UNIVERZA V LJUBLJANI
BIOTEHNIŠKA FAKULTETA

Uroš SLANA

# ALGORITEM ZA ISKANJE MAKSIMALNE KLIKE PRI PROBLEMU GRUPIRANJA SOSESK ZA *de novo* DOLOČANJE NUKLEOTIDNEGA ZAPOREDJA

MAGISTRSKO DELO

Ljubljana, 2016

UNIVERZA V LJUBLJANI
BIOTEHNIŠKA FAKULTETA

Uroš SLANA

# ALGORITEM ZA ISKANJE MAKSIMALNE KLIKE PRI PROBLEMU GRUPIRANJA SOSESK ZA *de novo* DOLOČANJE NUKLEOTIDNEGA ZAPOREDJA

MAGISTRSKO DELO

# ALGORITHM FOR FINDING THE MAXIMUM CLIQUE ON THE PROBLEM OF CONTIGS CLUSTERING FOR *de novo* DETERMINATION OF THE NUCLEOTIDE SEQUENCE

M.SC. THESIS

Ljubljana, 2016

Na podlagi Statuta Univerze v Ljubljani ter po sklepu Senata Biotehniške fakultete z dne 2.11.2015 je bilo potrjeno, da kandidat izpolnjuje pogoje za magistrski Podiplomski študij bioloških in biotehniških znanosti ter opravljanje magisterija znanosti s področja genetike. Za mentorja je bil imenovan doc. dr. Tomaž Curk.

Raziskava je bila v celoti opravljena v Laboratoriju za bioinformatiko, Fakultete za računalništvo in informatiko, Univerze v Ljubljani in na Katedri za genetiko, biotehnologijo, statistiko in žlahtnjenje rastlin, Biotehniške fakultete, Univerze v Ljubljani.

V komisijo za oceno in zagovor so bili imenovani:

| | |
|---|---|
| Predsednik: | prof. dr. Peter DOVČ,<br>Univerza v Ljubljani, Biotehniška fakulteta, oddelek za zootehniko |
| Član: | prof. dr. Jernej JAKŠE,<br>Univerza v Ljubljani, Biotehniška fakulteta, oddelek za agronomijo |
| Član: | prof. dr. Janez DEMŠAR,<br>Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, oddelek za bioinformatiko |

Datum zagovora: 22.4.2016

Podpisani izjavljam, da je naloga rezultat lastnega raziskovalnega dela. Izjavljam, da je elektronski izvod identičen tiskanemu. Na univerzo neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravici shranitve avtorskega dela v elektronski obliki in reproduciranja ter pravico omogočanja javnega dostopa do avtorskega dela na svetovnem spletu preko Digitalne knjižnice Biotehniške fakultete.

Uroš SLANA

# KLJUČNA DOKUMENTACIJSKA INFORMACIJA

ŠD Md
DK UDK 575:004(043)=163.6
KG algoritmi/maksimalna klika/sekvenciranje/grupiranje/soseske/teorija grafov/nukleotidno zaporedje
AV SLANA, Uroš, prof. šp. vzg.
SA CURK, Tomaž (mentor)
KZ SI-1000 Ljubljana, Jamnikarjeva 101
ZA Univerza v Ljubljani, Biotehniška fakulteta, Podiplomski študij bioloških in biotehniških znanosti, področje genetika
LI 2016
IN ALGORITEM ZA ISKANJE MAKSIMALNE KLIKE PRI PROBLEMU GRUPIRANJA SOSESK ZA *de novo* DOLOČANJE NUKLEOTIDNEGA ZAPOREDJA
TD Magistrsko delo
OP XI, 62 str., 15 pregl., 30 sl., 43 vir.
IJ sl
JI sl/en
AI V magistrski nalogi odgovarjamo na vprašanje, ali lahko z iskanjem simetrij v grafu izboljšamo trenutne algoritme za iskanje maksimalne klike in, če z dekompozicijo grafa z maksimalno kliko lahko grupiramo soseske, ki pripadajo istemu transkriptu. Novo razvit algoritem Gtc, ki za iskanje maksimalne klike uporablja simetrije v grafu, smo primerjali z algoritmom Mcqd. Izkazalo se je, da ima večina testiranih grafov takšne simetrije, da njihovo iskanje zahteva le dodaten čas pri iskanju maksimalne klike. Z dekompozicijo grafa z maksimalno kliko smo poskušali med seboj združiti soseske, na grafu, ki je bil zgrajen gleda na podobnost med soseskami in glede na ujemanja z znanimi referenčnimi sekvencami (BLASTx). Soseske, združene v maksimalni kliki, smo poravnali s programom Clustal Omega. Glede na prekrivanje med soseskami, ki pripadajo maksimalni kliki, menimo, da je na grafu zgrajenem glede na podobnost med soseskami mogoče grupirati soseske, ki pripadajo istemu transkriptu, če so v grafu med seboj povezane le soseske z vsaj 80 % podobnostjo. Glede na lastnosti teh grafov, menimo, da bi problem grupiranja sosesk na teh grafih, lahko hitreje rešili z iskanjem povezanih komponent.

# KEYWORDS DOCUMENTATION

ND Md
DC UDC 575:004(043)=163.6
CX algorithm/maximal clique/cluster/contig/sequencing/graph theory/ nucleotide sequence
AV SLANA, Uroš, prof. šp. vzg.
SA CURK, Tomaž (supervisor)
PP SI-1000 Ljubljana, Jamnikarjeva 101
PB University of Ljubljana, Biotechnical Faculty, Postgraduate Study of Biological and Biotechnical Sciences, Field: Genetics
PY 2016
TY ALGORITHM FOR FINDING THE MAXIMUM CLIQUE ON THE PROBLEM OF CONTIGS CLUSTERING FOR *de novo* DETERMINATION OF THE NUCLEOTIDE SEQUENCE
DT M.Sc. Thesis
NO XI, 62 p., 15 tab., 30 fig., 43 ref.
LA sl
Al sl/en
AB The purpose of this thesis was to find whether taking graph symmetries into account would make an improvement of current algorithms for finding a maximum clique. Another goal of the thesis was to determine if contigs of the same transcriptome could be clustered with graph decomposition based on the maximum clique method. The algorithm Mcqd and algorithm Gtc were compared from the computational standpoint. Gtc is a new algorithm taking graph symmetries into account when finding a maximum clique. It has been shown that, generally speaking, on four different types of graphs finding symmetries only takes extra computational time thus causing unnecessary overhead. With graph decomposition based on the maximum clique method, we clustered contigs on graph constructed from distance matrix and relation graph based on the BLASTx results. Contigs which were clustered into a maximum clique were aligned with the program Clustal omega. With respect to the visualization of alignments, graph decomposition into maximum clique could be a useful method for clustering contigs belonging to the same transcriptome on similarity graph if the cutoff point is $\leq 0.2$. This corresponds to at least 80 % of similarity between contigs. According to properties of the similarity graphs, the clustering problem could be solved more easily by simply finding all connected components.

# KAZALO VSEBINE

# KAZALO PREGLEDNIC

# KAZALO SLIK

# OKRAJŠAVE IN SIMBOLI

| | |
|---|---|
| $G$ | graf |
| $V$ | vozlišča |
| $E$ | povezave |
| $\|V\|$ | število vozlišč |
| $\|E\|$ | število povezav |
| $d(G)$ | gostota grafa |
| $\delta(G)$ | najmanjša stopnja vozlišča grafa $G$ |
| $\Delta(G)$ | največja stopnja vozlišča grafa $G$ |
| $c$ | klika |
| $c_{max}$ | maksimalna klika |
| komp. | komponenta |
| Št. p. komp. | število povezanih komponent |
| komp. max | največja komponenta |

# 1 UVOD

Stephens in sod. (2015) predvidevajo, da bo leta 2025 zbiranje, skladiščenje, distribucija in analiza podatkov na področju genomike verjetno bolj zahtevno, kot pa na področju astronomije, Twitterja in YouTuba. Slednja tri področja so ena izmed največjih proizvajalcev velikih podatkov. Količina podatkov iz sekvenciranja se skoraj podvoji vsakih sedem mesecev. Število podatkov sekvenciranja, ki jih hrani *Sequence Read Archive*, ki ga vzdržujeta Nacionalni inšitut za zdravje Združenih držav Amerike in Nacionalni center za biotehnološke informacije, je približno $3,2$ petabaz ($1x10^{15}$ baz, 4 *baze* = 1 *bajt*). Od tega je približno 32000 mikrobnih genomov, 5000 genomov rastlin in živali ter 250000 individualnih človeških genomov. To seveda niso vsi podatki, ki so v tem trenutku na svetu. Ob takšni rasti podatkov naj bi leta leta 2025 presegli količino 1 zetta baze sekvenciranih podatkov (1 zetta = $10^{21}$). Trenutno najbolj zahtevne analize so na področju iskanja razlik med genomi in poravnave dveh genomov. Poravnava med človeškim in mišjim genomom potrebuje približno 100 CPU ur računanja (Stephens in sod., 2015).

Pri analizi podatkov si velikokrat pomagajo tako, da problem modelirajo s pomočjo matematične strukture grafov. V bioinformatiki s pomočjo grafov določajo sekundarno strukturo RNA (Ji in sod., 2004), iščejo povezave med geni in biološkimi pojavi (Matsunaga in sod., 2009; Rokhlenko in sod., 2007) in analizirajo proteinske strukture (Dukka in sod., 2006). Vsem omenjenim problemom je skupno iskanje maksimalne klike v grafu, za iskanje katere je bilo razvitih mnogo algoritmov (Carraghan in Pardalos, 1990; Östergård, 2002; Tomita in Seki, 2003; Konc in Janežič, 2007). Kljub temu je potreba po hitrih algoritmih za reševanje omenjenega problema v porastu. Iskanje maksimalne klike je zahteven problem (Karp, 1972), ob tem pa velikost raziskovalnih grafov narašča. Splošen algoritem, ki se uporablja za iskanje maksimalne klike, deluje po strategiji razvejaj in omeji. Izboljšati ga poskušajo z določanjem boljših kriterijev za omejevanje (Tomita in sod., 2010) oziroma s paralelnim računanjem (Depolli in sod., 2013). Za omejevanje se je zelo dobro izkazalo barvanje vozlišč (Konc in Janežič, 2007).

Kot je bilo že omenjeno, je v genomiki določevanje zaporedja v porastu. Razvile so se nove metode sekvenciranja, ki jih uvrščamo v metode naslednje generacije, kamor sodi

tudi metoda RNA-seq, s katero določamo zaporedja izraženih regij genoma. Glavna značilnost te metode je predvsem večje število krajših odčitkov (Davidson in Oshlack, 2014; Wang in sod., 2009), ki se jih nato združi v soseske. Ker je le-teh več kot genov oziroma transkriptov, poskušajo programi z različnimi metodami grupirati tiste soseske, ki pripadajo istemu genu oziroma transkriptu (Davidson in Oshlack, 2014; Grabherr in sod., 2011).

Glede na problematiko je namen in cilj naloge ugotoviti, ali je z iskanjem maksimalne klike na grafu mogoče grupirati soseske, ki pripadajo istemu transkriptu. Predstavili bomo nov algoritem in program, ki za iskanje maksimalne klike upošteva simetrije na grafu. Ugotoviti želimo, ali bo razvit algoritem učinkovit tudi na področju bioinformatike.

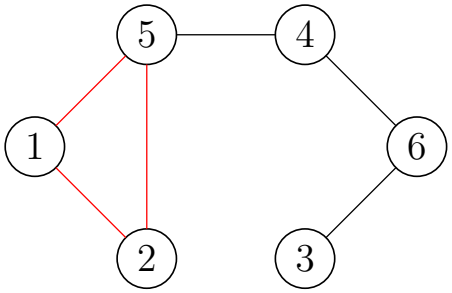
Glede na namen in cilje smo postavili naslednje raziskovalne hipoteze:

- Z upoštevanjem simetrije grafov se da izboljšati in pohitriti obstoječe algoritme za iskanje maksimalne klike.
- Izboljšan algoritem omogoča hitrejše iskanje maksimalne klike v grafu, ki je zgrajen glede na podobnost med soseskami in glede na ujemanja z znanimi referenčnimi sekvencami (BLASTx).
- Z iskanjem maksimalne klike je mogoče grupirati soseske, ki pripadajo istemu transkriptu.

# 2 PREGLED OBJAV

## 2.1 TEORIJA GRAFOV

Teorija grafov preučuje matematične strukture za modeliranje odnosov med pari objektov. Graf ($G$) je definiran kot par množice vozlišč $V(G)$, med katerimi so povezave $E(G)$. Pravimo, da sta vozlišči $v, u \in V(G)$ sosednji, če $(u, v) \in E(G)$ - na kratko pišemo $u \sim_G v$.




Slika 1: Graf s šestimi vozlišči $V(G) = \{1, 2, 3, 4, 5, 6\}$ in šestimi povezavami $E(G) = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 6\}, \{4, 5\}, \{4, 6\}\}$.
Figure 1: A graph with six vertices $V(G) = \{1, 2, 3, 4, 5, 6\}$ and six edges $E(G) = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 6\}, \{4, 5\}, \{4, 6\}\}$.

V primeru, da je smer povezave pomembna, uporabljamo pojem usmerjenega grafa. Pravimo, da je $u$ soseden z $v$, če je $(u, v) \in E(G)$ in pišemo $u \to v$.

Za prvi problem, ki je bil modeliran s pomočjo grafov, štejemo prečkanje mostov v mestu Königsberg (Wallis, 2010), danes pa se teorija grafov uporablja skoraj na vseh področjih. Z grafi analizirajo socialna omrežja, izdelujejo navigacijske pripomočke, sestavljajo računalniška omrežja itd. V nadaljevanju bomo opisali nekaj osnovnih pojmov in definicij, ki se uporabljajo pri teoriji grafov in bodo prišli prav pri našem delu.

Grafi lahko vsebujejo tudi vzporedne povezave in zanke. Takšnim grafom pravimo *multigrafi*.

*Stopnja vozlišča* $d(x)$ je število sosedov vozlišča $x$. Najmanjšo stopnjo vozlišča grafa $G$ označimo s $\delta(G)$, največjo pa z $\Delta(G)$. Če je $d(x) = 0$, potem rečemo, da je vozlišče $x$ izolirano. *Regularen graf* je graf, kjer imajo vsa vozlišča enako stopnjo. Vsota vseh

stopenj vozlišč je enaka dvakratniku povezav:

$$\sum_{v \epsilon V(G)} d(v) = 2e\,. \tag{1}$$

Graf, kjer so vsa vozlišča sosednja, imenujemo *poln* graf. Iz enačbe 1 sledi, da ima poln graf:

$$\frac{|V(G)| \cdot (|V(G)| - 1)}{2}\,, \tag{2}$$

povezav.

Od tod se gostota danega grafa $G$ definira kot:

$$d(G) = \frac{2|E(G)|}{|V(G)| \cdot (|V(G))| - 1)}\,. \tag{3}$$

Graf, kjer so vse stopnje vozlišč enake 0, imenujemo *prazen* graf.

*Komplement* grafa $G$ je graf $\overline{G}$, ki ima isto množico vozlišč kot graf $G$. Dve vozlišči pa sta sosednji natanko tedaj, ko nista sosednji v $G$.

Sprehod v grafu G je zaporedje vozlišč in povezav grafa G:

$$x_0, a_1, x_2, a_2, ...., a_n, x_n$$

kjer je $a_i$ povezava med $x_i - 1$ in $x_i$ za vsak $i$. Sprehodu, kjer se nobeno vozlišče ne ponovi, pravimo *pot*. Pot je *zaprta*, če je $x_0 = x_n$. Zaprti poti pravimo *cikel*. Graf, ki ne vsebuje cikla, imenujemo *gozd*.

Graf je *povezan*, če obstaja pot med vsakim parom vozlišč. Graf, ki ni povezan, imenujemo *nepovezan* graf. Povedano drugače, nepovezan graf je mogoče razdeliti v dve podmnožici, med katerima ni povezave.

*Dvodelen graf* je graf, kjer je mogoče vozlišča razdeliti v dve podmnožici tako, da si vozlišča znotraj podmnožice niso sosednja.

Vsaki povezavi je mogoče prirediti število, ki ji pravimo *utež*. Le-ta lahko predsta-

vlja razdaljo, ceno ali kakšno drugo količino. Takšnim grafom pravimo *uteženi* grafi. Formalno je utež funkcija:

$$f : E(G) \to \mathbb{R}\,.$$

### 2.1.1 Klika

Graf $H$ je *podgraf* grafa $G$, če velja:

$$V(H) \subseteq V(G) \bigwedge E(H) \subseteq E(G)\,.$$

Graf $H$ je *induciran podgraf* grafa $G$, če za vsak $x, y \epsilon V(H)$ velja:

$$x \sim_G y \implies x \sim_H y\,.$$

Graf $H$ je vpet *podgraf*, če je:

$$V(H) = V(G)\,.$$

*Klika* v grafu $G$ je polni podgraf. Klika $H$ je *maksimalna* v grafu $G$, če v grafu $G$ ne obstaja večja klika.

Na sliki 1 je maksimalna klika velikosti 3, ki jo tvorijo vozlišča $1, 2, 5$.

### 2.1.2 Barvanje vozlišč

Pri barvanju vozlišč razdelimo vozlišča v podmnožice tako, da si vozlišča, ki pripadajo isti podmnožici, med seboj niso sosednja. Različne podmnožice ustrezajo različnim barvam vozlišč. Pravilno pobarvati graf je lahka naloga, zelo težko pa je najti *kromatično število* $\chi(G)$, ki je minimalno število barv, s katerimi lahko pravilno pobarvamo graf. Z eno barvo je mogoče pobarvati grafe, ki ne vsebujejo povezav, z dvema pa je mogoče

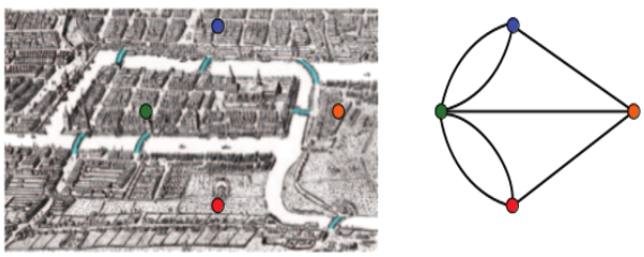
pobarvati vse dvodelne grafe. Za določanje kromatičnega števila ne poznamo učinkovitega algoritma. Za barvanje vozlišč se zato večinoma uporablja požrešen algoritem, ki deluje tako, da vsakemu vozlišču iz urejene množice določi prvo možno minimalno barvo. V kolikor nobena barva ni prosta, dodeli vozlišču novo barvo. Število barv, ki jih bo takšen algoritem porabil, je odvisno od vrste grafa in vrstnega reda vozlišč, ki so ponavadi urejena padajoče glede na stopnjo. Opisani algoritem v splošnem vrne dokaj optimalno barvanje, vendar obstajajo izrojeni primeri. Takšni so npr. neusmerjeni grafi z dvema množicama vozlišč $U$ in $V$ in povezavo med $u_i$ in $v_j$ kjer $i \neq j$. Za barvanje takšnega grafa bomo porabili $n/2$ barv, če bo vrstni red vozlišč takšen, kot ga prikazuje slika 2 desno. Lahko pa ga pobarvamo le z dvema barvama, če je vrstni red takšen, kot ga prikazuje slika 2 levo.


Slika 2: Vrstni red barvanja vozlišč določa kvaliteto rešitve požrešnega algoritma za barvanje grafov: optimalna rešitev (levo), zelo slaba rešitev (desno).
Figure 2: Graph vertex ordering, for which the greedy algorithm would find an optimal solution (left), and bad solution (right).

