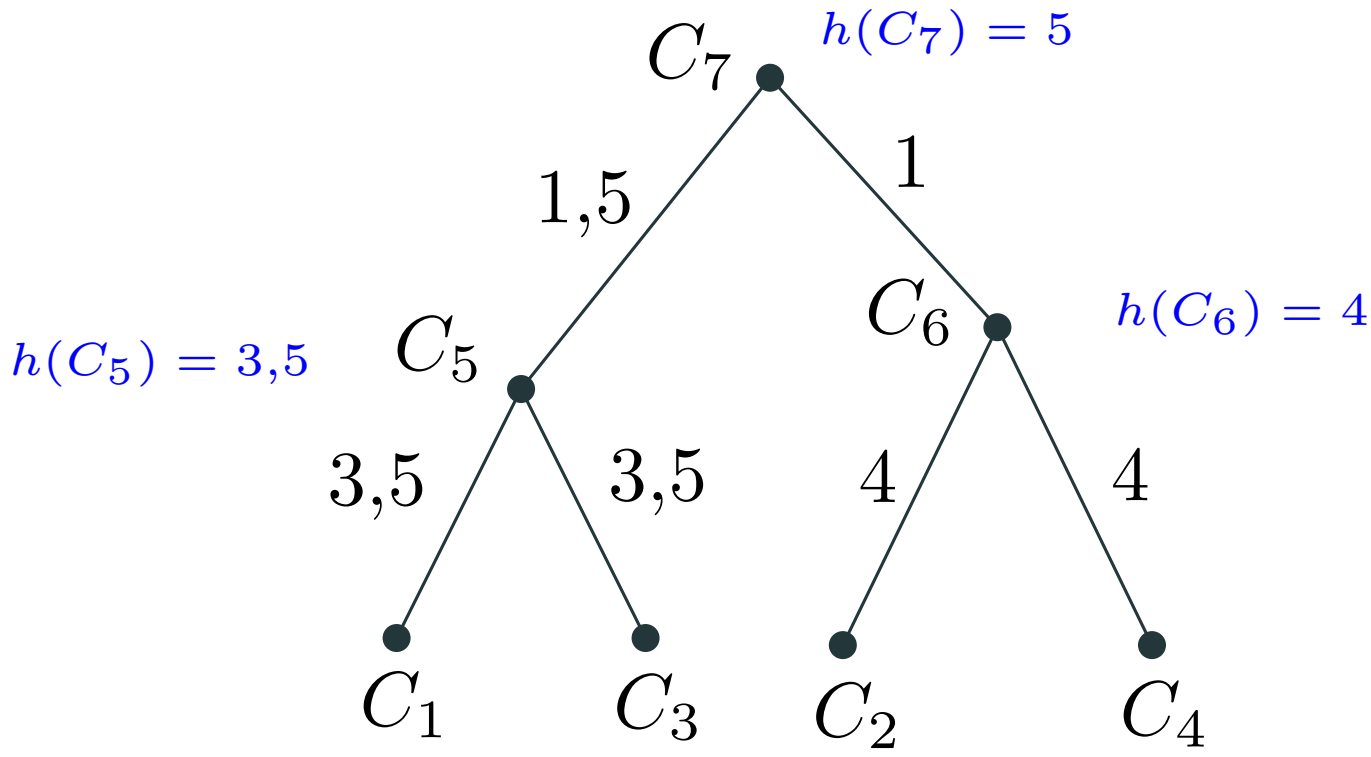
Povežemo gručo C_7 z gručama C_5 in C_6.

Določimo višino nove točke: $h(C_7) = 10/2 = 5$.

Izračunamo dolžini novih povezav na drevesu kot razliko višin krajišč.

	$C_5 = \{g_1, g_3\}$	$C_6 = \{g_2, g_4\}$
$C_5 = \{g_1, g_3\}$	0	**10**
$C_6 = \{g_2, g_4\}$		0

Končni rezultat algoritma:



Metode za (nehierarhično) gručenje

Metoda voditeljev

(angl. k-means clustering)

Ena najpopularnejših metod za gručenje je t.i. **metoda voditeljev**.

Predpostavimo, da je število gruč k znano vnaprej.

Danih je n točk $v_1, \ldots, v_n$ v m-dimenzionalnem prostoru.

Določiti želimo takih k točk

$$X = \{x_1, \ldots, x_k\}$$

v m-dimenzionalnem prostoru—t.i. **voditelji**—,

da bo vrednost naslednje kriterijske funkcije najmanjša možna:

$$\frac{\sum_{i=1}^{n} d(v_i, X)^2}{n},$$

kjer je $d(v_i, X) = \min_{1 \leq j \leq k} d(v_i, x_j)$ razdalja od točke v_i do najbližjega centra.