### 2.1.3 Eulerjeva pot

Eulerjeva pot je poimenovana po matematiku Leonhardu Eulerju. V 17. stoletju je reka Pregolya mesto Königsberg delila na štiri dele (slika 3), ki so bili med seboj povezani s sedmimi mostovi. Zanimalo jih je, ali obstaja obhod, kjer bi bilo možno le enkrat prečkati vsak most. Euler je problem modeliral s pomočjo grafa. Vsak del mesta je predstavil kot vozlišče, most pa kot povezavo. Dokazal je, da takšen obhod ni mogoč.

Eulerjeva pot je v teoriji grafov sprehod po grafu $G$, kjer vsako povezavo obiščemo



Slika 3: Mostovi v mestu Königsberg. Graf na desni prikazuje problem prečkanja mostov v mestu Königsberg (Compeau in sod., 2011).
Figure 3: Bridges of Königsberg. One the right side, the bridge-crossing problem is illustrated by a graph (Compeau in sod., 2011).

natanko enkrat. Eulerjev cikel je Eulerjeva pot, če je začetno in končno vozlišče enako. Graf ima Eulerjevo pot natanko takrat, ko:

- je povezan,
- imajo vse točke sodo stopnjo,
- ima 0 ali 2 dve vozlišči z liho stopnjo in začnemo sprehod v enem ter končamo v drugem vozlišču z liho stopnjo.

Za iskanje Eulerjevih ciklov se v praksi uporablja Fleurjev algoritem.

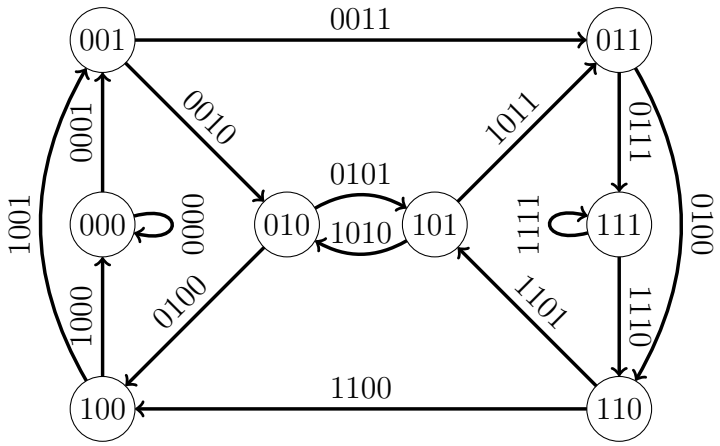
### 2.1.4 Hamiltonski cikel

V nasprotju z Eulerjevo potjo, kjer lahko vsako vozlišče obiščemo večkrat, nas pri Hamiltonskem ciklu zanima, ali obstaja cikel, kjer vsako vozlišče obiščemo natanko enkrat. Prav tako ne obstaja splošno pravilo, ki bi določalo, ali je graf Hamiltonski (Wallis, 2010).

### 2.1.5 De Bruijnov graf

Naj bo dana neka končna množica znakov $S = \{s_1, ..., s_n\}$. De Bruijnov graf $G(S, k)$ je usmerjen graf, katerega vozlišča so vse $k$-terice množice $S$. Iz vozlišča $u$ je usmerjena povezava v vozlišče $v$ natanko tedaj, ko je zapona vozlišča $u$ velikosti $k - 1$ enaka priponi vozlišča $v$.

Nicolas de Bruijn je s pomočjo grafa rešil problem "superstringa", kjer se iz besed velikosti $k$, sestavi besedo, ki bo vsebovala vse besede natanko enkrat. To je naredil tako, da je zgradil de Bruijnov graf in poiskal Eulerjevo pot (slika 4) (Compeau in sod., 2011).




Slika 4: De Bruijnov graf iz dveh znakov $S = \{0, 1\}$ in $k = 3$. "Superstring" je *0000110010111101* (povzeto po Compeau in sod. (2011)).
Figure 4: De Bruijn graph of two symbols $S = \{0, 1\}$ and $k = 3$. Formed "Superstring" is 0000110010111101 (reprinted from Compeau in sod. (2011)).

### 2.1.6 Simetrije v grafu

Simetrije pogosto modeliramo z grupami. Grupa je matematični objekt, definiran kot kot par neprazne množice in operacije na množici (enačba 4) (Godsil in Royle, 2001).

$$(G, *) \tag{4}$$

Operacija mora zadoščati štirim pogojem:

1. Zaprtost

$$g, h \, \epsilon \, G \rightarrow g * h \epsilon G \tag{5}$$

2. Asociativnost

$$g, h, i \, \epsilon \, G \rightarrow g * (h * i) = (g * h) * i \tag{6}$$

3. Obstoj enote

$$\exists\ e\ \epsilon\ G\ \forall g\ \epsilon\ G \rightarrow e * g = g \tag{7}$$

4. Obstoj inverza

$$\forall\ g\ \epsilon\ G\ \exists\ h\ \epsilon\ G \rightarrow g * h = e \tag{8}$$

Kot primer grupe lahko vzamemo množico celih števil $Z$ in operacijo seštevanja. Brez težav se prepričamo, da je množica celih števil grupa. Enota je število 0 in vsako celo število ima svoj inverz. Drugi primer predstavlja množica vseh permutacij $\{1 \dots n\}$ z operacijo kompozicij permutacij. Slednji grupi rečemo simetrična grupa in jo označimo s $S_n$. Enačba 9 predstavlja dve permutaciji.

$$g = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 3 & 1 & 4 \end{pmatrix} h = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 2 & 5 & 1 \end{pmatrix} \tag{9}$$

Kompozicija $z = g \circ h$ je permutacija, ki slika $x\ \epsilon\ \{1...n\}$ v element $g(h(x))$.

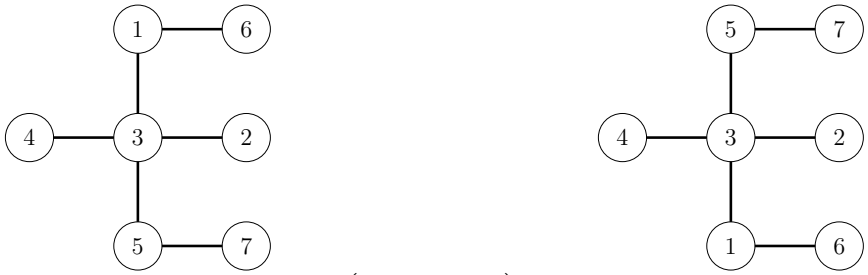
$$z = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 4 & 2 \end{pmatrix} \tag{10}$$

Naj bo $H$ podmnožica $G$. Če za $H$ veljajo prej omenjeni pogoji, potem pravimo, da je $H$ podgrupa $G$. Za grupo celih števil in operacije seštevanja je primer podgrupe množica sodih števil.

Za iskanje simetrij na grafih so pomembne permutacijske grupe, ki jih definiramo kot podgrupe simetrične grupe. Za razumevanje pojma simetrije grafa bomo vpeljali nov pojem, *avtomorfizem* grafa. Naj bo $G$ graf. Naj bo $f$ permutacija $V(G)$. Če za vsak par vozlišč $u, v$ velja:

$$u \sim v \leftrightarrow f(u) \sim f(v), \tag{11}$$

potem pravimo, da je $f$ avtomorfizem grafa $G$. Na sliki 5 je primer avtomorfizma grafa $G$, ki ga ponazorimo s permutacijo.




$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 2 & 3 & 4 & 1 & 7 & 6 \end{pmatrix}$$

Slika 5: Avtomorfizem grafa.
Figure 5: Graph automorphism.

Množico vseh avtomorfizmov danega grafa označimo z $Aut(G)$.

$$Aut(G) = \{\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 4 & 3 & 2 & 5 & 6 & 7 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 2 & 3 & 4 & 1 & 7 & 6 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 4 & 3 & 2 & 1 & 7 & 6 \end{pmatrix}\} \quad (12)$$

*Orbita* vozlišča $v - O(v)$ je maksimalna množica vozlišč, tako da za vsak $u \epsilon O(v)$ obstaja avtomorfizem $f$, tako da:

$$f(u) = v. \quad (13)$$

Množica orbit za vsa vozlišča tvori razbitje $V(G)$, ki ga označimo z $O(G)$. To pomeni, da za $A, B \epsilon O(G)$ velja $A = B$ ali $A \bigcap B = 0$.

Pred zaključkom poglavja bomo definirali še pojem *generatorji grupe*, ki ga bomo potrebovali pri algoritmu Gtc (slika 15, na strani 26). Množici $S \subseteq G$ pravimo generator grupe $(G, *)$, če lahko vsak element iz $G$ zapišemo kot nek produkt elementov iz $S$.

## 2.2 SESTAVLJANJE NUKLEOTIDNEGA ZAPOREDJA

Moderna doba sekvenciranja DNA se je začela leta 1977 s kemično metodo Maxama in Gilberta ter Sangerjevo dideoksi metodo (Hutchison, 2007), ki je v uporabi že skoraj štirideset let. Pri Sangerjevi metodi pripravijo DNA za sekvenciranje na dva načina. Za *de novo* sekvencioniranje DNA naključno razrežejo, ga vstavijo v plazmid in ga v

bakteriji E. coli namnožijo. Za kopiranje točno določenih delov DNA uporabijo verižno reakcijo s polimerazo (PCRI). Namnoženo DNA razdelijo v štiri različne raztopine s fluorescentno označenimi dideoksinukleotidi, DNA-polimerazo in deoksinukleotidi. Zaradi prisotnosti vseh štirih deokisinukleotidov začne DNA-polimeraza graditi verigo DNA, dokler ne pride do vezave dideoksinukleotida. Rezultat so različno dolge verige DNA, ki so, glede na končni nukleotid, označene z ustreznim fluorom. S kapilarno elektroforezo z visoko ločljivostjo nato odčitajo zaporedje nukleotidov (Shendure in Ji, 2008).

Kljub temu, da je Sangerjeva metoda učinkovita, sta njeni pomanjkljivosti predvsem cena in porabljen čas. Gel, kot ločitveni medij, ne omogoča vzporedne obdelave večje količine vzorcev, prav tako ne omogoča popolne avtomatizacije priprave vzorcev. Zato so se razvile nove metode določanja nukleotidnega zaporedja, ki jih imenujemo metode naslednje generacije (NGS). V splošnem lahko metode naslednje generacije delimo na štiri vrste:

- mikro elektroforetske metode,
- sekvenciranje s hibridizacijo,
- sekvenciranje v realnem času z opazovanjem posameznih molekul,
- ciklično mrežno sekvenciranje (Shendure in Ji, 2008).

Najbolj se uporablja ciklično mrežno sekvenciranje, kjer DNA fragmentirajo, sledi *in vitro* ligacija skupnih DNA adapterjev in namnožitev DNA fragmentov. Sekvenciranje poteka v cikličnih encimskih reakcijah, kjer pride do podaljševanja DNA s fluorescentno označenimi nukleotidi in zajemanja slike celotne mreže. Prednosti teh metod so predvsem v nižji ceni, hitrejšemu določanju zaporedja, *in vitro* namnožitvi klonov in hkratnemu določanju večjega števila zaporedij. Slabosti pa sta predvsem krajša dolžina odčitkov in manj natančna določitev nukleotidnega zaporedja (Shendure in Ji, 2008).

V zadnjih letih so se v genetiki veliko ukvarjali z določanjem nukleotidnega zaporedja DNA. Uspelo jim je določiti referenčne genome za različne organizme. V zadnjem času pa se veliko pozornosti usmerja v sekvenciranje RNA, saj ima le-to velik pomen pri preučevanju izražanja genov, kar je omogočilo razvoj novih metod za *de novo* določanje

nukleotidnega zaporedja. Metoda RNA-seq zato spada v naslednjo generacijo določanja nukleotidnega zaporedja, saj ga je mogoče določiti brez referenčnega genoma (Davidson in Oshlack, 2014; Wang in sod., 2009). Prednosti te metode so:

- natančno določanje lokacije transkriptnih območij na natančnost enega nukleotida,
- analiza kompleksnih transkriptomov, saj glede na velikost odčitka določi povezavo med eksoni,
- odsotnost šuma, saj lahko direktno določi cDNA (Wang in sod., 2009).

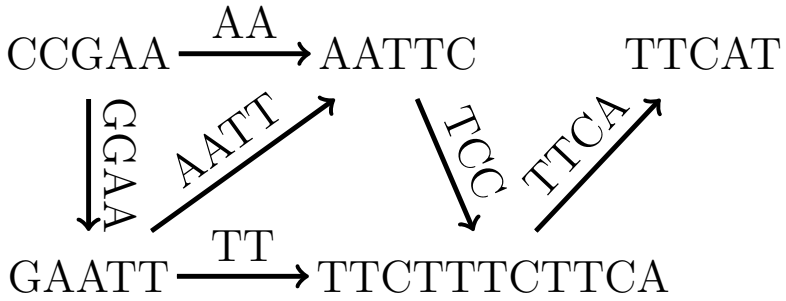
Končni produkt vseh metod naslednje generacije sekvenciranje DNA/RNA so kratki odčitki, ki jih je potrebno združiti. Za združevanje odčitkov se večinoma uporabljata dva algoritma: "overlap-layout-consensus" (OLC) in združevanje z de Bruijnovim grafom (Li in sod., 2012). Algoritem OLC zgradi prekrivajoč usmerjen graf, kjer so odčitki vozlišča, povezave pa so prekrivajoči deli med odčitki. Pri de Bruijnovem grafu vozlišča na grafu predstavljajo deli odčitkov velikosti $k$, povezave pa so med vozlišči, kjer je pripona velikosti $k-1$ enaka predponi velikosti $k-1$.

Osnovno delovanje obeh algoritmov lahko predstavimo z modelom Lander–Waterman, ki je prvi matematični model za sestavljanje nukleotidnega zaporedja in nam pri sekvenciranju omogoča izračun števila sosesk (Li in sod., 2012). Iz velikosti genoma ($g$), dolžine odčitkov ($L$), števila odčitkov ($N$), globine sekvenciranja ($c$) in velikosti prekrivanja, ki je potrebno, da obravnavamo odčitka kot prekrivajoča ($T$), lahko izračunamo število sosesk $s$ (enačba 14) (Lander in Waterman, 1988).

$$\begin{aligned} \theta &= \frac{T}{L} \\ \sigma &= 1 - \theta \\ c &= (L \cdot N)/g \\ s &= N \cdot e^{-c \cdot \theta} \end{aligned} \tag{14}$$

Kot že ime algoritma OLC pove, poteka združevanje odčitkov v treh stopnjah. V prvi stopnji algoritem poišče odčitke, ki se med seboj prekrivajo. Programi paroma primerjajo odčitke tako, da med seboj primerjajo predpone in pripone in obratno.

Kriteriji za prekrivanje odčitkov se med programi razlikujejo. Pri programu Celera, ki je eden izmed prvih takšnih programov, mora biti razlika pri prekrivanju manj kot 6 % in mora vključevati vsaj 40 nukleotidov (Myers in sod., 2000). Večina programov za ugotavljanje prekrivanja uporablja algoritem "seed-and-extend", ki ga uporabljajo pri BLAST-u. Algoritem temelji na uporabi zgoščenih tabel, v katere se kot ključ shrani podnize odčitka določene velikosti, kot vrednost pa mesto začetka podnizov. V kolikor algoritem najde prekrivajoča se podniza, sledi primerjanje/poravnava z dinamičnim programiranjem, kjer so dovoljena odstopanja v prekrivanju. Iz prekrivajočih parov se zgradi usmerjen graf, kjer vsak odčitek predstavlja vozlišče, povezava pa je prekrivajoči del med odčitkoma (slika 6) (Myers in sod., 2000; Miller in sod., 2010; Li in Homer, 2010). Glede na Lander–Watermanov model je prekrivajoči del pri algoritmu OLC definiran enako kot spremenljivka $T$ (Li in sod., 2012).

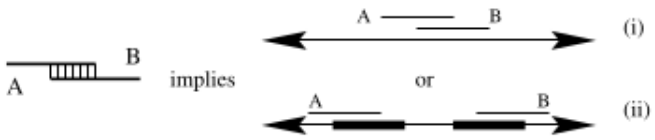



Slika 6: Prekrivajoči graf. Vsako vozlišče je odčitek, povezava pa prekrivajoči del med odčitkoma.
Figure 6: Overlap graph. Each vertex represents a read, while the edge is an overlap between two reads.

.

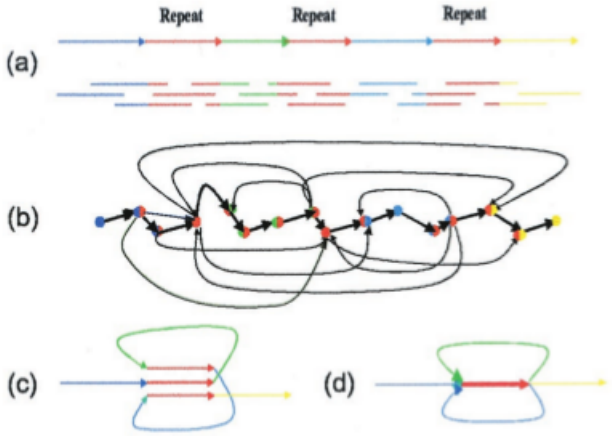
Za sestavo genoma v prekrivajočem grafu je potrebno poiskati Hamiltonsko pot. Problem tega pristopa je, da ne obstaja učinkoviti algoritem za iskanje takšne poti, hkrati pa je možnih več takšnih poti. Izpostaviti je potrebno, da povezave v grafu tvorita dve vrsti prekrivanj, ki jih lahko poimenujemo kot resnična prekrivanja (slika 7 (i)) in kot prekrivanja zaradi ponovitev (slika 7 (ii)). Značilnost ponavljajočih sekvenc je velika pogostost. Odkriva se jih s pomočjo statističnih metod (Myers in sod., 2000).

V fazi postavitve se najprej združijo tisti odčitki, katerih prekrivanje je nedvoumno. Tako nastanejo soseske, ki jih pri programu Celera imenujejo "unitigi", ki se nato povežejo v večje soseske (Myers in sod., 2000).




Slika 7: Odčitka se lahko prekrivata, če izhajata iz istega dela genoma (i) ali, če izhajata iz ponavljajočih se delov (ii) (Myers in sod., 2000).
Figure 7: Reads may overlap, because they arised from the same portion of the genome (i), or because the overlapping portion is a part of a repeated sequence (Myers in sod., 2000).

V nasprotju z algoritmom OLC, ki v prekrivajočem grafu išče Hamiltonsko pot, pa algoritem združevanja z de Bruijnovim grafom išče Eulerjevo pot (Pevzner in sod., 2001a). Na sliki 8 je prikazan del DNA s tremi edinstvenim deli in tremi ponavljajočimi zaporedji, ki prikazuje sestavo obeh omenjenih grafov. Razvidno je, da je prekrivajoči graf zelo gost in predstavlja velik računski izziv za iskanje Hamiltonske poti. V nasprotju pa je De Bruijnov graf zelo enostaven in je iskanje Eulerjeve poti računsko enostavnejši problem. Pristop z de Bruijnovimi grafi je boljši, ker dela manjše število napak pri združevanju odčitkov kot programi PHRAP, CAP3 in TIGR, ki uporabljajo algoritem OLC (Pevzner in sod., 2001a).




Slika 8: (a) DNA s tremi ponavljajočimi deli; (b) prekrivajoči graf; (c in d) de Bruijnov graf (Pevzner in sod., 2001a).
Figure 8: (a) DNA sequence with triple repeate; (b) overlapping graph, (c and d) de Bruijn graph (Pevzner in sod., 2001a).

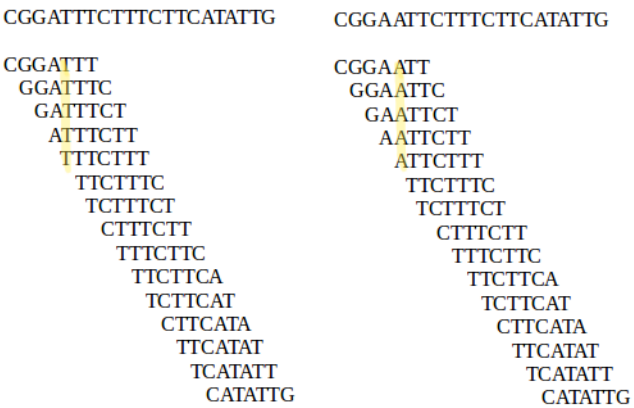
Za sestavo de Bruijinovega grafa je potrebno najprej odpraviti napake, ki so nastale pri sekvenciranju. Pevzner in sod. (2001b) za popravljanje napak odčitke razrežejo na

dele velikosti $l$ in jim, glede na pogostost njihovega pojavljanja, zamenjajo posamezen nukleotid. Na ta način zmanjšajo število delov.

Imamo množico odčitkov $S = \{s1, \ldots, sn\}$. $Sl$ je množica delov velikosti $l$ odčitkov $s_1, \ldots, s_n$ in $\overline{s_1}, \ldots, \overline{s_n}$, kjer je $\overline{s}$ obrnjen komplement. Glede na frekvenco pojavljanja delov v $Sl$, jih delimo na močne in šibke. Slednji se pojavljajo manj pogosto in ponavadi vsebujejo napako. Popravijo jih tako, da jim najdejo sosede, s katerimi se razlikuje samo v enem nukleotidu. Šibki deli, ki jih popravijo, morajo imeti naslednje lastnosti:

- pojavnost ($m$) šibkega dela ($a$) v odčitkih $m(a) < M$, kjer je $M$ v naprej določen prag,
- imajo samo enega soseda ($b$),
- $m(a) <= m(b)$.

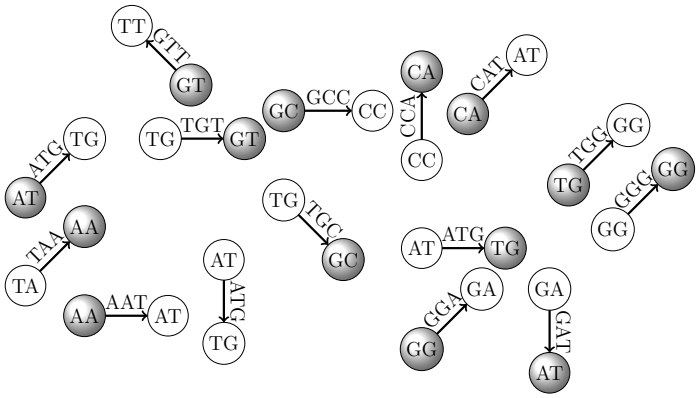
Na sliki 9 je primer napake na petem nukleotidu. Na levi strani je zaporedje brez napake, na desni pa z napako. Napako popravijo tako, da $A$ spremenijo v $T$. Pri tem se $Sl$ zmanjšanja za $2l$ oziroma za $2d$, kjer je $d$ razdalja od začetka odčitka do mesta napake in je $d < l$. Posamezen odčitek se lahko popravi le, če je zgornja meja manjša, kot je vnaprej določena vrednost. S tem načinom se napaka v povprečju zmanjša iz 4,8/odčitek na 0,11/odčitek (Pevzner in sod., 2001b).




Slika 9: Popravek napake na petem nukleotidu.
Figure 9: Error correction on the fifth nucleotide.

Za izgradnjo de Bruijnovega grafa odčitke razdelijo na dele velikosti $k$. Iz teh delov zgradijo usmerjene grafe z dvema vozliščema. Povezava je označena s $k$-delom, vozlišče

pa s predpono oziroma s pripono dela velikosti $k-1$ (slika 10). Nato postopno združujejo enaka vozlišča, dokler ne ostanejo samo edinstvena vozlišča (slika 11). Vozlišče, ki nima vstopne povezave je začetno vozlišče. Vozlišče, ki nima izstopne povezave, pa je končno vozlišče. Razvejana vozlišča so tista, ki imajo več vstopnih in izstopnih povezav.



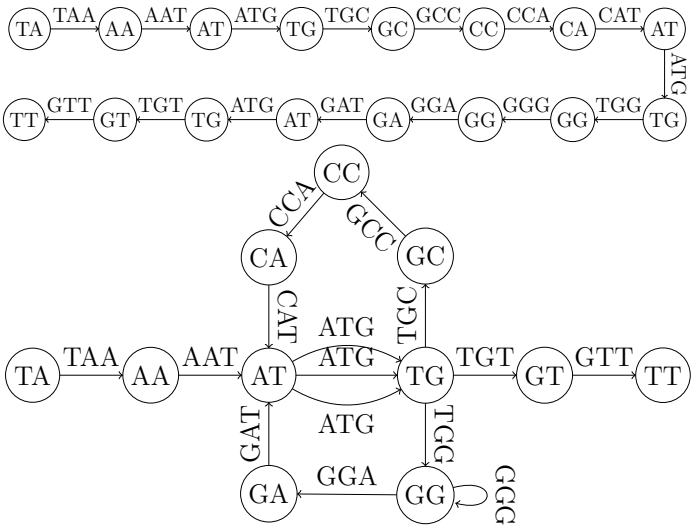

Slika 10: Izgradnja de Bruijnovega grafa za odčitek TAATGCCATGGGATGTT ($k$ = 3) (Compeau in Pevzner, 2014).
Figure 10: de Bruijn graph construction for read TAATGCCATGGGATGTT (k = 3) (Compeau in Pevzner, 2014)

Velikost $k$-dela je ekvivalentna spremenljivki $T$ v Lander-Watermanovem modelu. Število $k$ delov določimo s $k_n = N \cdot (L - T + 1)$. Globino za $k_n$ izračunamo z $d_k = \frac{k_n}{g}$. Število sosesk je enako $(\frac{G \cdot c}{L}) \cdot e^{-c \cdot (L-T+1/L)}$, če predpostavimo, da je število sosesk odvisno od števila najbolj desnih odčitkov, kjer je $e^{-c \cdot (L-T/L)}$ verjetnost, da je odčitek najbolj desni. Število sosesk za de Bruijnov algoritem se izračuna po enačbi 15. Iz tega je razvidno, da je število sosesk v večji meri odvisno od globine $k$-delov, namesto od globine sekvenciranja (Li in sod., 2012).

$$s = (\frac{d_k}{L - K + 1}) \cdot (G \cdot e^{-d_k}) \tag{15}$$

Sestava de Bruijnovega grafa je odvisna od velikosti $k$-delov in pokritosti genoma, ki jo dobimo z odčitki. Zaradi nepopolne pokritosti, napak pri sekvenciranju in ponavljanj, se tudi z Eulerejvo potjo ne more sestaviti celotnega genoma, saj je v grafu lahko več poti (slika 11) ali pa le-te ni. Zaradi tega se večinoma odčitke združuje v soseske, ki so deli genoma. Pri de Bruijnovem grafu so to vse nerazvejane poti, ki jih sestavljajo tista vozlišča, kjer je število vstopnih in iztopnih povezav enako, kar ne velja za začetno

in končno vozlišče. Za zgornji primer bi tako dobili devet sosesk (TAAT, TGTT, TGCCAT, ATG, ATG, ATG, TGG, GGG in GGAT).




Slika 11: Združevanje enako označenih vozlišč (povzeto po Compeau in Pevzner (2014)).
Figure 11: Gluing all identically nodes (reprinted from Compeau in Pevzner (2014)).

Vsak algoritem za sestavo nukleotidnega zaporedja ima svoje prednosti in slabosti. Zaradi napak pri sekvenciranju in ponavljanj oba pristopa ne moreta sestaviti celotnega genoma in iz odčitkov sestavita le soseske. Glede na časovno in prostorsko zahtevnost algoritma je za krajše odčitke z večjo pokritostjo bolje izbrati algoritem z de Bruijnovim grafom. Pri daljših odčitkih z manjšo pokritostjo je bolj smiselno izbrati algoritem OLC. Izbira algoritma je torej odvisna predvsem od vrste sekvenciranega genoma in od tehnologije sekvenciranja (Li in sod., 2012).

### 2.2.1 Grupiranje sosesk

Pri združevanju odčitkov se pojavlja tudi problem števila sosesk. Programi tvorijo večje število sosesk, kot je genov, ker ločijo med alelnimi oblikami istega gena, zaradi napak pri sekvenciranju, ponavljanj, polimorfizmov (SNP) itd. Zato poskušajo genetiki poiskati različne pristope, s katerimi bi bilo mogoče določiti soseske, ki pripadajo istemu genu oziroma transkriptu (Davidson in Oshlack, 2014; Grabherr in sod., 2011).

Za grupiranje sosesk, in s tem določanje ekspresije genov, obstajajo različni programi.

En izmed takih je Corset, ki odčitke najprej združi v soseske in jih nato grupira z algoritmom za hierarhično grupiranje. Razdalja med soseskama je določena glede na število skupnih odčitkov (enačba 16) (Davidson in Oshlack, 2014).

$$d(a,b) = \begin{cases} 1 - \frac{R_{ab}}{min(R_a,R_b)} \\ 1 \end{cases} \tag{16}$$

$R_a$ je skupno število odčitkov, ki se mapirajo na sosesko $a$, $R_{ab}$ je skupno število odčitkov, ki se mapirajo na sosesko $a$ in $b$. Vrednost razdalje je tako med 0, ki pove, da sta soseski podobni in spadata v isto grupo, in 1, ki pove, da soseski ne spadata v isto grupo. Hierarhično grupiranje poteka tako, da se najprej združi soseski, med katerima je najmanjša razdalja. Število odčitkov, ki pripadajo grupi ($R_{a'}$), se izračuna ponovno (enačba 17) in prav tako se na novo izračuna tabela razdalj (enačba 16).

$$R_{a'} = R_a + R_b - R_{ab} \tag{17}$$

Računanje razdalj in dodajanje soseske v grupo se ponavlja, dokler ne zgrupiramo vseh sosesk oziroma, dokler ni minimalna razdalja večja od vnaprej določene vrednosti.

Nekoliko drugačen pristop za določanje transkriptoma uporablja metoda Trinity, ki je sestavljena iz treh programskih modulov: Inchworm, Chrysalis in Butterfly (Grabherr in sod., 2011). Inchworm (i) iz odčitkov zgradi seznam podnizov velikosti $k$ ($k$ = 25), (ii) odstrani podnize, ki bi lahko vsebovali napako, (iii) izbere najbolj pogost podniz in ga odstrani iz seznama, (iv) združi podniz z naslednjim najbolj pogostim podnizom, kjer je prekrivanje $k-1$ in ga odstrani iz seznama, (v) podaljšuje podniz, dokler je to mogoče, (vi) ponovi korak od (iii) do (v), dokler ni seznam prazen. Chrysalis (i) grupira soseske, če je med njimi prekrivanje $k-1$, (ii) zgradi de Bruijnove grafe za posamezno skupino in za vsako povezavo določi utež, ki je število podnizov iz odčitkov, iz katere je povezava sestavljena, (iii) dodeli odčitek k tisti komponenti, s katero deli največje število podnizov. Butterfly zgradi transkripte za izooblike in paralogne gene.

## 2.3 ALGORITMI ZA ISKANJE MAKSIMALNE KLIKE

Za algoritme je ključnega pomena čas, ki ga potrebujejo za iskanje rešitve. Na instanci velikosti $n$ čas računanja definiramo kot maksimalno število korakov, ki jih algoritem potrebuje, da poda odgovor za poljubno instanco velikosti $n$. Za primer, če ima algoritem časovno zahtevnost $f(n) = n^3$, pomeni, da bo na vhodnih podatkih velikosti 4, naš algoritem naredil največ 64 korakov. Za določanje zgornje meje časovne zahtevnosti algoritma se v računalništvu uporablja asimptotična notacija, ki jo označimo z velikim $\mathcal{O}$ (Cormen in sod., 2009). V matematiki $\mathcal{O}$ notacija opiše obnašanje funkcije, ko se njen argument približuje neki vrednosti oziroma neskončnosti. Če imamo funkcijo $f(n)$ in $g(n)$, potem je:

$$\mathcal{O}(g(n)) = \{f(n) : \text{če } \textit{obstajata pozitivni konstanti } c \textit{ in } n_0, \textit{ tako da}$$
$$0 \leq f(n) \leq cg(n) \; \forall \; n > n_0\}$$

$\mathcal{O}(n^3)$ pomeni, da algoritem, za instanco velikosti $n \geq n_0$, ne bo nikoli porabil več kot $cn^3$ korakov.

Če za nek problem obstaja algoritem, katerega časovna zahtevnost je polinom velikosti vhoda, potem pravimo, da obstaja polinomski algoritem za reševanje našega problema. V nasprotnem primeru pravimo, da polinomski algoritem ne obstaja. Pojem NP predstavlja množico problemov za katere obstaja polinomski algoritem na nedeterminističnem (Turingovem) stroju (Sedgewick in Wayne, 2011). V množico NP spada tudi problem iskanja maksimalne klike.

V osnovi probleme iz NP rešujemo s:

- hevrističnimi algoritmi,
- aproksimacijskimi algoritmi,
- verjetnostnimi algoritmi (Cormen in sod., 2009).

Značilnost hevrističnih algoritmov je, da ne garantirajo, da je dobljena rešitev tudi najboljša končna rešitev. V to skupino uvrščamo požrešne algoritme, s katerim je možno reševati problem trgovskega potnika. Naloga trgovskega potnika je obiskati vsa mesta in se na koncu vrniti v izhodišče, tako da bo skupna opravljena razdalja

najkrajša. Problem lahko modeliramo s pomočjo utežnega grafa in ga rešimo tako, da za vsako naslednje vozliče (kraj), ki ga bomo obiskali, izberemo vozlišče, do katerega je najkrajša pot. Kot je bilo opisano v poglavju 2.1.2, in bomo videli pri algoritmu za iskanje maksimalne klike, lahko s požrešnim algoritmom rešujemo tudi problem barvanja grafov.

Aproksimacijski algoritmi zagotavljajo, da se rešitev $C$, za poljubno instanco velikosti $n$, od optimalne rešitve $C^*$, ne bo razlikovala za več kot faktor $p(n)$ (enačba 18).

$$max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq p(n)) \tag{18}$$

Glede na naravo problema je lahko optimalna rešitev z minimalno vrednostjo $0 < C^* < C$ ali pa maksimalno vrednostjo $0 < C < C^*$ (Cormen in sod., 2009). Izkazalo se je, da z aproksimacijskimi algoritmi ni možno najti rešitve za iskanje maksimalne klike, ki bi bila zadovoljivo blizu optimalni rešitvi (Håstad, 1999).

Verjetnostni algoritmi, z vključevanjem verjetnosti v korake računanja, lahko izboljšajo časovno zahtevnost algoritma in na ta način hitreje najdejo oziroma se približajo optimalni rešitvi. Tipičen primer takšnega algoritma je *quicksort* (Cormen in sod., 2009).

Kljub temu, da je iskanje maksimalne klike zahteven problem, so razvili algoritme, ki s pomočjo metode deli in omeji hitro najdejo kliko na veliko praktičnih primerih (Carraghan in Pardalos, 1990; Östergård, 2002; Tomita in Seki, 2003; Konc in Janežič, 2007). Metoda razveji in omeji izmed vseh mogočih rešitev pregleduje le tiste, ki lahko dajo boljšo rešitev od trenutno najdene. Bistvo te metode je iskanje dobrega kriterija za omejevanje, oziroma ugotavljanje, kdaj lahko določeno rešitev zavržemo. V nadaljevanju bomo opisali algoritem Mcqd (Konc in Janežič, 2007), ki za omejevanje uporablja barvanje vozlič.

Algoritem Mcqd (slika 12) uporablja dve globalni spremenljivki ($Q$ in $Q_{max}$). $Q$, ki je množica vozlišč v trenutni kliki, in $Q_{max}$, ki je množica trenutno največje najdene klike. Vhodni podatki so množica vozlišč $R \subseteq V(G)$, ki so urejena naraščajoče, glede na pripadajočo barvo, množico barv in nivo (*level*), na katerem se nahaja rekurzivna funkcija. Algoritem izbere vozlišče $p \, \epsilon \, R$ z največjo vrednostjo barve in ga odstrani iz

$R$. Če velja $|Q| + C(p) > |Q_{max}|$, potem vozlišče $p$ doda v $Q$ in iz njegovih sosedov ($\Gamma(p)$) izračuna novo množico vozlišč ($R_p$). Sledi novo barvanje vozlišč in klic Mcqd, če je $R_p \neq \emptyset$. V nasprotnem primeru se vozlišča iz množice $Q$ kopirajo v množico $Q_{max}$, če velja $|Q| > |Q_{max}|$. Algoritem začne s sestopanjem tako, da odstrani $p$ iz $Q$ in izbere novo vozlišče iz $R$. To se ponavlja, dokler je $R \neq \emptyset$.

```
1: procedure MCQD(R, C, level)
2:     S[level] := S[level] + S[level − 1] − S_old[level];
3:     S_old[level] := S[level − 1];
4:     while R ≠ ∅ do
5:         izberi vozlišče p z max(C(p)) zadnje vozlišče iz R;
6:         R := R\{p};
7:         if |Q| + C(p) > |Q_max] then
8:             Q := Q ∪ {p};
9:             R_p = R ∩ Γ(p)
10:            if R_p ≠ ∅ then
11:                if S[level]/ALL_STEPS < T_limit then
12:                    izračunaj stopnjo vozlišč v R_p;
13:                    uredi vozlišča v R_p padajoče
14:                    glede na stopnjo;
15:                ColorSort(R_p, C')
16:                S[level] := S[level] + 1;
17:                ALL_STEPS := ALL_STEPS + 1;
18:                MCQD(R_p, C', level + 1);
19:            else if |Q| > |Q_max| then
20:                Q_max := Q;
21:            Q := Q\p
22:        else
23:            return
```

Slika 12: Algoritem Mcqd (Konc in Janežič, 2007).
Figure 12: Algorithm Mcqd (Konc in Janežič, 2007).

Za določanje zgornje meje maksimalne klike algoritem Mcqd uporablja barvanje vozlišč. Pred začetkom se vozlišča pobarva tako, da se jih najprej glede na stopnjo uredi padajoče, nato se jim določi vrednost barve, tako da velja (Tomita in Seki, 2003):

$$C[R[i]] := i \ za \ i \leq \Delta(G)$$

in

$$C[R[i]] = \Delta(G) + 1 \ za \ \Delta(G) + 1 \leq i \leq |G(V)|$$

V vseh nadaljnjih korakih se uporabi požrešen algoritem za barvanje vozlišč (slika 13). Barvanje poteka tako, da se vsako vozliče iz $R$ dodeli v prvo možno barvno skupino ($C_k$), v kateri nima soseda. V kolikor ima vozlišče v vseh barvnih skupinah soseda ($k > max\_no$), algoritem doda novo barvo. Algoritem vrne novo množico vozlišč $R$, ki so urejena naraščajoče glede na $k$, in množico barv kateri pripadajo vozliča.