Metoda voditeljev (Lloydov algoritem):

1. Določi k **voditeljev** poljubno (npr. z naključno izbiro k točk izmed danih točk).

2. Dodeli vsako točko k njej najbližjemu voditelju.

Na ta način dobimo k **gruč, po eno za vsakega voditelja**.

3. Ponastavi vsakega voditelja v težišče

$$\frac{1}{|C|} \sum_{v \in C} v$$

ustrezne gruče C in pojdi na korak 2.

Postopek ponavljamo, dokler ni sprememba kriterijske funkcije dovolj majhna.

Variacije

Možne so tudi druge izbire kriterijskih funkcij, ki jih minimiziramo:

- $\sum_{i=1}^{n} d(v_i, X)$ (angl. k-**median clustering**)
- $\max_{i=1}^{n} d(v_i, X)$ (angl. k-**center clustering**)

Vsi ti kriteriji poudarjajo homogenost, ne zagotavljajo pa dobre separacije.

Za nobenega od teh problemov niso znani učinkoviti algoritmi.

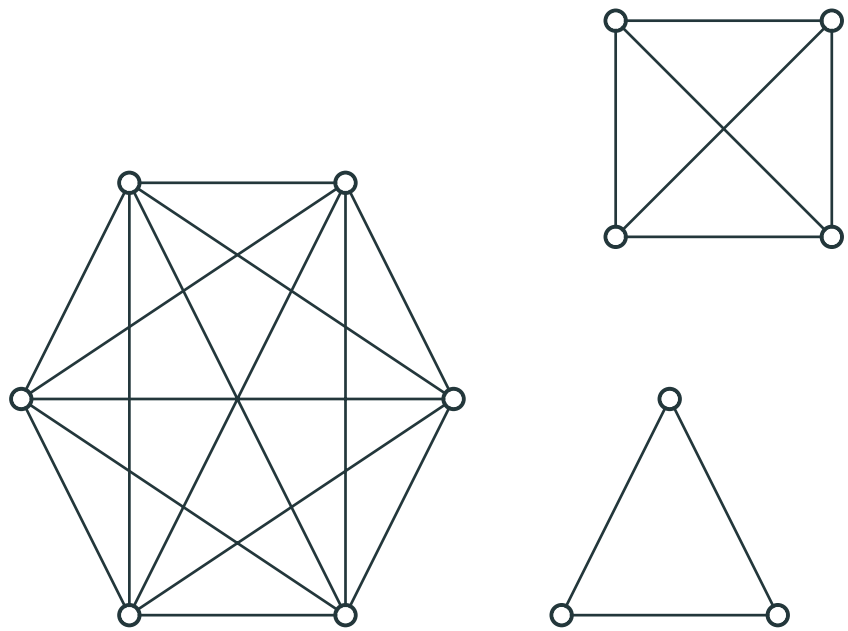
Znane so le hevristike.

“Poškodovane” klike

Poln graf, K_n : graf na n točkah, v katerem sta vsaki dve točki povezani s povezavo