Iz algoritma Mcqd je razvidno, da so vozlišča dodana v množico $Q$ le, če je $|Q| + C[p] > |Q_{max}|$. Požrešen algoritem za barvanje vozlišč najprej določi minimalno številko barve ($k_{min}$), s katero mora biti vozlišče pobarvano, da je lahko dodano v množico $Q$. Vsa vozlišča, ki pripadajo barvnim skupinam $k < k_{min}$, ni potrebno razvrstiti glede na dodeljeno barvo in so zato postavljena na začetek množice $R$. Zadnjemu takemu vozlišču v $R$ je dodeljena barva 0.

Algoritem Mcqd je izboljšana verzija algoritma za iskanje maksimalne klike (MCQ), ki sta ga razvila Tomita in Seki (2003). V nasprotju z MCQ, ki vozlišča glede na stopnjo uredi le enkrat na začetku, se Mcqd pred ponovnim barvanjem dinamično odloča za urejevanje vozlišč glede na stopnjo v induciranem grafu $G = (R, E)$, saj se tako tesneje določi zgornja meja za maksimalno kliko (Konc in Janežič, 2007).

Algoritem Mcqd ima dve globalni spremenljivki $S[level]$ in $S_{old}[level]$. Prva hrani vsoto vseh korakov, ki jih je algoritem naredil do trenutnega nivoja, druga pa vsoto vseh korakov, ki jih je algoritem naredil do prejšnjega nivoja. Na vsakem nivoju algoritem najprej izračuna $S[level] := S[level] + S[level - 1] - S_{old}[level]$ in $S_{old}[level] := S[level - 1]$. Za urejevanje vozlišč glede na stopnjo pa se odloči, če je vrednost $S[level]/ALL\_STEPS < T_{limit}$. $ALL\_STEPS$ je števec vseh korakov, ki jih je algoritem naredil. $T_{limit}$ je vrednost, ki je bila eksperimentalno določena in znaša 0,05 za goste grafe ter 0,0 za redke grafe (Konc in Janežič, 2007). Konc in Janežič (2007) sta za približno mejo izbrala 0,025.

```
1: procedure ColorSort(R, C)
2:     max_no := 1;
3:     k_min := |Q_max| − |Q| + 1;
4:     if k_min ≤ 0 then
5:         k_min := 1;
6:     j := 0;
7:     C_1 := ∅;
8:     C_2 := ∅;
9:     for i := 0 do |R| − 1 do
10:        p := R[i];
11:        k := 1;
12:        while C_k ∩ Γ(p) ≠ ∅ do
13:            k := k + 1;
14:        if k > max_no then
15:            max_no := k;
16:            C_{max_no+1} := ∅;
17:        C_k := C_k ∪ p;
18:        if k < k_min then
19:            R[j] := R[i];
20:            j := j + 1;
21:    C[j − 1] := 0;
22:    for k := k_min do max_no do
23:        for i := 1 do |C_k| do
24:            R[j] := C_k[i];
25:            C[j] := k;
26:            j := j + 1;
```

Slika 13: Požrešen algoritem za barvanje vozlišč (Konc in Janežič, 2007).
Figure 13: Greedy algorithm for the vertex coloring problem (Konc in Janežič, 2007).

## 2.4 KLIKA KOT METODA GRUPIRANJA

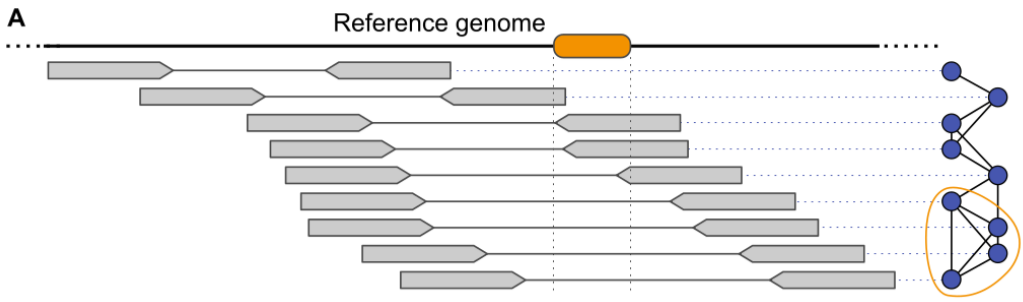
Velikokrat želimo v množici objektov (podatkov) med seboj združiti tiste, ki imajo podobne lastnosti. Problem grupiranja je mogoče reševati tudi s pomočjo grafov, kar se je pokazalo tudi na področju bioinformatike, in sicer pri:

- določanju sekundarne strukture RNA (Ji in sod., 2004),
- iskanju povezav med geni za določanje različnih bioloških pojavov (Matsunaga in sod., 2009; Rokhlenko in sod., 2007),
- analizi proteinskih struktur (Dukka in sod., 2006).

Ključnega pomena pri reševanju omenjenih problemov je iskanje klike v grafu, ki je ena izmed glavnih metod za grupiranje.

Matsunaga in sod. (2009) so s pomočjo maksimalne klike iskali gene, ki naj bi vplivali na bolezni. S pomočjo baze Online Mendelian Inheritance in Man so zgradil graf. Vozlišča so bila geni in bolezni v bazi. Povezava med vozlišči je bila narejena, če je obstajala povezava med geni oziroma geni in boleznijo znotraj baze. Z iskanjem maksimalnih klik v takšnem grafu, je mogoče bolje razumeti kompleksne mehanizme, ki pripomorejo k razvoju bolezni.

Določanje virusnega haplotipa znotraj gostitelja je pomemben podatek za določanje virulence in patogeneze virusev. Töpfer in sod. (2014) so z metodo računanja maksimalnih klik na parnih odčitkih, pridobljenih z metodo sekvenciranja naslednje generacije, uspeli rekonstruirati strukturo virusne kvazivrste. Razvili so program HaploClique, ki na grafu poravnave med odčitki išče maksimalne klike. V grafu poravnave vsak odčitek predstavlja vozlišče, med vozliščema pa je povezava, če je med odčitkoma dovolj dobro prekrivanje (slika 14). V takšnem grafu maksimalna klika predstavlja množico odčit-




Slika 14: Graf poravnave med odčitki. Zadnji štirje odčitki se med seboj prekrivajo in tvorijo maksimalno kliko v grafu (Töpfer in sod., 2014).
Figure 14: The read alignment graph. Overlapping of the last four reads, forming a maximal clique (Töpfer in sod., 2014).

kov, ki se med seboj prekrivajo in lokalno predstavljajo isti del sekvence. HaploClique vsako maksimalno kliko poveže v novo daljšo sekvenco, ki tvori novo vozlišče na grafu poravnave med odčitki, nato poišče maksimalno kliko na novem grafu in to ponavlja, dokler je možno tvoriti daljše sekvence (Töpfer in sod., 2014).

# 3 MATERIALI IN METODE

## 3.1 PODATKI ZA PRIMERJAVO HITROSTI ALGORITMOV

Hitrost delovanja algoritmov za iskanje maksimalne klike smo testirali na naslednjih podatkih:

- grafi za iskanje maksimalne klike iz drugega tekmovanja DIMACS leta 1992-1993 (Johnson in Trick, 1996),
- primerjalni grafi, ki so bili zgrajeni z metodo zvezdnega komplementa in se uporabljajo za rekonstrukcijo danega grafa $G$, pri čemer poznamo njegov manjši induciran podgraf $H$, ki ima določene lastnosti (Cvetkovic in sod., 1997),
- proteinski produktni grafi, ki se jih uporablja za ugotavljanje povezanosti med proteini (Depolli in sod., 2013),
- graf podobnosti sosesk (poglavje 3.3.2) in graf ujemanja z znanimi referenčnimi sekvencami (poglavje 3.3.2).

## 3.2 ANALIZA ALGORITMOV ZA ISKANJE MAKSIMALNE KLIKE

Za hitrost iskanja maksimalne klike smo primerjali algoritma Mcqd (Konc in Janežič, 2007) in Gtc. Ideja za Gtc je bila predstavljena v Azarija in Marc (2015), implementacija pa v Azarija in Slana (v pripravi). Algoritem Gtc za iskanje maksimalne klike združuje algoritem za iskanje simetrij v grafu in Mcqd.

Algoritem Gtc (slika 15) za iskanje maksimalne klike uporablja pojem simetrij. Vozlišča iz iste orbite ležijo v enako veliki kliki, zato omejimo iskanje le na posamično vozlišče iz orbite z največ vozlišči (slika 15, vrstica 9). Vhodni podatek je graf ($G$). Če je velikost grafa večja od naprej določene vrednosti ($|V(G)| > N$), algoritem najprej izračuna orbite s funkcijo *ComputeOrbits* (slika 15, vrstica 6). Le-ta vsak par vozlišč $(u, v)$ grafa $G$ doda v skupno orbito, če obstaja generator $g_i$ tako, da je $g_i(u) = v$. To opravilo naredi s podatkovno strukturo disjunktnih dreves. Algoritem generatorje grupe $Aut(G)$ poišče s programom *bliss-0.72* (Junttila in Kaski, 2007). Nato iz največje orbite izbere

prvo vozliče $v$ (slika 15, vrstica 9) ter iz njegovih sosedov zgradi nov induciran podgraf $U$. V kolikor je njegova velikost večja od 0 ($|U(G)| > 0$) se ponovno kliče Gtc. Vozliča iz trenutne največje orbite se odstrani iz grafa in množice orbit (slika 15, vrstica 16 in 17). Algoritem to ponavlja tako dolgo dokler je velikost grafa večja od naprej določene vrednosti ($|V(G)| > N$). Sledi izračun maksimalne klike s programom Mcqd (slika 15, vrstica 18). Pri sestopanju algoritem ugotavlja ali je velikost izračunane klike ($cl$) večja od velikosti trenutne maksimalne klike ($|cl| + 1 > |cmax\_clique|$). Pri resničnem pogoju, trenutna maksimalna klika postane na novo izračunana klika in se ji doda trenutno izbrano vozlišče ($v$).

```
1:  procedure GTC(G)
2:      cl := ∅;
3:      cmax_clique := ∅;
4:      orbits := ∅;
5:      if |V(G)| > N then
6:          orbits := COMPUTEORBITS(G);
7:      while |V(G)| > N do
8:          o := PICKMAXORBIT(orbits)
9:          v := o[0]
10:         U := G ∩ Γ(v)
11:         if |V(U)| > 0 then
12:             cl = GTC(U)
13:             if |cl| + 1 > |cmax_clique| then
14:                 cmax_clique = cl
15:                 cmax_clique = cmax_clique ∪ v
16:         G := G\{o}
17:         orbits := orbits\{o}
18:     cl := COMPUTEMCQD(G)
19:     if |cl| > |cmax_clique| then
20:         return cl
21:     else
22:         return cmax_clique
```

Slika 15: Algoritem Gtc.
Figure 15: Algorithm Gtc.

Število iskanj simetrij oziroma globino rekurzije pri algoritmu Gtc je mogoče določiti na dva načina. Lahko jo definiramo z velikostjo grafa ali pa z maksimalnim številom nivojev, do katerega se išče simetrije. Iz slike 16 je razvidno, da bo Gtc na grafu poiskal simetrije, dokler bo $|G| > N$ (npr. nivo 1). Na nivoju dva je $|G| \leq N$, zato

algoritem takoj izračuna maksimalno kliko. Pri sestopanju se iz vsake veje nivoja vrne maksimalno kliko, ki je bila najdena, bodisi na ostanku grafa bodisi iz posamezne orbite.

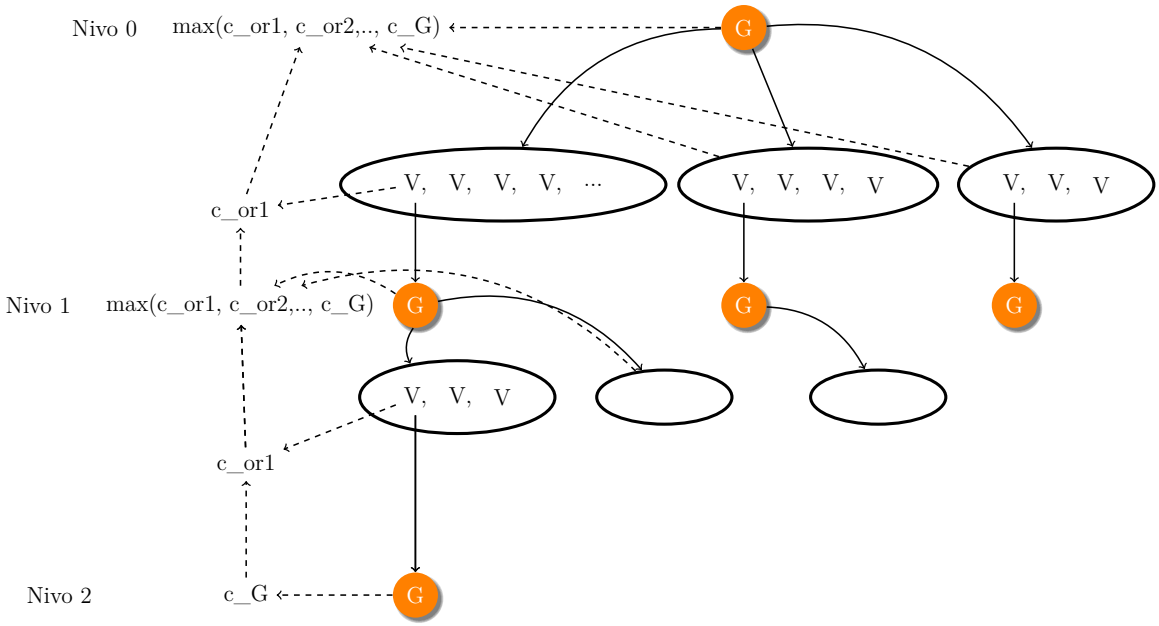
Algoritem smo testirali z obema možnostma za število iskanj simetrij:

1. Prvič smo globino rekurzije določili z relativno velikostjo vhodnega grafa ($N = |V(G)| \cdot 0{,}50$ in $N = |V(G)| \cdot 0{,}75$).
2. Drugič smo globino rekurzije določili s številom nivojev (L1, L4, L16) in z relativno velikostjo vhodnega grafa ($N = |V(G)| \cdot 0,1$).

V obeh primerih je algoritem iskal simetrije na grafu vsaj enkrat.

Pri programu Mcqd in Mcqd v Gtc-ju je bila vrednost $T_{limit} = 0{,}025$. Za prevajanje programske kode smo uporabili prevajalnik *g++-4.9* z optimizacijo *-O3*.

Maksimalno kliko smo na vsakem grafu z obema programoma iskali trikrat. Za čas iskanja maksimalne klike smo določili povprečje vseh treh časov. Če program v dveh urah ni našel maksimalne klike, smo iskanje prekinili.




Slika 16: Prikaz delovanja algoritma Gtc.
Figure 16: Demonstration of algorithm Gtc.

## 3.3 GRUPIRANJE PODOBNIH SOSESK

Za testiranje grupiranja sosesk smo uporabili soseske, ki so bile zgrajene iz RNA hmelja (*Humulus lupulus L.*) okuženega z glivo *Verticillium albo-atrum*. Vzorci so bili pridobljeni iz dveh kultivarjev v štirih časovnih obdobjih (6, 12, 18, 30 dni) po okužbi. Sekvenciranje je bilo narejeno na napravi Illumina GA HiSeq2000 z velikostjo odčitkov 50bp. Sestavljanje odčitkov je bilo narejeno s programom CLC-Genomics. Iz skupno 822632796 odčitkov je bilo sestavljenih 89201 sosesk. Podatki so bili pridobljeni na Katedri za genetiko, biotehnologijo, statistiko in žlahtnjenje rastlin, Biotehniške fakultete, Univerze v Ljubljani. Objava podatkov je še v pripravi.

Iz sosesk smo zgradili dve vrsti grafov, ki sta bila zgrajena glede na podobnost med soseskami in glede na ujemanja z znanimi referenčnimi sekvencami (BLASTx).

Problem združevanja podobnih sosesk smo definirali na naslednji način. Za dani graf $G$ poišči minimalen $k$ tako, da obstaja particija $V_1, \ldots, V_k$, da za vsak $1 \leq i \leq k$ velja, da $V_i$ vsebuje vozlišča, ki pripadajo istemu genu oziroma transkriptu.

Iz zgornje definicije izstopa problem, kako zagotoviti, da vozlišča oziroma soseske pripadajo istemu genu oziroma transkriptu. Problem smo zato prevedli na iskanje maksimalne klike v grafu. Za opis postopka združevanja bomo vpeljali pojem *dekompozicija grafa z maksimalno kliko*. Naj bo $G$ nek graf. Zaporedju podgrafov $H_1, \ldots, H_n$ grafa $G$ pravimo dekompozicija na maksimalne klike, če velja:

1. $V(H_i) \cap V(H_j) = \emptyset$ za vsak $1 \leq i < j \leq n$
2. $V(H_1) \cup \cdots \cup V(H_k) = V(G)$
3. $H_i$ *je maksimalna klika v* $G - (V(H_1) \cup \cdots \cup V(H_{i-1}))$ za vsak $1 \leq i < j \leq n$

Dekompozicijo dobimo z algoritmom na sliki 17.

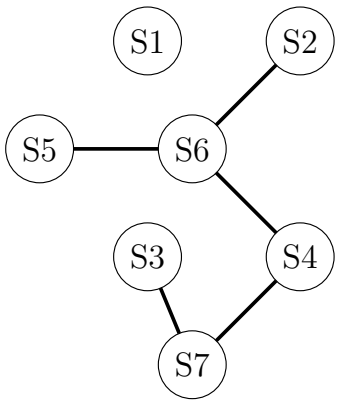
Predvidevamo, da soseske, ki so združene v maksimalni kliki, pripadajo istemu genu oziroma transkriptu. Ugotavljali smo, kolikšna je povezanost sosesk znotraj klike in velikost ter število klik v posameznem grafu.

1: $AllClq := \emptyset$
2: $cl :=$ computeMaxClq($G$)
3: **while** $|cl| > 1$ **do**
4: $\quad AllClq \leftarrow cl$
5: $\quad$ **for** $i := 0$ *do* $|cl|$ **do**
6: $\quad\quad G :=$ removeVertex($G$, $cl[i]$)
7: $\quad cl :=$ computeMaxClq($G$)

Slika 17: Algoritem dekompozicije grafa z maksimalno kliko.
Figure 17: Algorithm of graph decomposition based on the maximum clique method.

### 3.3.1 Graf podobnosti sosesk

Graf glede na podobnost med soseskami smo zgradili s programom Clustal Omega (Sievers in sod., 2011), ki je odprtokodni program za večsegmentno poravnavo proteinov, DNA in RNA. Večsegmentno poravnavo naredi s pomočjo filogentskega drevesa, ki ga zgradi iz matrike razdalj med soseskami. Za računanje razdalj med soseskami uporablja metodo *K-tuple*, ki je natančneje opisana v Wilbur in Lipman (1983). Matriko razdalj med soseskami smo uporabili za izgradnjo grafa podobnosti sosesk. Vrednost 0 v matriki pomeni popolno povezanost med soseskama, vrednost 1 pa nepovezanost med soseskama. V grafu podobnosti sosesk vsaka soseska predstavlja vozlišče. Med vozliščema je povezava, če je vrednost razdalje (uteži) med soseskama manjša od mejno določene. Prva mejna vrednost uteži je bila 0,95 zadnja pa 0,05. Vrednost uteži smo zniževali za faktor 0,05. Na sliki 18 je primer izgradnje grafa podobnosti sosesk za mejno vrednost uteži 0,2, ki smo ga zgradili iz podatkov v preglednici 1.




Slika 18: Graf podobnosti sosesk. Soseski sta povezani, če je razdalja med njima $< 0{,}2$.
Figure 18: Similarity graph. There is an edge between two contigs if distance between them is $< 0{,}2$.