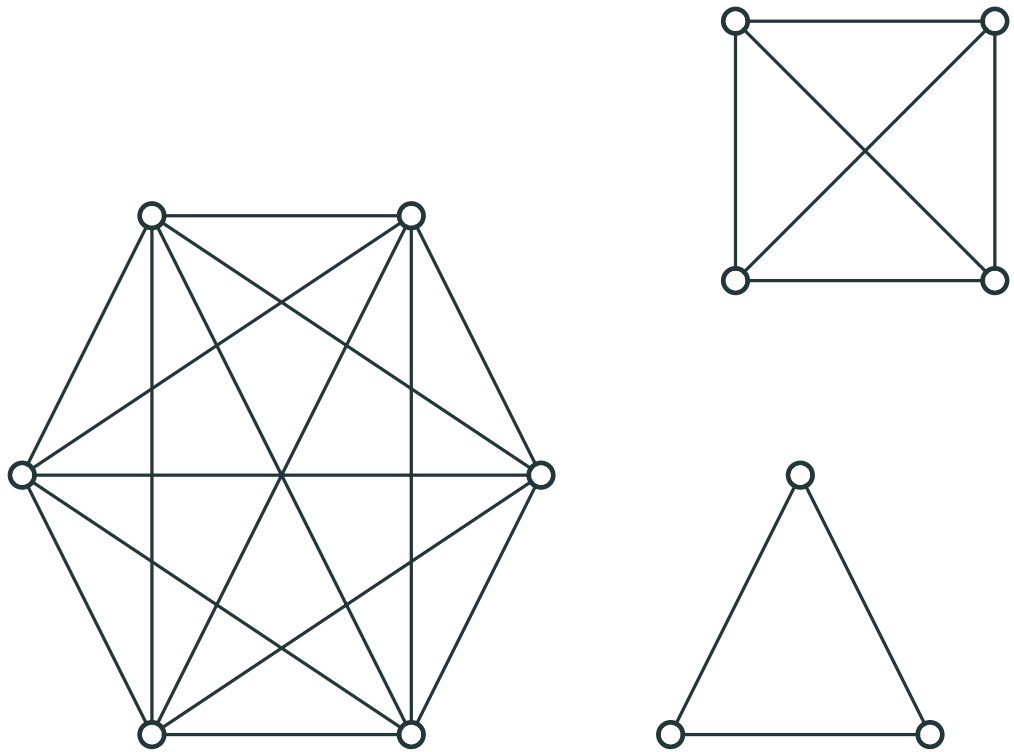
Klika v grafu: poln podgraf

Klični graf: graf, ki ima za povezane komponente same polne grafe



Gručenje n-elementne množice ustreza kličnemu grafu na n točkah:

- točke grafa so elementi množice,
- povezane komponente so polni grafi, ki ustrezajo gručam.



Gene pogosto gručimo na naslednji način:

1. Iz matrike izražanja izračunamo razdaljno matriko D.

2. Tvorimo **razdaljni graf**:

   - točke so geni $1, \ldots, n$,
   - i in j tvorita povezavo natanko tedaj, ko je $D_{ij} < \theta$ (kjer je $\theta > 0$ neka vnaprej izbrana konstanta).

V idealnem primeru dobimo klični graf, iz katerega razberemo iskano gručenje.

Zaradi eksperimentalnih napak pa pogosto dobimo graf, ki ni sestavljen iz samih klik, temveč vsebuje še nekaj dodatnih povezav, kakšna povezava pa mu lahko do kličnega grafa tudi manjka.

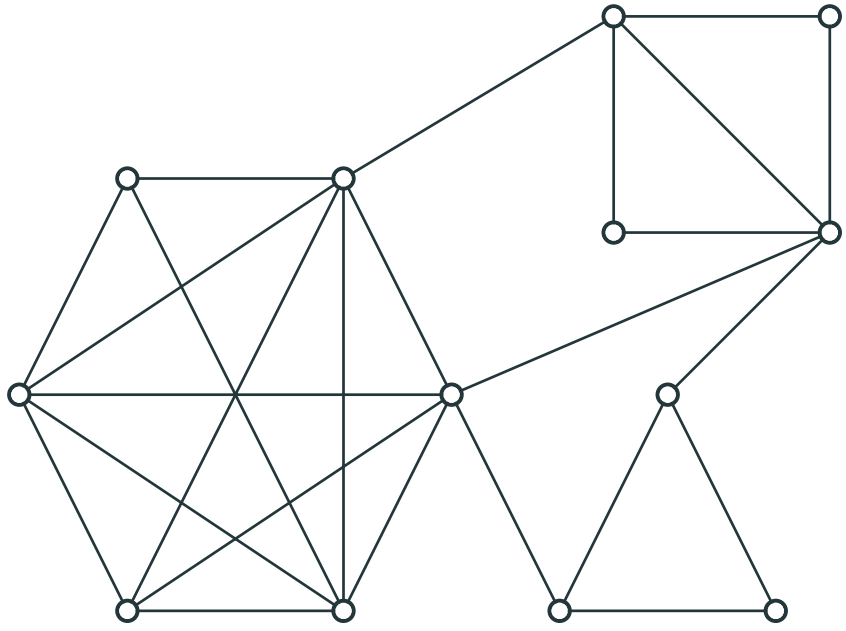
Zato rešujemo naslednji

Problem poškodovanih klik:

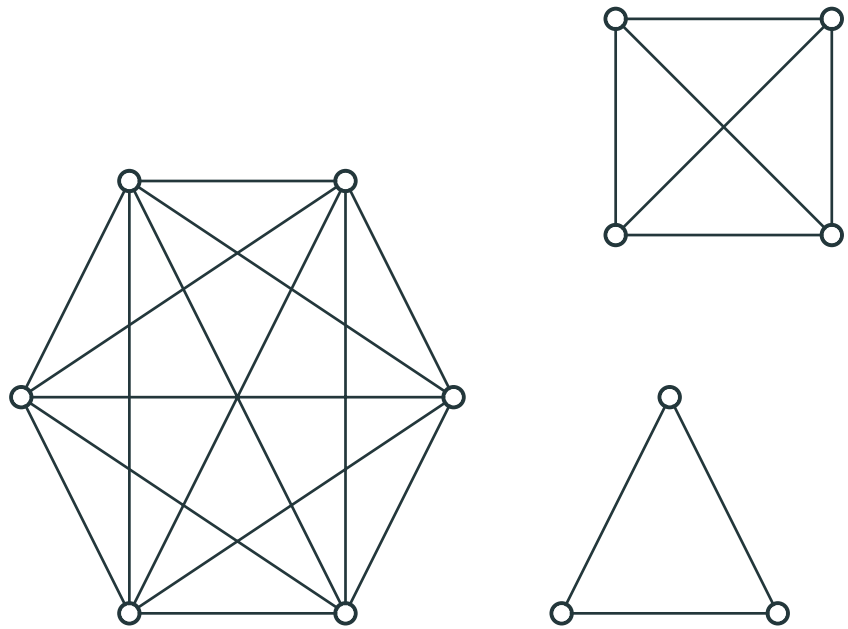
Dan je graf G. Določi najmanjše število povezav, ki jih moramo grafu dodati ali odvzeti, da dobimo klični graf.

Problem je NP-težek. V uporabi so številne hevristike.

Zgled eksperimentalno pridobljenega grafa:



Danemu grafu “najbližji” klični graf (iz katerega lahko rekonstruiramo pripadajoče gručenje):



Iz danega grafa je dovolj izbrisati 4 obstoječe povezave in mu dodati 3 nove.

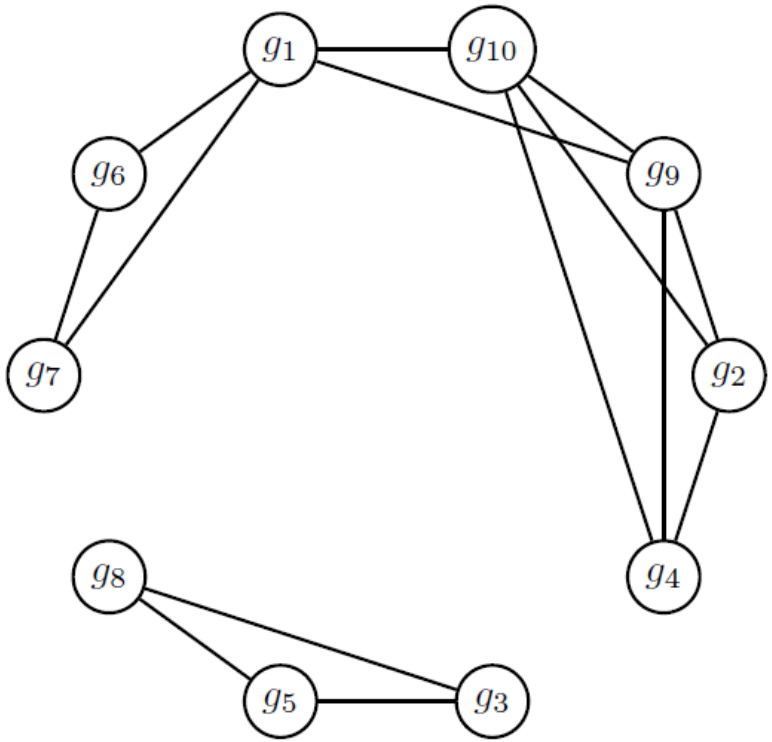
Še en zgled: matrika razdalj, razdaljni graf ($\theta = 7$) in klični graf

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	**2.3**	**5.1**	10.2	**6.1**	**7.0**
g_2	8.1	0.0	12.0	**0.9**	12.0	9.5	10.1	12.8	**2.0**	**1.0**
g_3	9.2	12.0	0.0	11.2	**0.7**	11.1	8.1	**1.1**	10.5	11.5
g_4	7.7	**0.9**	11.2	0.0	11.2	9.2	9.5	12.0	**1.6**	**1.1**
g_5	9.3	12.0	**0.7**	11.2	0.0	11.2	8.5	**1.0**	10.6	11.6
g_6	**2.3**	9.5	11.1	9.2	11.2	0.0	**5.6**	12.1	7.7	8.5
g_7	**5.1**	10.1	8.1	9.5	8.5	**5.6**	0.0	9.1	8.3	9.3
g_8	10.2	12.8	**1.1**	12.0	**1.0**	12.1	9.1	0.0	11.4	12.4
g_9	**6.1**	**2.0**	10.5	**1.6**	10.6	7.7	8.3	11.4	0.0	**1.1**
g_{10}	**7.0**	**1.0**	11.5	**1.1**	11.6	8.5	9.3	12.4	**1.1**	0.0