Preglednica 1: Matrika razdalj med sedmimi soseskami.
Table 1: Pairwise distance matrix between seven contigs.

| Soseska | S1 | S2 | S3 | S4 | S5 | S6 | S7 |
|---|---|---|---|---|---|---|---|
| S1 | 0 | 0,28 | 0,27 | 0,25 | 0,21 | 0,29 | 0,26 |
| S2 | 0,28 | 0 | 0,27 | 0,2 | 0,24 | 0,05 | 0,23 |
| S3 | 0,27 | 0,27 | 0 | 0,24 | 0,24 | 0,22 | 0,17 |
| S4 | 0,25 | 0,2 | 0,24 | 0 | 0,42 | 0,15 | 0,19 |
| S5 | 0,21 | 0,24 | 0,24 | 0,42 | 0 | 0,14 | 0,20 |
| S6 | 0,29 | 0,05 | 0,22 | 0,15 | 0,14 | 0 | 0,25 |
| S7 | 0,26 | 0,23 | 0,17 | 0,19 | 0,20 | 0,25 | 0 |

Program Clustal Omega smo zagnali z naslednjim ukazom in možnostmi:

```
$ ./clustalo -i input_file.fa -o output_alignment.fa
--distmat-out=output_distance.mat --full --threads=8 -v
```

- *i* za vhodno datoteko s soseskami,
- *o* za izhodno datoteko s poravnavami,
- *distmat-out* za zapis matrike razdalj v datoteko,
- *full* za izračun razdalje med vsem pari,
- *threads* za število procesov,
- *v* za "verbose mode".

Uporabili smo verzijo programa 1.2.1, v kateri smo odkrili hrošča v datoteki *pair_dist.c* vrstica 317. Parametra *iend* in *jend* v funkciji *PairDistances* sta bila deklarirana kot tip *int*. Tako je pri računanju števila parov, med katerimi smo morali izračunati razdalje, prišlo do celoštevilskega preliva. Kodo smo popravili tako, da smo oba parametra spremenili v tip *const unsigned long int*.

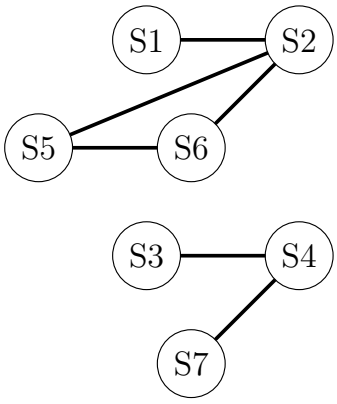
### 3.3.2 Graf ujemanja z znanimi referenčnimi sekvencami

BLASTx je program za lokalno poravnavo povpraševalnega zaporedja DNA/RNA s proteinskim zaporedjem v podatkovni bazi. Povpraševalno zaporedje DNA/RNA prevede v šest bralnih okvirjev (Wheeler in Bhagwat, 2007). Poleg poravnave BLASTx

vrne vrednost bit ($S^{'}$) in pričakovano vrednost $E$. Bit vrednost je korigirana vrednost poravnave ($S$), ki nam pove, kako dobro se sekvenci prekrivata. Iz bit vrednosti se lahko izračuna pričakovano vrednost $E$ (enačba 19), ki pove, koliko zadetkov lahko po naključju najdemo v bazi z enako vrednostjo $S$. Vrednost $m$ je dolžina iskalnega zaporedja, vrednost $n$ pa je celotna dolžina baze oziroma število vseh nukleotidnih ostankov v bazi (Kerfeld in Scott, 2011).

$$E = \frac{n \cdot m}{2^{S'}} \tag{19}$$

Graf ujemanja z znanimi referenčnimi sekvencami smo zgradili tako, da smo med seboj povezali soseski, če jih je BLASTx poravnal z isto znano sekvenco (proteinom). Pri povpraševanju smo za zgornjo pričakovano vrednost določili $E = 0,001$. Na sliki 19 je primer izgradnje grafa ujemanja z znanimi referenčnimi sekvencami, ki smo ga zgradili iz podatkov v preglednici 2.




Slika 19: Graf ujemanja z znanimi referenčnimi sekvencami. Soseski sta povezani, če jih je BLASTx poravnal z isto znano sekvenco.
Figure 19: Relation graph. There is an edge between two contigs if BLASTx aligned them with the same sequence.

Preglednica 2: Poravnava povpraševalnih zaporedij (soseske) s proteini v bazi. Imena proteinov so izmišljena in služijo le kot primer.
Table 2: Alignment of query sequnces with proteins in the database. Protein names are arbitrary, used to illustrate examples.

| Soseska | S1 | S2 | S2 | S3 | S3 | S4 | S4 | S4 | S5 | S5 | S6 | S6 | S7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Protein | PX1 | PX1 | PY3 | PX2 | PX5 | PY9 | PX2 | PX5 | PX8 | PY3 | PY3 | PX8 | PY9 |

Program smo zagnali z naslednjimi možnostmi:

```
blastx -db nr -evalue 0.001 -max_target_seqs 10 -outfmt "6 std stitle
```

```
sacc␣slen␣qlen␣qcovs"
```

- *db* podatkovna baza,
- *evalue* zgornja meja za pričakovano vrednost $E$,
- *max_target_seqs* število poravnanih zaporedji, ki so vključena v rezultat,
- *outfmt* oblika izpisa v tabelarni obliki.

## 3.4 Strojna oprema

Vse računske operacije so bil narejene na računalniku z naslednjmi lastnostmi:

- Procesor: Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
- Število procesorjev: 8
- Pomnilnik: 96 GB

Na računalniku je bil nameščen operacijski sistem Ubuntu 14.04.3 LTS (GNU/Linux 3.16.0-30-generic x86_64).

# 4 REZULTATI

Za testiranje smo uporabili 68 grafov. V prvem delu poglavja so predstavljene njihove značilnosti, v nadaljevanju pa časi, ki sta jih algoritma potrebovala za iskanje maksimalne klike v grafih. Vsi rezultati so v sekundah. Pri algoritmu Gtc smo nivo rekurzije oziroma števila iskanj simetrij v grafu določili z relativno velikostjo vhodnega grafa (0,5 in 0,75) in s številom nivojev (L1, L4, L16) (slika 16). V zadnjem delu so predstavljeni rezultati grupiranja sosesk v grafu podobnosti sosesk in v grafu ujemanja z znanimi referenčnimi sekvencami.

## 4.1 ZNAČILNOSTI GRAFOV

### 4.1.1 Graf podobnosti sosesk in graf ujemanja z znanimi referenčnimisekvencami

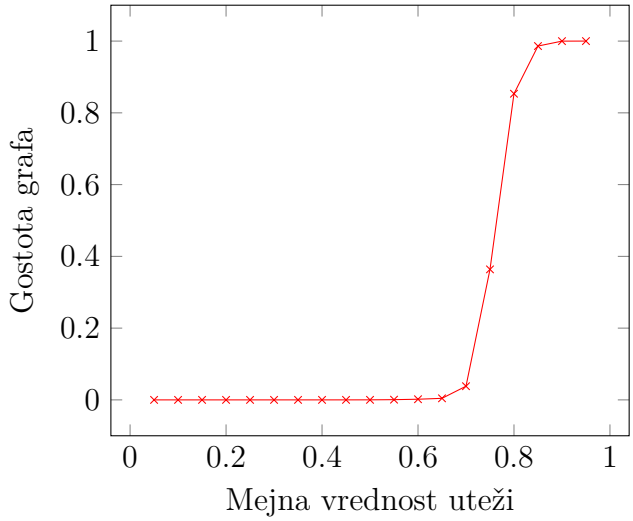
Iz podatkov v preglednici 3 lahko razberemo, da je med večino sosesk podobnost vsaj 20 %, saj je gostota grafa podobnosti sosesk, pri vrednosti uteži večji od 0,75, zelo velika. Z nižanjem vrednosti uteži pod 0,75 oziroma določanjem tesnejših mej za podobnost med soseskami, pa gostota grafa pada in se pojavi veliko izoliranih vozlišč. Največji padec gostote je v območju mejne vrednosti uteži od 0,75 do 0,65 (slika 20). Zelo redek je tudi graf ujemanja z znanimi referenčnimi sekvencami (*blast.clq*). Glede na število povezav in povprečno število povezav na vozlišče bi ga lahko primerjali z grafom z mejno vrednostjo uteži 0,5, vendar pa je največja stopnja vozlišča veliko manjša.

Preglednica 3: Značilnost grafa podobnost sosesk glede na mejno vrednost in grafa ujemanja z znanimi referenčnimi sekvencami.
Table 3: Properties of similarity graph depending on the cutoff point and properties of a relation graph.

| $G$ | $\|V\|$ | $\|E\|$ | $d(G)$ | $\Delta(G)$ | $\delta(G)$ | $2\|E\|/\|V\|$ |
|---|---|---|---|---|---|---|
| cutoff_0.05.clq | 89201 | 1234 | 0,000000310178 | 3 | 0 | 0,03 |
| cutoff_0.1.clq | 89201 | 2322 | 0,000000583657 | 4 | 0 | 0,05 |
| cutoff_0.15.clq | 89201 | 3312 | 0,000000832503 | 9 | 0 | 0,07 |
| cutoff_0.2.clq | 89201 | 4636 | 0,0000011653 | 13 | 0 | 0,1 |
| cutoff_0.25.clq | 89201 | 6522 | 0,00000163937 | 18 | 0 | 0,15 |
| cutoff_0.3.clq | 89201 | 9554 | 0,00000240149 | 171 | 0 | 0,21 |
| cutoff_0.35.clq | 89201 | 18494 | 0,00000464864 | 2587 | 0 | 0,41 |
| cutoff_0.4.clq | 89201 | 36346 | 0,00000913591 | 6073 | 0 | 0,81 |
| cutoff_0.45.clq | 89201 | 105588 | 0,0000265406 | 6956 | 0 | 2,37 |
| cutoff_0.5.clq | 89201 | 497952 | 0,000125165 | 7877 | 0 | 11,16 |
| cutoff_0.55.clq | 89201 | 2520723 | 0,000633608 | 8898 | 0 | 56,52 |
| cutoff_0.6.clq | 89201 | 7227049 | 0,00181659 | 9791 | 0 | 162,04 |
| cutoff_0.65.clq | 89201 | 17277721 | 0,00434292 | 10715 | 0 | 387,39 |
| cutoff_0.7.clq | 89201 | 151541707 | 0,0380915 | 40479 | 0 | 3397,76 |
| cutoff_0.75.clq | 89201 | 1447002908 | 0,363718 | 70438 | 1581 | 32443,6 |
| cutoff_0.8.clq | 89201 | 3394901458 | 0,853341 | 86862 | 11797 | 76118 |
| cutoff_0.85.clq | 89201 | 3923290306 | 0,986157 | 89200 | 20437 | 87965,2 |
| cutoff_0.9.clq | 89201 | 3977877226 | 0,999877 | 89200 | 75247 | 89189,1 |
| cutoff_0.95.clq | 89201 | 3978364578 | 1 | 89200 | 89194 | 89200 |
| blast.clq | 89201 | 688557 | 0,000173075 | 785 | 0 | 15,44 |

### 4.1.2 Grafi DIMACS

Za testiranje smo izbrali nekaj grafov DIMACS različnih velikosti (preglednica 4). Število vozlišč je od 64 pa do 3300. Vsi izbrani grafi so zelo gosti.




Slika 20: Gostota grafa podobnosti sosesk glede na mejno vrednost uteži.
Figure 20: Density of the similarity graph depending on the cutoff point.

Preglednica 4: Značilnosti grafov DIMACS.
Table 4: Properties of graphs DIMACS

| $G$ | $\|V\|$ | $\|E\|$ | $d(G)$ | $\Delta(G)$ | $\delta(G)$ | $2\|E\|/\|V\|$ |
|---|---|---|---|---|---|---|
| ž hamming6-2.clq | 64 | 1824 | 0,90 | 57 | 57 | 57 |
| hamming6-4.clq | 64 | 704 | 0,35 | 22 | 22 | 22 |
| johnson16-2-4.clq | 120 | 5460 | 0,76 | 91 | 91 | 91 |
| keller4.clq | 171 | 9435 | 0,65 | 124 | 102 | 110,35 |
| brock200_1.clq | 200 | 14834 | 0,75 | 165 | 130 | 148,34 |
| brock200_2.clq | 200 | 9876 | 0,50 | 114 | 78 | 98,76 |
| brock200_3.clq | 200 | 12048 | 0,61 | 134 | 99 | 120,48 |
| gen200_p0.9_55.clq | 200 | 17910 | 0,90 | 190 | 164 | 179,1 |
| san200_0.7_1.clq | 200 | 13930 | 0,70 | 155 | 125 | 139,3 |
| san200_0.7_2.clq | 200 | 13930 | 0,70 | 164 | 103 | 139,3 |
| san200_0.9_1.clq | 200 | 17910 | 0,90 | 191 | 159 | 179,1 |
| san200_0.9_2.clq | 200 | 17910 | 0,90 | 188 | 169 | 179,1 |
| san200_0.9_3.clq | 200 | 17910 | 0,90 | 187 | 166 | 179,1 |
| sanr200_0.9.clq | 200 | 17863 | 0,90 | 189 | 166 | 178,63 |
| C250.9.clq | 250 | 27984 | 0,90 | 236 | 203 | 223,87 |
| hamming8-4.clq | 256 | 20864 | 0,64 | 163 | 163 | 163 |
| brock400_1.clq | 400 | 59723 | 0,75 | 320 | 272 | 298,62 |
| brock400_2.clq | 400 | 59786 | 0,75 | 328 | 274 | 298,93 |
| brock400_3.clq | 400 | 59681 | 0,75 | 322 | 275 | 298,41 |

se nadaljuje

Preglednica 4 – nadaljevanje

| $G$ | $\|V\|$ | $\|E\|$ | $d(G)$ | $\Delta(G)$ | $\delta(G)$ | $2\|E\|/\|V\|$ |
|---|---|---|---|---|---|---|
| brock400_4.clq | 400 | 59765 | 0,75 | 326 | 275 | 298,83 |
| sanr400_0.5.clq | 400 | 39984 | 0,50 | 233 | 161 | 199,92 |
| sanr400_0.7.clq | 400 | 55869 | 0,70 | 310 | 252 | 279,35 |
| johnson32-2-4.clq | 496 | 107880 | 0,88 | 435 | 435 | 435 |
| C500.9.clq | 500 | 112332 | 0,90 | 468 | 431 | 449,33 |
| DSJC500_5.clq | 500 | 62624 | 0,50 | 286 | 220 | 250,50 |
| p_hat700-2.clq | 700 | 121728 | 0,50 | 539 | 157 | 347,79 |
| keller5.clq | 776 | 225990 | 0,75 | 638 | 560 | 582,45 |
| brock800_1.clq | 800 | 207505 | 0,65 | 560 | 477 | 518,76 |
| brock800_2.clq | 800 | 208166 | 0,65 | 566 | 472 | 520,42 |
| DSJC1000_5.clq | 1000 | 249826 | 0,50 | 551 | 447 | 499,65 |
| hamming10-2.clq | 1024 | 518656 | 0,99 | 1013 | 1013 | 1013 |
| keller6.clq | 3361 | 4619898 | 0,82 | 2952 | 2690 | 2749,12 |

### 4.1.3 Primerjalni grafi

Za testiranje hitrosti algoritmov smo izbrali nekaj primerjalnih grafov, ki so bili tvorjeni z metodo zvezdnega komplementa. Značilnost teh grafov je manjša gostota, velikost pa od 500 do 1500 vozlišč (preglednica 5).

Preglednica 5: Značilnosti primerjalnih grafov.
Table 5: Properties of comparability graphs.

| $G$ | $\|V\|$ | $\|E\|$ | $d(G)$ | $\Delta(G)$ | $\delta(G)$ | $2\|E\|/\|V\|$ |
|---|---|---|---|---|---|---|
| sim_1567_249281 | 1567 | 249281 | 0,20 | 1530 | 245 | 318,16 |
| sim_528_51954 | 528 | 51954 | 0,37 | 517 | 135 | 196,80 |
| sim_755_57086 | 755 | 57086 | 0,20 | 734 | 84 | 151,22 |
| sim_998_101738 | 998 | 101738 | 0,20 | 991 | 157 | 203,88 |
| sim_998_104321 | 998 | 104321 | 0,21 | 994 | 160 | 209,06 |
| sim_998_99631 | 998 | 99631 | 0,20 | 946 | 110 | 199,66 |

### 4.1.4 Proteinski produktni grafi

Velikost izbranih proteinskih produktnih grafov je zelo različna, vsi pa so zelo gosti (preglednica 6).

Preglednica 6: Značilnosti proteinskih produktnih grafov.
Table 6: Properties of protein product graphs.

| $G$ | $\|V\|$ | $\|E\|$ | $d(G)$ | $\Delta(G)$ | $\delta(G)$ | $2\|E\|/\|V\|$ |
|---|---|---|---|---|---|---|
| 1allA_3dbjC_41 | 451 | 98868 | 0,97 | 450 | 352 | 438,44 |
| 1f82A_1zb7A_5 | 655 | 207589 | 0,97 | 654 | 454 | 633,86 |
| 1KZKA_3KT2A_78 | 271 | 36044 | 0,99 | 270 | 197 | 266,01 |
| 2FDVC_1PO5A_83 | 750 | 270340 | 0,96 | 749 | 419 | 720,91 |
| 2UV8I_2J6IA_13107 | 200 | 17076 | 0,86 | 199 | 133 | 170,76 |
| 2W00B_3H1TA_10858 | 346 | 54257 | 0,91 | 345 | 221 | 313,62 |
| 2W4JA_2A2AD_0 | 563 | 155034 | 0,98 | 562 | 417 | 550,74 |
| 3HRZA_2HR0A_476 | 905 | 385020 | 0,94 | 904 | 527 | 850,87 |
| 3P0KA_3GWLB_0 | 138 | 8907 | 0,94 | 137 | 97 | 129,09 |
| 3ZY0D_3ZY1A_110 | 61 | 1792 | 0,98 | 60 | 53 | 58,75 |

## 4.2 PRIMERJAVA ČASOVNE ZAHTEVNOSTI ALGORITMOV

### 4.2.1 Grafi DIMACS

Na skoraj vseh grafih DIMACS je algoritem Mcqd delal hitreje (preglednica 7). Gtc je delal hitreje le na 6 grafih od 30. Iz tega je tudi razvidno, da bo Gtc deloval najhitreje, če bo izvedel iskanje simetrij le na enem nivoju. Zanimiv je predvsem graf *johnson32-2-4.clq*, kjer Mcqd maksimalne klike ne najde, Gtc pa jo najde v manj kot eni sekundi. Vendar je v tem primeru iskal simetrije na šestnajstih nivojih, oziroma dokler je bil podgraf večji od 10 % vseh vozlišč.