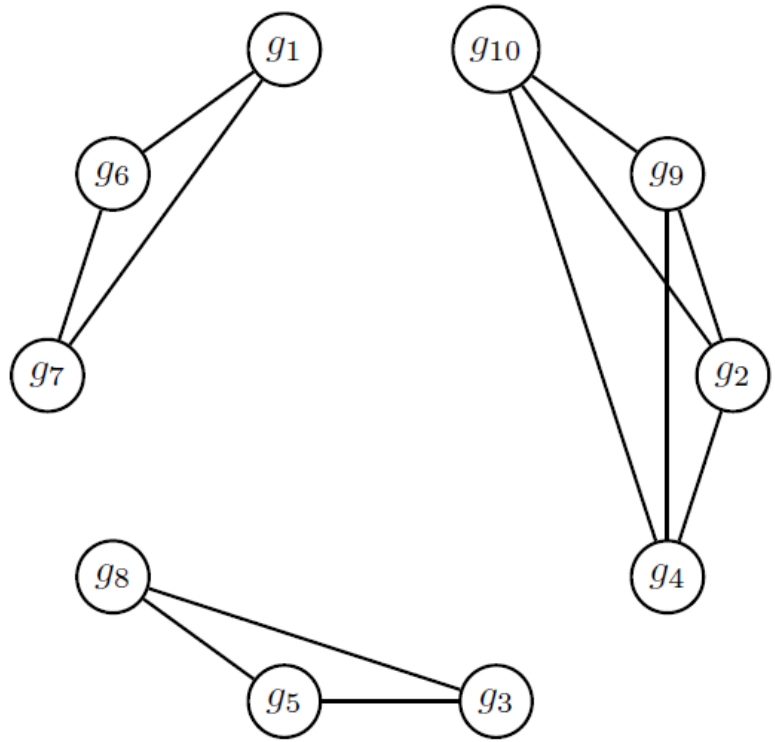
(a) Distance matrix, **d** (distances shorter than 7 are shown in bold).

Vir slike: Jones-Pevzner, Introduction to Bioinformatics Algorithms, str. 351

Še en zgled: matrika razdalj, razdaljni graf ($\theta = 7$) in klični graf



(b) Distance graph for $\theta = 7$.



(c) Clique graph.

Vir slike: Jones-Pevzner, Introduction to Bioinformatics Algorithms, str. 351

Hevristika CAST

Oglejmo si še hevristiko CAST za problem gručenja na osnovi razdaljnega grafa.

(CAST = Cluster Affinity Search Technique)

razdaljo med genom i in gručo C definiramo kot povprečno razdaljo do vseh točk v gruči:

$$d(i, C) = \frac{\sum_{j \in C} d(i, j)}{|C|} .$$

Naj bo $\theta > 0$ vnaprej izbrana konstanta.

Za dano gručo C pravimo, da je gen i gruči C **bližnji gen**, če je $d(i, C) < \theta$, sicer pa je **oddaljen**.

Algoritem CAST izračuna gručenje dane množice genov S glede na razdaljni graf G in vrednost konstante θ.

Iterativno izračuna particijo P množice S, tako da na vsakem koraku poišče gručo C, za katero velja, da

noben gen $i \notin C$ ni gruči C bližnji gen in

noben gen $i \in C$ ni gruči C oddaljen.

(tj. gruča C vsebuje natanko vse gruči C bližnje gene).

Spomnimo se:

stopnja točke x grafa G = število povezav, ki vsebujejo točko x

Metoda CAST(G, θ)*:*

$S \leftarrow$ množica točk razdaljnega grafa, $P \leftarrow \emptyset$

while $S \neq \emptyset$

$v \leftarrow$ točka največje stopnje v razdaljnem grafu

$C \leftarrow \{v\}$

while obstaja bližnji gen $i \notin C$ ali oddaljeni gen $i \in C$

poišči najbližji gen $i \notin C$ in ga dodaj C

poišči najbolj oddaljen gen $i \in C$ in ga odstrani iz C

end while

dodaj gručo C particiji P

$S \leftarrow S \setminus C$

Odstrani točke gruče C iz razdaljnega grafa G

end while

return P

Kljub temu da je CAST hevristika brez garancije o kvaliteti rešitve (in celo brez zagotovila o konvergenci),

na podatkih, pridobljenih iz meritev izražanja genov, deluje zelo dobro.