Preglednica 7: Primerjava časovne zathevnosti med algoritmoma na grafih DIMACS.
Table 7: Comparison of time complexity between algorithms on graphs DIMACS.

| | | | Gtc | | | | |
|---|---|---|---|---|---|---|---|
| *G* | *c_max* | Mcqd | 0,50 | 0,75 | L1 | L4 | L16 |
| MANN_a9.clq | 16 | **0,0001** | 0,0063 | 0,002 | 0,0012 | 0,0067 | 0,2356 |
| brock200_1.clq | 21 | **0,6672** | 4,9631 | 1,8523 | 1,9611 | 202,046 | 5219,8633 |
| brock200_2.clq | 12 | **0,0089** | 0,0369 | 0,0272 | 0,0385 | 0,9518 | 0,9467 |
| brock200_3.clq | 15 | **0,0362** | 0,2039 | 0,1052 | 0,1335 | 10,7537 | 14,4485 |
| brock400_1.clq | 27 | **499,4107** | 2919,8433 | 1418,5367 | 1462,4333 | $> 2h$ | $> 2h$ |
| brock400_2.clq | 29 | **213,6473** | 3256,42 | 1433,22 | 1489,1133 | $> 2h$ | $> 2h$ |
| brock400_3.clq | 31 | **428,8157** | 2715,4467 | 1256,87 | 1338,2967 | $> 2h$ | $> 2h$ |
| brock400_4.clq | 33 | **223,1473** | 3052,9 | 1296,43 | 1334,02 | $> 2h$ | $> 2h$ |
| brock800_1.clq | 23 | **7048,42** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| brock800_2.clq | 24 | **6020,96** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| C250,9.clq | 44 | **2444,8467** | $> 2h$ | $> 2h$ | 5430,9600 | $> 2h$ | $> 2h$ |
| gen200_p0,9_55.clq | 55 | **0,7089** | $> 2h$ | 89,0224 | 12,3004 | 6100,9767 | $> 2h$ |
| hamming10-2.clq | 512 | 5,9151 | $> 2h$ | $> 2h$ | **1,4141** | 991,1063 | $> 2h$ |
| hamming6-2.clq | 32 | **0,0002** | 0,0077 | 0,0046 | 0,0022 | 0,0074 | 1,8786 |
| hamming6-4.clq | 4 | **0,0001** | 0,0009 | 0,0009 | 0,0009 | 0,001 | 0,0011 |
| hamming8-4.clq | 16 | 0,0454 | 0,0453 | 0,0364 | **0,036** | 0,0511 | 0,0524 |
| johnson16-2-4.clq | 8 | 0,2746 | 0,0144 | **0,0143** | 0,0263 | 0,0149 | 0,0157 |
| johnson32-2-4.clq | 16 | $> 2h$ | 1414,16 | $> 2h$ | $> 2h$ | $> 2h$ | **0,9777** |
| keller4.clq | 11 | 0,0173 | 0,015 | 0,0133 | **0,0126** | 0,0615 | 0,096 |
| keller5.clq | 27 | $> 2h$ | **111,1303** | 575,7467 | 586,0393 | 2847,3533 | $> 2h$ |
| keller6.clq | | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| p_hat700-2.clq | 44 | **8,9725** | 193,2757 | 45,3087 | 49,9243 | $> 2h$ | $> 2h$ |
| san200_0,7_1.clq | 30 | **0,0051** | 0,5523 | 0,1066 | 0,1413 | 131,939 | $> 2h$ |
| san200_0,7_2.clq | 18 | **0,0071** | 1,1599 | 0,0448 | 0,0579 | 150,9787 | $> 2h$ |
| san200_0,9_1.clq | 70 | **0,056** | $> 2h$ | 6,0255 | 0,8951 | 3776,9667 | $> 2h$ |
| san200_0,9_2.clq | 60 | **0,5803** | $> 2h$ | 55,4346 | 11,1339 | 5563,0067 | $> 2h$ |
| san200_0,9_3.clq | 44 | **2,4223** | $> 2h$ | 430,7477 | 68,4938 | $> 2h$ | $> 2h$ |
| sanr200_0,9.clq | 42 | **34,4408** | $> 2h$ | 367,9413 | 123,365 | $> 2h$ | $> 2h$ |
| sanr400_0,5.clq | 13 | **0,6146** | 0,9591 | 0,8511 | 1,0063 | 23,5299 | 23,8989 |
| sanr400_0,7.clq | 21 | **124,305** | 377,9733 | 204,526 | 215,4133 | $> 2h$ | $> 2h$ |

### 4.2.2 Graf podobnosti sosesk in graf ujemanja z znanimi referenčnimi sekvencami

Na vseh grafih podobnosti sosesk in na grafu zadetkov na znane sekvence je algoritem Mcqd delal hitreje. Gtc-ju v dveh urah ni uspelo najti maksimalne klike na nobenem grafu (preglednica 8). Maksimalne klike pri mejni vrednosti večji od 0,55 ni uspelo najti niti Mcqd-ju, zato maksimalne klike na grafih z večjo mejno vrednostjo nismo iskali.

Preglednica 8: Primerjava časovne zahtevnosti med algoritmoma na grafih podobnosti sosesk in na grafu ujemanja z znanimi referenčnimi sekvencami.

Table 8: Comparison of time complexity between algorithms on similarity graphs and on realtion graph.

| | | | Gtc | | | | |
|---|---|---|---|---|---|---|---|
| *G* | *c_max* | Mcqd | 0,50 | 0,75 | L1 | L4 | L16 |
| blast.clq | 454 | **20,90** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.05,clq | 3 | **18.45** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.1.clq | 3 | **18,77** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.15.clq | 4 | **19,26** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.2.clq | 5 | **19,21** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.25.clq | 5 | **19,33** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.3.clq | 7 | **19,09** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.35.clq | 12 | **19,20** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.4.clq | 17 | **19,35** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutofff_0.45.clq | 46 | **19,66** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutofff_0.5.clq | 167 | **40,94** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.55.clq | 449 | **651,21** | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.6.clq | | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |
| cutoff_0.65.clq | | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ | $> 2h$ |

### 4.2.3 Proteinski produktni grafi

Na vseh proteinskih produktnih grafih je Mcqd našel maksimalno kliko veliko hitreje kot Gtc (preglednica 9). Tako na grafih DIMACS kot na proteinskih produktnih grafih, je iskanje maksimalne klike z iskanjem simetrij najhitrejše, če računamo simetrije do prvega nivoja (L1).

Preglednica 9: Primerjava časovne zahtevnosti med algoritmoma na proteinskih produktnih grafih.
Table 9: Comparison of time complexity between algorithms on protein product graphs.

| | | | Gtc | | | | |
|---|---|---|---|---|---|---|---|
| *G* | *c_max* | Mcqd | 0,50 | 0,75 | L1 | L4 | L16 |
| 1KZKA_3KT2A_78 | 247 | **0,0525** | 2839,5533 | 4,9392 | 0,2227 | 52,6234 | $> 2h$ |
| 1allA_3dbjC_41 | 346 | **0,2054** | $> 2h$ | 78,8108 | 2,5879 | $> 2h$ | $> 2h$ |
| 1f82A_1zb7A_5 | 500 | **0,6751** | $> 2h$ | $> 2h$ | 51,6496 | $> 2h$ | $> 2h$ |
| 2FDVC_1PO5A_83 | 556 | **1,1329** | $> 2h$ | $> 2h$ | 116,9097 | $> 2h$ | $> 2h$ |
| 2UV8I_2J6IA_13107 | 69 | **0,0111** | $> 2h$ | 9,8055 | 0,3407 | 2029,73 | $> 2h$ |
| 2W00B_3H1TA_10858 | 143 | **0,5155** | $> 2h$ | $> 2h$ | 8,6746 | $> 2h$ | $> 2h$ |
| 2W4JA_2A2AD_0 | 447 | **0,4286** | $> 2h$ | 1710,3567 | 7,9972 | $> 2h$ | $> 2h$ |
| 3HRZA_2HR0A_476 | 563 | **5,796** | $> 2h$ | $> 2h$ | 844,5033 | $> 2h$ | $> 2h$ |
| 3P0KA_3GWLB_0 | 89 | **0,0021** | $> 2h$ | 2,1145 | 0,0308 | 70,4099 | $> 2h$ |
| 3ZY0D_3ZY1A_110 | 52 | **0,0003** | 0,0299 | 0,0251 | 0,0031 | 0,0222 | 0,3374 |

### 4.2.4 Primerjalni grafi

Na vseh primerjalnih grafih je iskanje maksimalne klike s pomočjo simetrij veliko hitrejše kot samo z Mcqd. Na izbranih primerjalnih grafih se je izkazalo, da je smiselno iskati simetrije na več nivojih (preglednica 10). V teh primerih so bili optimalni štirje nivoji.

Preglednica 10: Primerjava časovne zahtevnosti med algoritmoma na primerjalnih grafih.
Table 10: Comparison of time complexity between algorithms on comparability graphs.

| | | | Gtc | | | | |
|---|---|---|---|---|---|---|---|
| *G* | *c_max* | Mcqd | 0,50 | 0,75 | L1 | L4 | L16 |
| sim_1567_249281 | 29 | $> 2h$ | 3308,01 | 3528,01 | 3430,64 | **523,23** | 528,36 |
| sim_528_51954 | 31 | $> 2h$ | 1259,12 | 1396,81 | 1300,07 | **205,657** | $> 2h$ |
| sim_755_57086 | 30 | 453,15 | 50,53 | 51,98 | 52,37 | **8,8905** | 13,3718 |
| sim_998_101738 | 21 | 406,30 | 26,21 | 151,42 | 28,31 | **0,39** | 0,4 |
| sim_998_104321 | 21 | 300,17 | 7,84 | 8,04 | 8,22 | **0,30** | 0,31 |
| sim_998_99631 | 29 | $> 2h$ | 2125,15 | 1958,54 | 2272,21 | **304,37** | 317,59 |

## 4.3 GRUPIRANJE PODOBNIH SOSESK

Grupiranje sosesk v grafu podobnosti sosesk nad mejno vrednostjo 0,5 nismo testirali, saj združevanje sosesk z več kot 50 % različnostjo ne bi imelo smisla. Prav tako maksimalne klike v teh grafih nismo uspeli najti z nobenim algoritmom.

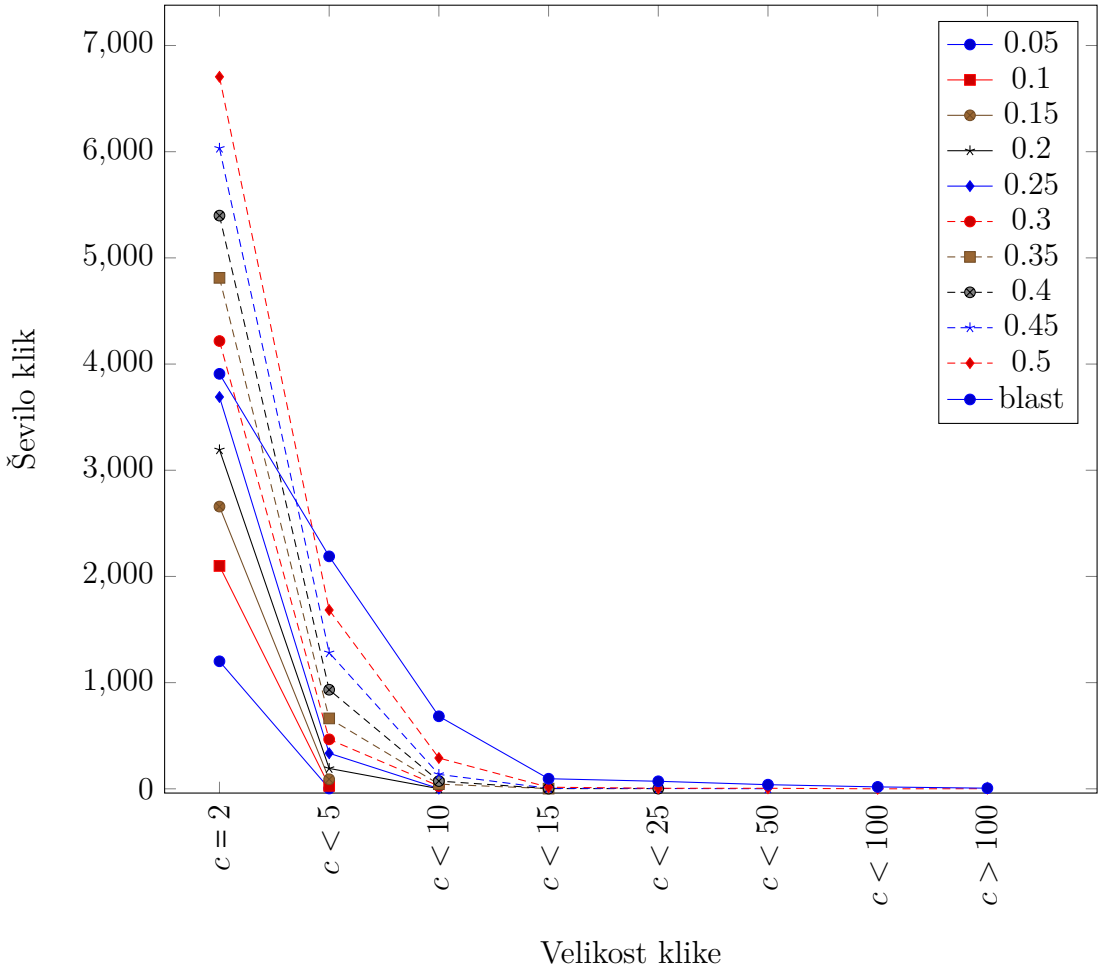
Klike smo, glede na njihovo velikost, razdelili v osem razredov (preglednica 11). Kot pričakovano je največ klik pri mejni vrednosti 0,5. Razpon velikosti klik se začne večati nad mejno vrednostjo 0,35. Do mejne vrednosti 0,3 so vse klike do velikosti 10. Največ klik pa je velikosti 2 oziroma do velikosti 3 (slika 21).

Preglednica 11: Število vseh klik in njihovo število glede na velikost v grafih podobnosti sosesk in v grafu ujemanja z znanimi referenčnimi sekvencami.
Table 11: The number of all cliques and their number depending on the size in similarity graphs and in relation graph.

| $G$ | Št. c | c max | $c = 2$ | $c < 5$ | $c < 10$ | $c < 15$ | $c < 25$ | $c < 50$ | $c < 100$ | $100 < c$ |
|---|---|---|---|---|---|---|---|---|---|---|
| cutoff_0.05.clq | 1204 | 3 | 1201 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.1.clq | 2127 | 3 | 2099 | 28 | 0 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.15.clq | 2748 | 4 | 2658 | 90 | 0 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.2.clq | 3386 | 5 | 3193 | 192 | 1 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.25.clq | 4027 | 5 | 3690 | 335 | 2 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.3.clq | 4709 | 7 | 4217 | 466 | 26 | 0 | 0 | 0 | 0 | 0 |
| cutoff_0.35.clq | 5521 | 12 | 4812 | 663 | 44 | 2 | 0 | 0 | 0 | 0 |
| cutoff_0.4.clq | 6408 | 17 | 5398 | 933 | 73 | 2 | 2 | 0 | 0 | 0 |
| cutoff_0.45.clq | 7460 | 46 | 6033 | 1282 | 136 | 5 | 3 | 1 | 0 | 0 |
| cutoff_0.5.clq | 8704 | 167 | 6705 | 1684 | 290 | 15 | 5 | 4 | 0 | 1 |
| blast.clq | 7009 | 454 | 3908 | 2189 | 683 | 95 | 71 | 39 | 18 | 6 |

Pri grafu ujemanja z znanimi referenčnimi sekvencami je največja klika 454, vendar je skupno število klik manjše kot pri grafih podobnosti sosesk z mejno vrednostjo 0,45 in 0,5, kar je posledica odstranitve večjega števila povezav pri odstranjevanju vozlišč. Prav tako je razpon klik večji kot pri grafih podobnosti sosesk.

Soseske v klikah smo poravnali s programom Clustal Omega in preverili, kakšno je njihovo prekrivanje. Zelo dobro so se prekrivale soseske v klikah, ki so bile najdene v grafu podobnosti sosesk z nižjo mejno vrednostjo. Na sliki 22 je izsek poravnave prve najdene klike v grafu z mejno vrednostjo 0,05. V kliko so bile združene ena daljša (HumLup_Tr-assembly-2015-06_10924) in dve krajši soseski (HumLup_Tr-assembly-2015-06_10923, HumLup_Tr-assembly-2015-06_10925). Daljša soseska je imela dolžino 1289 nukleotidov, krajši pa 251 oziroma 273 nukleotidov. Kot je razvidno, je




Slika 21: Prikaz števila klik glede na velikost v grafih podobnosti sosesk in v grafu ujemanja z znanimi referenčnimi sekvencami.

Figure 21: The number of cliques, depending on their size in similarity graphs and in the relation graph.

prekrivanje zelo dobro in ni odsekov, kjer bi se soseske zaporedoma razlikovale v več kot enem nukleotidu. Prav tako pri poravnavi ni nobene notranje vrzeli. Za soseske iz te klike bi lahko rekli, da pripadajo istemu transkriptu.

```
HumLup_Tr-assembly-2015-06_10924      GAGATTGTACTGCTGCTCTTAACCACCAAAAATCATCAACATTGGCCCAATTCCCAGTCT
HumLup_Tr-assembly-2015-06_10923      ------------TGCTCTTAACCACCGAAAATCATCAACATTGGCCCAATTCCCAGTCT
HumLup_Tr-assembly-2015-06_10925      ------------CTGCTCTTAACCACCGAAAATCATCGACATTGGCCCAATTCCCAGTCT
                                                 ************** ********* **********************

HumLup_Tr-assembly-2015-06_10924      CTTCAGCAAGGCTGGCCTCCTGAAGATCTCTATGAAACCCATCATAGCCCAAACAGTATG
HumLup_Tr-assembly-2015-06_10923      CTTCAGCAAGGCTGGCCTCCTGAAGATCTCTATGAATCCCATCATATCCCAAACAGTATG
HumLup_Tr-assembly-2015-06_10925      CTTCAGCAAGGCTGGCCTCCTGAAGATCTCTATGAATCCCATCATAGCCCAGACAGTATG
                                      ************************************ ********* **** ********

HumLup_Tr-assembly-2015-06_10924      GCGCAAACCTCACACCACTACTGTCCTCGATTATGGGCCTGCTCCTGACTCTAAGATAAA
HumLup_Tr-assembly-2015-06_10923      GCGCAAACCTCACACCACTACTGTCCTCGATTATGGGCCTGCTCCTGACTCTGAGATAAA
HumLup_Tr-assembly-2015-06_10925      GGGCAAACCTCACACCACTGCTGTCCTCGATTATGGGCCTGCTCCTGACTCTGAGATGAA
                                      * ***************** ******************************** **** **

HumLup_Tr-assembly-2015-06_10924      AATCAGTGGCCATGGCATGATGAATTCGAATCTGATGAGACGCCATTACGAAAACACACC
HumLup_Tr-assembly-2015-06_10923      AATCAGTGGCCTTGGCGTGATGAATTCGAATCTGATGAGACGCCATTACGAAAACACAAC
HumLup_Tr-assembly-2015-06_10925      AATCAGTGGCCTTGGCGTGATGAATTCGAATCTGATGAGACGCCATTACGAAAACACACC
                                      *********** **** ****************************************** *

HumLup_Tr-assembly-2015-06_10924      CCTCAACTTGCTCGAACAGAATTGAATTCATCACAGGACCTCCGTAGACTCGACAATTCC
HumLup_Tr-assembly-2015-06_10923      CCTCAACTTGCTCGAACAGAATTG----------------------------------
HumLup_Tr-assembly-2015-06_10925      CCTCAACTTGCTCGAACAGGATTGAATTCATCACAGGACCTCCGT---------------
                                      ****************** ****
```

Slika 22: Izsek poravnave sosesk iz prve maksimalne klike, ki je bila najdena v grafu podobnosti sosesk z mejno vrednostjo 0,05. Prekrivanje med soseskami je zelo dobro, saj ni nobene notranje vrzeli in ni odsekov, kjer bi se soseske zaporedoma razlikovale v več kot enem nukleotidu. Te soseske bi lahko pripadale istemu transkriptu.

Figure 22: Section of contigs alignment of first maximal clique in similarity graph with cutoff point 0.05. Overlapping of contigs is very good. There is no parts where contigs in succession differed in more than one nucleotide and there are no internal gaps. Those contigs could belong to the same transcriptome.

Na sliki 23 je prva maksimalna klika, ki je bila najdena v grafu podobnosti sosesk z mejno vrednostjo 0,35. V maksimalno kliko je bilo grupiranih 12 sosesk. Najdaljša je bila soseska HumLup_Tr-assembly-2015-06_12560 z dolžino 2348 nukleotidov. Najkrajša je pa bila soseska HumLup_Tr-assembly-2015-06_48580 z dolžino 215 nukleotidov. Mejna vrednost 0,35 je tudi prva zgornja mejna vrednost, kjer ni odseka s popolnim prekrivanjem med soseskami iz prve maksimalne klike. Na določenih delih poravnave so prisotne tudi notranje vrzeli. Glede na značilnosti poravnave soseske iz te klike verjetno ne pripadajo istemu transkriptu.


Slika 23: Izsek poravnave prve klike, ki je bila najdena v grafu podobnosti sosesk z mejno vrednostjo 0,35. Med soseskami ni popolnega prekrivanja. Na določenih delih poravnave so prisotne tudi notranje vrzeli. Soseske iz te maksimalna klike verjetno ne pripadajo istemu transkriptu.

Figure 23: Section of contigs alignment of first maximal clique in similarity graph with cutoff point 0.35. There is no overlap of contigs on any region. These contigs probably do not belong to the same transcriptome.

Največja maksimalna klika je bila najdena v grafu ujemanja z znanimi referenčnimi sekvencami (*blast.clq*), kjer prav tako ni bilo odseka s popolnim prekrivanjem med soseskami, kar velja tudi za manjše klike. Na sliki 24 je odsek poravnave 1693. najdene maksimalne klike. Poravnava je brez daljšega odseka s popolnim prekrivanjem med soseskami.

V tabeli 12 so štiri soseske med katerimi so izračunane razdalje s programom Clustal Omega. Soseska 362 je dolga 4960 baz in je najdaljša med njimi. Z vsemi soseskami se zelo lepo poravna, saj je podobnost več kot 90 %. Na grafu podobnosti sosesk z mejno vrednostjo od 0,09 do 0,76 bi te soseske tvorile induciran podgraf na sliki 25.

Preglednica 12: Matrika razdalj med štirimi sosekami.
Table 12: Distance matrix between four contigs.

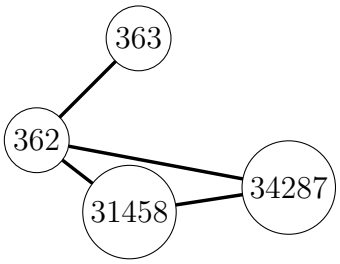
| soseska | HumLup-362 | HumLup-363 | HumLup-31458 | HumLup-34287 |
|---|---|---|---|---|
| HumLup-362 | 0 | 0,07 | 0,04 | 0,09 |
| HumLup-363 | 0,07 | 0 | 0,77 | 0,76 |
| HumLup-31458 | 0,04 | 0,77 | 0 | 0,10 |
| HumLup-34287 | 0,09 | 0,76 | 0,10 | 0 |

Z našo metodo dekompozicije grafa z maksimalno kliko so v 19. maksimalni kliki v grafu *cutoff_0.2.clq* bile soseske 362, 31458 in 34287, ki se med seboj poravnajo na odseku od 3255. do 3758. baze glede na sosesko 362 (slika 26). Po odstranitvi teh

```
HumLup_Tr-assembly-2015-06_52507      TCAGCGNNNNNNNNNNNNNNNNNNNNNNN------------NNNNNNNNNNNNNNNNN
HumLup_Tr-assembly-2015-06_49778      TCAGATCCGACGACCTCTTCCGGCACCATGTATCTCCCATCAGTTGGTGGATCTCATTCG
HumLup_Tr-assembly-2015-06_82070      TCATCTCTTTCGAAATCTCCGGCGGGTAGGAGTCTCCACCGATTACTGGAATCAGA----
                                      ***

HumLup_Tr-assembly-2015-06_52507      NNNNNNNNNGTTTTTGGAAGAAGAGTTTCGTGCTCTTCTCGAGGATTCAAAAAACGACAA
HumLup_Tr-assembly-2015-06_49778      ACCCTTTGATCTTTTGAACGATGTATCTCCCGTTGTTCATCAGGAACACGT-AGCTCAGT
HumLup_Tr-assembly-2015-06_82070      ----------------------TCACGGAGGGTCTCGTACATGTCGAGGA-ACTTAAAG
                                                           * * *     * *       *

HumLup_Tr-assembly-2015-06_52507      CAATAGTAAGGATTCATCATCATCATCATCCGATCACAAATCCAATGACAAGACTTCGAA
HumLup_Tr-assembly-2015-06_49778      GATGGGTCTCTGTACAGCTTCGACT---TCATCTCCAGATTCGAGTCCAA----------
HumLup_Tr-assembly-2015-06_82070      AGTTTCTCGGCGGAGCGCTTGGT----------------------TAGAA----------
                                            *         * *                        *   *

HumLup_Tr-assembly-2015-06_52507      ACAATCGTCGTTTAACAACAACAATCATGAATATTCCGATCGTTG----------TGTTA
HumLup_Tr-assembly-2015-06_49778      CAGATCCATCATCGACATCAGCTGTTTCGAAAACGGCGACGTTTGGGACTCCCCTGATTG
HumLup_Tr-assembly-2015-06_82070      CCACAGCCTCGGCGAAATCGAGAATCTTGACGAC--------------------------
                                                  * * *     *   **  *

HumLup_Tr-assembly-2015-06_52507      TAGGTGGTGAGTCCGATCAAAAACAATCTGATCAG-----------GCTGCAAAGGAAG
HumLup_Tr-assembly-2015-06_49778      ATCGTCCTGAGCCCACTGATGAACGTTAGGTTCAACGTGATGATCACTTTGATCGTCGTC
HumLup_Tr-assembly-2015-06_82070      -------------------------------AAACGCGCGCGCCATGTTGCTG------
                                                                     *             **
```

Slika 24: Izsek poravnave sosesk, ki so bile v kliki velikosti 3 najdene kot 1693. maksimalna klika v grafu *blast.clq*.

Figure 24: Section of contigs alignment of clique size 3, found as the 1693th clique in the graph*blast.clq*.




Slika 25: Induciran podgraf, ki bi ga tvorile soseske iz preglednice 12 na grafu podobnosti sosesk z mejno vrednostjo od 0,09 do 0,76.

Figure 25: Induced subgraph, which will be formed by contigs from table 12 in similarity) graph with cutoff point from 0.09 to 0.76.

vozlišč iz grafa je soseska 363 postala izolirano vozlišče, saj je imela edino povezavo s sosesko 362. Zaradi tega je algoritem ni mogel povezati s sosesko 362, kljub temu, da je medsebojno prekrivanje zelo dobro (slika 27), saj se poravnata v odseku od 32. do 466. baze. Glede na poravnavo, verjetno vse štiri soseske pripadajo istemu transkriptu. Sosesko 362 bi lahko obravnavali kot neke vrste reprezentančno sosesko.

```
HumLup_Tr-assembly-2015-06_00362      GGGGCTCGAAAAAGGGCTTGTGCTTGGTTGAACTATCCCCGCCTGTAGCATCTCAGTCAA
HumLup_Tr-assembly-2015-06_31458      GGGGCTCGAAAAAGGGCTTGTGCTTGGCTGAACTATCCCCGCTTGTAGCATCTCAGTCAA
HumLup_Tr-assembly-2015-06_34287      CGGGCTCGAAAAAGGACTTATGCTTGGTTGAACTATCCCCGCCTGTAGCATCTCAGTCAA
                                       ************** *** ******* ************** ******************

HumLup_Tr-assembly-2015-06_00362      CAACTTCTCTATCTCATCTTTTTGTATCTGCGCGTAGCGATATGGTCGAACTGAAATGGG
HumLup_Tr-assembly-2015-06_31458      CAACTTCTCTATCTCATCTTTTTGTATCTGTGCGTATTGGTATGGTCGAACTGAAATGGG
HumLup_Tr-assembly-2015-06_34287      CAACTTCTCTATCTCATCTTTTTGTATTTGTGCGTATCGGTATGGGCGAACCGAAATGGG
                                      *************************** ** *****  * ***** ***** ********

HumLup_Tr-assembly-2015-06_00362      ACCCGACCCCTCCCTCATGGTGATGTGATGCTCGTGGGAACGGTGTGGTGGCAGCCCTTT
HumLup_Tr-assembly-2015-06_31458      ACCCGACCCCTCCCTCATGGTGATGTGATGCTCGTGGGAACGGTGTGGTGGCAGCCCTTT
HumLup_Tr-assembly-2015-06_34287      ACCCGACCCCTCCTTCATGGTGATGTGGTGCTCGTGGGAACGATGTGGTGGCAACCCTTT
                                      ************* ************* ************** ********** ******

HumLup_Tr-assembly-2015-06_00362      CGGCATGTCGAAAACAGCTTCATGTTTCTGCAGCAATTCTTGCATTTCAGCCTCATTTTC
HumLup_Tr-assembly-2015-06_31458      CGGCATGTCGAAAACAGCTTCATGTTTCTGCAGCAGTTCTTGTATTTCAGCCTCATTGTC
HumLup_Tr-assembly-2015-06_34287      CGGCATGTCGAAAACTGCCTCGTGTTTCTGAAGTAATTCTTGCATTTCAGCCCCATTCTC
                                      *************** ** ** ******** ** * ****** ********* **** **
```

Slika 26: Izsek poravnave 19. maksimalne klike v grafu podobnosti sosesk z mejno vrednostjo 0,20.

Figure 26: Section of contigs alignment of clique found as the 19th clique in the similarity graph with cutoff point 0.20.

```
HumLup_Tr-assembly-2015-06_00362      TATTCGCGGTGGTCTGAAATCGCGAGGCATCTGTGATTCGGGTGGTGCATAAGGAGGCCC
HumLup_Tr-assembly-2015-06_00363      TATTCGCGGTGGTCTGAAATCGCGAGGCAGCTGTGAGTCGGGTGGTGCATAAGGATGCCC
HumLup_Tr-assembly-2015-06_31458      ------------------------------------------------------------
HumLup_Tr-assembly-2015-06_34287      ------------------------------------------------------------

HumLup_Tr-assembly-2015-06_00362      CGTCGACGTCGCTGGGGGTGGGTGCATCGCCGAGGGAGTCCCCGTTGGGCGATGAGAGGA
HumLup_Tr-assembly-2015-06_00363      CGTCGAAGTCGCTGGGGGTGGGTGCATCGCCGAGGGAGTCCCCGTAGGTGGATGAGAGGA
HumLup_Tr-assembly-2015-06_31458      ------------------------------------------------------------
HumLup_Tr-assembly-2015-06_34287      ------------------------------------------------------------

HumLup_Tr-assembly-2015-06_00362      GGAGCTGTCGCCGTCGACCGCCTGCTTCTGTCGATCTTCCGTCGCACTGGTACCCGCCAT
HumLup_Tr-assembly-2015-06_00363      GGAGCTGTCGCCATCGACCGCTTGCTTCTGTCGATCGTCCGTCGCACTCAGACCCGCCAT
HumLup_Tr-assembly-2015-06_31458      ------------------------------------------------------------
HumLup_Tr-assembly-2015-06_34287      ------------------------------------------------------------
```

Slika 27: Izsek poravnave med soseskami iz preglednice 12.

Figure 27: Section of contigs alignment of contigs from table 12.

# 5 RAZPRAVA IN SKLEPI

## 5.1 RAZPRAVA

### 5.1.1 Primerjava algoritmov za iskanje maksimalne klike

Algoritma Mcqd in Gtc smo primerjali na štirih različnih vrstah grafov. Gtc algoritem se je izkazal za hitrejšega le na primerjalnih grafih. Z vidika kasnejšega raziskovanja nas je zanimala predvsem hitrost iskanja maksimalne klike na grafu podobnosti sosesk in na grafu ujemanja z znanimi referenčnimi sekvencami, ki smo jih uporabili za združevanje sosesk. Na vseh teh grafih je Mcqd poiskal maksimalno kliko hitreje kot Gtc, zato smo za dekompozicijo grafa z maksimalno kliko priredili program Mcqd, ki sta ga napisala Konc in Janežič (2007).

Gtc algoritem poskuša iskalni prostor omejiti z iskanjem simetrij, zato je njegova uspešnost odvisna predvsem od tipa grafov oziroma od simetrij, ki jih imajo grafi in inducirani podgrafi. Na prvi stopnji so vsi grafi za združevanje sosesk imeli simetrije. Podobno so na različnih bioloških, sociooloških in drugih grafih iz vsakodnevnega življenja odkrili tudi MacArthur in sod. (2008). En izmed razlogov za počasno delovanje Gtc je v velikem številu izoliranih vozlišč v grafih, saj posledično potrebuje program *bliss-0.72* veliko časa za računanje generatorjev grupe $Aut(G)$. Kasneje smo v nekaterih grafih odstranili izolirana vozlišča in testiranje s programom Gtc ponovili (preglednica 13). Graf *cutoff_45.clq* je po odstranitvi vozlišč imel velikost $|V(G)| = 24795$. Maksimalno kliko smo, za $N = |V(G)| \cdot 0{,}1$ in $nivo = 5$, našli v 79 sekundah. Podobno je bilo tudi pri grafu *blast.clq*. Njegova velikost po odstranitvi izoliranih vozlišč je bila $|V(G)| = 26996$, maksimalna klika pa je bila, za $N = |V(G)| \cdot 0{,}005$ in $nivo = 1$, najdena v 93 sekundah. Kot je razvidno, je prišlo do občutne izboljšave delovanja programa Gtc, saj pred odstranitvijo izoliranih vozlišč za omenjena grafa maksimalne klike z Gtc nismo našli (preglednica 8).

Testiranje smo ponovili še na grafu *cutoff_55.clq*, vendar izboljšave nismo dosegli. Pogledali smo, koliko povezanih komponent imajo grafi. Povezana komponenta na neusmerjenem grafu je podgraf, kjer so vsa vozlišča medsebojno povezana s potjo. Graf *cutoff_55.clq* je imel 4465 povezanih komponent. Večina jih je bila velikosti 2 in

Preglednica 13: Velikost grafov in čas potreben za iskanje maksimalne klike po odstranitvi izoliranih vozlišč.
Table 13: Size of the graphs and time needed to find the maximal clique after removal of isolated vertices.

| $G$ | $\|V(G)\|$ | $N$ | nivo | čas (s) |
|---|---|---|---|---|
| cutoff_45.clq | 24795 | $\|V(G)\|\cdot 0{,}1$ | 5 | 79 |
| blast.clq | 26996 | $\|V(G)\|\cdot 0{,}005$ | 1 | 93 |

3, največja je imela velikost 22394, druga največja pa velikost 32. Podgraf iz največje komponente je na prvem nivoju imel 18939 orbit. Od tega jih je bilo le 1199 z več kot enim vozliščem, povprečna stopnja teh je bila 3,1. Prav tako pa je bilo kar 3153 orbit z enim vozliščem, ki so imela stopnjo $\geq 500$. Zato, kljub odstranitvi izoliranih vozlišč v tem grafu, iskanje simetrij v grafu pri iskanju maksimalne klike zahteva le dodaten čas.

### 5.1.2 Grupiranje podobnih sosesk

V grafu ujemanja z znanimi referenčnimi sekvencami smo našli največjo maksimalno kliko, prav tako sta bila razpon klik in njihova velikost največja. Soseske iz klik smo poravnali s programom Clustal Omega in nato pregledali, kako dobro se soseske med seboj prekrivajo (slike 22, 23, 24). Za graf ujemanja z znanimi referenčnimi sekvencami je nemogoče določiti, pri kateri velikosti klike se soseske lepo poravnajo. Tudi večina sosesk pri kliki velikosti 3 (slika 24) je popolnoma neskladnih, kar lahko pojasnimo z delovanjem algoritma BLASTx, ki je namenjen lokalni poravnavi. V kliki so lahko sekvence, ki se med seboj ne prekrivajo, vendar se posamezna sekvenca prekriva na različnem delu tarčnega proteina. Zato povratno preverjanje klik z globalno poravnavo na grafu zgrajenem z BLASTx-om, ne moremo uporabiti za ocenjevanje pravilnosti rezultata.

Problem lokalne poravnave je tudi velikost klik, ki smo jih dobili v grafu *blast.clq*, saj so le-te prevelike. Glede na njihovo velikost menimo, da je v grafu med soseskami veliko povezav, ki v resnici ne pripadajo istemu genu. Prevod sekvence na šest različnih bralnih okvirjev in lažja poravnava krajših sekvenc s sekvencami v bazi, povečuje število parov sekvenca–protein. Kljub temu, da vrednost $E = 0,001$ predstavlja mejo, pri

kateri BLASTx vrne le zelo kvalitetne povezave sekvenca–protein (Wheeler in Bhagwat, 2007), menimo, da bi morali za lokalno poravnavo določiti nižjo vrednostjo $E$.

Z iskanjem maksimalne klike v grafu podobnosti sosesk smo želeli določiti mejno vrednost, pri kateri je mogoče grupirati soseske, ki pripadajo istemu transkriptu. Po pregledu poravnav, ki smo jih naredili med soseskami znotraj klike, smo za zgornjo mejno vrednost določili 0,2, kar ustreza vsaj 80 % podobnosti med soseskami znotraj klike. Nad to mejno vrednostjo so bile v klikah soseske, ki so pri poravnavi imele na daljših odsekih različne nukleotide in vstavljene daljše vrzeli (slika 23).

Edini problem pri uporabljeni metodi dekompozicije grafa z maksimalno kliko na grafu podobnosti sosesk, so različno dolge soseske, ki se dobro poravnajo z daljšo skupno sosesko, med seboj pa se ne prekrivajo (preglednica 12). Glede na to, da induciran podgraf (slika 25) predstavlja povezano komponento na grafu podobnosti z mejno vrednostjo 0,2, nas je zanimalo, ali bi lahko omenjeni problem rešili z iskanjem povezanih komponent. V preglednici 14 je število povezanih komponent glede na njihovo velikost, ki smo jih našli v grafu podobnosti sosesk z mejno vrednostjo 0,1. Skupno število povezanih komponent je manjše kot skupno število klik na tem grafu (preglednica 11), vendar se je povečalo število grup, s 3 do 4 soseskami in število grup, s 5 do 9 soseskami, zato se je skupno število grupiranih sosesk povečalo.

Preglednica 14: Število vseh povezanih komponent in njihovo število glede na velikost v grafu podobnosti sosesk z mejno vrednostjo 0,1.
Table 14: The number of all connected components and their number according to the size in similarity graph with cutoff point 0.1.

| $G$ | Št. p. komp. | komp. max | $komp. = 2$ | $komp. < 5$ | $komp. < 10$ |
|---|---|---|---|---|---|
| cutoff_0.1.clq | 2110 | 7 | 1975 | 125 | 10 |

Po pregledu poravnav je več povezanih komponent imelo eno daljšo in več krajših sosesk. Kvaliteta poravnav z več kot 4 soseskami v povezani komponenti je bila zelo različna. Poravnave so imele notranje vrzeli, na določenih odsekih pa so se soseske razlikovale v več kot enem nukleotidu. Poravnave sosesk iz manjših komponent so bile bolj kvalitetne. V preglednici 15 so izračunane razdalje med soseskami iz povezane komponente velikosti 3. Najdaljša je soseska 17783 z dolžino 676 nukleotidov in je zelo podobna ostalima dvema. Soseski 15512 in 17784 pa sta si zelo različni.

Preglednica 15: Matrika razdalj med soseskami v povezani komponenti v grafu podobnosti sosesk z mejno vrednostjo 0,1.

Table 15: Distance matrix of contigs in connected component in similarity graph with cutoff point 0.1.

| soseska | HumLup-15512 | HumLup-17783 | HumLup-17784 |
|---|---|---|---|
| HumLup-15512 | 0 | 0,05 | 0,76 |
| HumLup-17783 | 0,05 | 0 | 0,03 |
| HumLup-17784 | 0,76 | 0,03 | 0 |

Izsek poravnave med soseskami iz preglednice 15 je prikazan na sliki 28. Soseski 17783 in 17784 se medsebojno poravnata na odseku od 12. do 231. baze glede na sosesko 17783. Soseski 15512 in 17783 se medsebojno poravnata na odseku od 207. do 426. baze glede na sosesko 17783. Vse tri pa se medsebojno poravnajo v odseku od 207. do 231. baze glede na sosesko 17783. Soseske iz te komponente bi lahko pripadale istemu transkriptu.


Slika 28: Izsek poravnave med soseskami iz preglednice 15.

Figure 28: Section of contigs alignment from table 15.

## 5.2 SKLEPI

### 5.2.1 Primerjava algoritmov za iskanje maksimalne klike

Iskanje maksimalne klike spada v množico NP problemov (Karp, 1972), kar pomeni, da najverjetneje ne obstaja polinomski algoritem za reševanje tega problema. Izbira

algoritma bo zato toliko bolj odvisna od tipa oziroma od značilnosti grafa, kar smo tudi potrdili s testiranjem algoritmov na različnih grafih. Na primerjalnih grafih (preglednica 10) smo maksimalno kliko našli najhitreje z algoritmom Gtc. Na večini ostalih grafov (preglednice 7, 8, 9) pa se je izkazalo, da se maksimalno kliko veliko hitreje najde, če za omejitve iskanja uporabimo le barvanje grafov. Kljub temu, da ima graf podobnosti sosesk in graf ujemanja z znanimi referenčnimi sekvencami simetrije, so le-te takšne (majhne orbite in vozlišča z nizko stopnjo), da zahteva njihovo iskanje dodaten čas pri iskanju maksimalne klike.

Čeprav je iskanje maksimalne klike z algoritmom Gtc večinoma počasnejše, smo mnenja, da je njegova uporaba smiselna, saj lahko vedno deluje enako hitro kot algoritem Mcqd. Z nastavitvijo $N = 0$ in $nivo = 0$ algoritem Gtc takoj prične iskati maksimalno kliko z algoritmom Mcqd. V prihodnosti lahko pričakujemo povečanje količine podatkov, zato bo možna sprememba značilnosti grafov, s katerimi bomo poskušali reševati biološke probleme.

Algoritem Gtc je možno še izboljšati:

- Testirana verzija algoritma Gtc je trenutno maksimalno kliko hranila le lokalno in jo za primerjavo pošiljala v Mcqd. Z globalno hrambo maksimalne klike lahko izboljšamo delovanje algoritma Gtc, ker ni potrebno obravnavati orbit z vozlišči, ki imajo manjšo stopnjo od trenutno najdene maksimalne klike (slika 16).
- Algoritem Gtc vedno izbere vozlišče iz največje orbite. Nekateri grafi imajo veliko simetričnih vozlišč, ki tvorijo velike orbite, vendar imajo ta vozlišča velikokrat majhno stopnjo. Namesto izbire vozlišča iz velike orbite, lahko izberemo vozlišče iz najmanjše in dobimo velik induciran podgraf ter najprej poiščemo veliko kliko. V določnih primerih lahko to, skupaj z globalno hrambo maksimalne klike, hitreje omeji iskalni prostor.
- Algoritem Gtc za iskanje simetrij uporablja algoritem *bliss-0.72* (Junttila in Kaski, 2007). Izkazalo se je, da ta algoritem zelo slabo deluje na grafih z veliko izoliranimi vozlišči, kar je najverjetneje posledica napake v algoritmu. Algoritem Gtc je zato možno izboljšati z vključitvijo ostalih algoritmov za iskanje simetrij, kot je na primer *nauty* (McKay in Piperno, 2014).

### 5.2.2 Grupiranje podobnih sosesk

Na grafu podobnosti sosesk in grafu ujemanja z znanimi referenčnimi sekvencami smo z metodo dekompozicije grafa z maksimalno kliko želeli grupirati soseske, ki pripadajo istemu transkriptu oziroma genu. Analiza poravnav znotraj klik v grafih podobnosti sosesk je pokazala, da metoda dekompozicije grafa z maksimalno kliko omogoča grupiranje sosesk, ki pripadajo istemu transkriptu. Nižja kot je mejna vrednost, bolj zanesljivo soseske iz klike pripadajo istemu transkriptu. Na podlagi pregleda poravnav se je za zgornjo mejno vrednost izkazala vrednost $\leq 0{,}2$. Večinoma so se pri tej mejni vrednosti soseske iz klike na daljših odsekih razlikovale le v posameznem nukleotidu in so bile poravnane brez notranjih vrzeli.

Za nadaljnjo uporabo tega pristopa na grafu podobnosti sosesk priporočamo spremembo algoritma dekompozicije grafa z maksimalno kliko, saj se zaradi odstranitve vozlišč iz grafa, ki so bile v kliki, določene soseske med seboj ne grupirajo. Daljše soseske, ki se prekrivajo s krajšimi soseskami na različnih delih (slika 25), bodo povezane le s soseskami s katerimi so bile združene v prvi najdeni kliki, zato predlagamo sledečo definicijo dekompozicije grafa. Naj bo $G$ nek graf. Zaporedju podgrafov $H_1, \ldots, H_n$ grafa $G$ pravimo dekompozicija na maksimalne klike, če velja:

1. $E(H_i) \cap E(H_j) = \emptyset$ za vsak $1 \leq i < j \leq n$
2. $E(H_1) \cup \cdots \cup E(H_k) = E(G)$
3. $H_i$ *je maksimalna klika v* $G - (E(H_1) \cup \cdots \cup E(V_{i-1}))$ za vsak $1 \leq i < j \leq n$

Nov algoritem za dekompozicijo grafa z maksimalno kliko je na sliki 29.

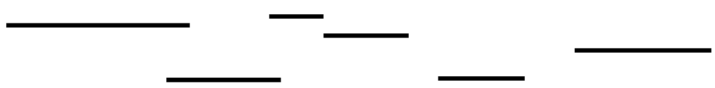
```
1: AllClq := ∅
2: cl := computeMaxClq(G)
3: while |cl| > 1 do
4:     AllClq ← cl
5:     for i := 0 do |cl| do
6:         removeEdge(G, cl, i)
7:     cl := computeMaxClq(G)
```

Slika 29: Spremenjen algoritem dekompozicije grafa z maksimalno kliko.
Figure 29: New algorithm of graph decomposition based on the maximum clique method.

Število klik se bo povečalo, saj se bodo nekatere soseske pojavile v več klikah, nato naj se združijo klike z enakimi soseskami.

Glede na to, da v klikah lahko obstajajo reprezentativne soseske in so grafi zelo redki, je mogoče problem grupiranja sosesk na grafih podobnosti sosesk rešiti z iskanjem povezanih komponent. Prednost tega pristopa bi bil tudi čas, saj je časovna zahtevnost algoritmov za iskanje povezanih komponent manjša kot pri algoritmih za iskanje maksimalne klike.

Z grafom ujemanja z znanimi referenčnimi sekvencami nismo dobili želenih rezultatov, vendar menimo, da bi le-te dobili s popravkom kriterijev pri izgradnji grafa. S programom Clustal omega je mogoče združiti le soseske, ki se med seboj prekrivajo, zato bi lahko z BLASTx-om in metodo dekompozicije grafa z maksimalno kliko grupirali soseske, ki medsebojnega prekrivanja nimajo oziroma se prekrivajo le na koncih (slika 30).



Slika 30: Soseske, ki se med seboj ne prekrivajo oziroma se prekrivajo le na koncih.
Figure 30: Overlapping contigs and contigs, overlapping only at the ends.

Možnosti za izboljšavo tega pristopa so sledeče:

- Uporaba strožje meje (nižja pričakovana vrednost $E$) za določanje povezave med soseskama pri gradnji grafa ujemanja z znanimi referenčnimi sekvencami.
- Dekompozicije grafa z maksimalno kliko na združenem grafu podobnosti sosesk in grafu ujemanja z znanimi referenčnimi sekvencami.
- Zmanjšanje števila vhodnih podatkov za BLASTx. Obravnavati le soseske, ki niso v nobeni kliki in reprezentativne (najdaljše) soseske iz vsake klike v grafu podobnosti sosesk.

Z metodo dekompozicije grafa z maksimalno kliko na grafih podobnosti sosesk je mogoče grupirati podobne soseske in na ta način zmanjšati število sosesk pridobljenih z *de novo* določitvijo nukleotidnega zaporedja.

# 6 POVZETEK (SUMMARY)

## 6.1 POVZETEK

Doba sekvenciranja DNA se je začela leta 1977 s kemično metodo Maxama in Gilberta ter Sangerjevo dideoksi metodo (Hutchison, 2007). V zadnjem času se za določanje nukleotidnega zaporedja vedno več uporablja metode naslednje generacije, katerih prednosti so predvsem cena izvedbe, čas in količina pridobljenih podatkov. (Shendure in Ji, 2008). Med te metode uvrščamo tudi RNA-seq, prednost katere je natančna lokalizacija transkriptnih območji na en nukleotid, analiza kompleksnih transkriptomov in direktno določanje cDNA (Wang in sod., 2009).

Za združevanje odčitkov se v osnovi uporablja dva algoritma, OLC in združevanje z de Bruijnovim grafom. Slednji daje boljše in hitrejše rezultate pri združevanju kratkih odčitkov (Li in sod., 2012). Glavni problem združevanja odčitkov je, da se tvori večje število sosesk, kot je genov. Zato poskušajo programi z različnimi metodami grupirati tiste soseske, ki pripadajo istemu genu.

Pri analizi podatkov si velikokrat pomagamo tako, da problem modeliramo s pomočjo matematične strukture grafov. V bioinformatiki s pomočjo grafov določajo sekundarno strukturo RNA (Ji in sod., 2004), iščejo povezave med geni in biološkimi pojavi (Matsunaga in sod., 2009; Rokhlenko in sod., 2007) in analizirajo proteinske strukture (Dukka in sod., 2006). Vsem omenjenim problemom je skupno, da se prevedejo na iskanje maksimalne klike v grafu.

V prvem delu raziskave smo med seboj primerjali algoritma Mcqd (Konc in Janežič, 2007) in Gtc (Azarija in Marc, 2015; Azarija in Slana, v pripravi). Gtc je na novo razvit algoritem, ki pri iskanju maksimalne klike upošteva simetrije v grafu. V prvem delu naloge smo raziskovali, ali se z upoštevanjem simetrije grafov lahko izboljša in pohitri obstoječe algoritme. Na večini testiranih grafov upoštevanje simetrij za iskanje maksimalne klike ne prinese izboljšanja, saj, zaradi značilnosti grafov, računanje simetrij zahteva le dodaten čas. Kljub temu menimo, da je uporaba algoritma Gtc smiselna, saj se lahko, glede na značilnosti grafa, odločimo, ali bomo pri iskanju maksimalne klike upoštevali simetrije. Posledično, algoritem Gtc deluje vedno enako hitro

kot Mcqd.

V drugem delu raziskave smo z metodo dekompozicijo grafa z maksimalno kliko poskušali grupirati soseske, ki pripadajo istemu transkriptu. Iz sosesk RNA hmelja (*Humus lupulus L.*) smo zgradili dve vrsti grafov. Prvi je bil graf podobnosti sosesk, ki je bil zgrajen iz polnega utežnega graf tako, da smo odstranili povezave, ki so imele večjo utežno vrednost od določene. Vrednost uteži je bila določena na podlagi podobnosti med soseskami. Drugi graf smo zgradili glede na ujemanje z znanimi referenčnimi sekvencami tako, da smo vozlišči povezali, če jih je BLASTx povezal z isto znano sekvenco. Soseske, združene v maksimalni kliki, smo poravnali s programom Clustal Omega (Sievers in sod., 2011) in pogledali, kako dobro se med seboj prekrivajo. Glede na prekrivanje sosesk v maksimalni kliki menimo, da je, z metodo dekompozicije grafa podobnosti sosesk z maksimalno kliko, mogoče grupirati soseske iz istega transkripta, če je zgornja mejna vrednost pri izgradnji grafa $\leq 0{,}2$, kar ustreza vsaj 80 % podobnosti med soseskami. Ugotovili smo, da so značilnosti grafov pod mejno vrednostjo $\leq 0{,}2$ takšne, da bi bilo problem grupiranja mogoče bolje reševati z iskanjem povezanih komponent. Prednost tega pristopa bi bila večja učinkovitost, saj imajo algoritmi za iskanje povezanih komponent manjšo časovno zahtevnost. Izognili bi se tudi problemu, ko v kliki obstaja daljša reprezentančna soseska in se na različnih delih poravna s krajšimi soseskami (slika 25).

Na grafu ujemanja z znanimi referenčnimi sekvencami nismo dobili želenih rezultatov, saj so bile maksimalne klike prevelike. Pri izgradnji grafa bi morali uporabiti tesnejše meje za določanje povezav med soseskama (nižjo pričakovano vrednost $E$). V prihodnosti bi bilo smiselno narediti dekompozicijo grafa z maksimalno kliko v združenem grafu iz grafa podobnosti sosesk in iz grafa presekov zadetkov na znane sekvence.

## 6.2 SUMMARY

In 1977, two modern methods of DNA sequencing were developed. Maxam and Gilbert developed the chemical method, while Sanger developed the dideoxi method for DNA sequencing (Hutchison, 2007). Recently, methods of next-generation sequencing have become increasingly popular for being time efficient and cost effective (Shendure in Ji,

2008). One of the next generation methods is also RNA-seq, the benefits of which are revealing the precise location of transcription boundaries to a single-base resolution, the possibility for studying complex transcriptome and directly determining the cDNA sequence (Wang in sod., 2009).

The OLC and de Bruijn graph assembly are two major algorithms for assembly, the later one being more suitable for high-coverage short reads. Neither of the two algorithms can assembly the whole genome, only individual reads into contigs (Li in Homer, 2010). The main problem is that many contigs are produced per gene, which is why different programs use different methods to cluster contigs belonging to the same gene.

Graph theory can be a good tool for modeling a problem and analyzing the data. Apart from being useful for reads assembly, graph theory is also successful at predicting the common RNA secondary structure motif (Ji in sod., 2004) to find relations between genes and biological phenomena (Matsunaga in sod., 2009; Rokhlenko in sod., 2007) and to analyze protein structure (Dukka in sod., 2006). All the aforementioned problems are reduced to the problem of finding a maximal clique in a graph. In the first part of the thesis, we compared the Mcqd (Konc in Janežič, 2007) and Gtc (Azarija in Marc, 2015; Azarija in Slana, v pripravi) algorithm. Gtc is a new algorithm that we developed. Gtc considers graph symmetries when searching for maximum clique. In our thesis we tried to find out if taking graph symmetries into account could improve in the current algorithms for finding a maximum clique on our instances. On four different types of graphs tested, finding symmetries did not result in faster run times. Nevertheless, Gtc algorithm can always skip the symmetry finding phase, making it as fast as Mcqd.

In the second part of the thesis we clustered contigs using graph decomposition based on the maximum clique method. From *Humus lupulus L.* RNA contigs, two types of graph were constructed. The first type was undirected weighted graph constructed from distance matrix obtained by the program Clustal omega (Sievers in sod., 2011). Then, depending on the similarity cutoff point, different similarity graphs were formed from this complete graph. The second type of graph was a relation graph based on the BLASTx results. Two vertices were adjacent if two contigs were aligned to the same protein sequence by BLASTx. Contigs which were clustered into a maximum

clique were aligned with program Clustal omega. With respect to the visualization of alignments, graph decomposition into maximum clique could be a useful method for clustering contigs belonging to the same transcriptome on similarity graph if the cutoff point is $\leq 0.2$. This corresponds to at least 80 % of similarity between contigs. Similarity graphs at cutoff point $\leq 0.2$ are sparse and contain a lot of small connected components. According to these properties, the clustering problem could be solved more easily by simply finding all the connected components. Algorithms for finding connected components are in general more efficient than the corresponding problem of finding maximal cliques. In addition, we could avoid the problem arising when some longer contig have a good similarity with a small contig on different part of its length (figure 25).

With relation graph (BLASTx), we did not get adequate results because all maximum cliques were to big. The main reason for this is that the query parameter for expectation value (0.001) was not low enough.

A problem to consider for further work would be constructing a single graph from the similarity and relation graph to perform decomposition based on the maximum clique method.

# 7 VIRI

Azarija J., Marc T. 2015. There is no $(75, 32, 10, 16)$ strongly regular graph. arXiv preprint arXiv:1509.05933: 1-28
http://arxiv.org/abs/1509.05933 (25.9.2015)

Azarija J., Slana U. v pripravi. An efficient algorithm for the maximal clique problem

Carraghan R., Pardalos P. M. 1990. An exact algorithm for the maximum clique problem. Operations Research Letters, 9, 6: 375–382

Compeau P., Pevzner P. 2014. Bioinformatics algorithms an active learning ppproach. La Jolla, Active Learning Publishers: 392 str.

Compeau P. E., Pevzner P. A., Tesler G. 2011. How to apply de Bruijn graphs to genome assembly. Nature Biotechnology, 29, 11: 987–991

Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. 2009. Introduction to Algorithms. 3rd ed. Cambridge, The MIT Press: 1312 str.

Cvetkovic D. M., Doob M., Sachs H. 1997. Spectra of Graphs: Theory and Applications. Leipzig, Huthig Pub Ltd: 447 str.

Davidson N. M., Oshlack A. 2014. Corset: enabling differential gene expression analysis for de novo assembled transcriptomes. Genome Biology, 15, 7: 410

Depolli M., Konc J., Rozman K., Trobec R., Janezic D. 2013. Exact parallel maximum clique algorithm for general and protein graphs. Journal of Chemical Information and Modeling, 53, 9: 2217–2228

Dukka B. K., Tomita E., Suzuki J., Horimoto K., Akutsu T. 2006. Protein threading with profiles and distance constraints using clique based algorithms. Journal of Bioinformatics and Computational Biology, 4, 01: 19–42

Godsil C., Royle G. 2001. Algebraic graph theory. Graduate Texts in Mathematics, vol. 207. New York, Springer-Verlag: 433 str.

Grabherr M. G., Haas B. J., Yassour M., Levin J. Z., Thompson D. A., Amit I., Adiconis X., Fan L., Raychowdhury R., Zeng Q., ... Regev A. 2011. Full-length

transcriptome assembly from RNA-Seq data without a reference genome. Nature Biotechnology, 29, 7: 644–652

Håstad J. 1999. Clique is hard to approximate within $n^{1-\varepsilon}$. Acta Mathematica, 182, 1: 105–142

Hutchison C. A. 2007. DNA sequencing: bench to bedside and beyond. Nucleic Acids Research, 35, 18: 6227–6237

Ji Y., Xu X., Stormo G. D. 2004. A graph theoretical approach for predicting common RNA secondary structure motifs including pseudoknots in unaligned sequences. Bioinformatics, 20, 10: 1591–1602

Johnson D. S., Trick M. A. 1996. Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993, vol. 26. Boston, American Mathematical Society.: 657 str.

Junttila T., Kaski P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. V: Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithms and Combinatorics. Applegate D., Brodal G. S., Panario D., Sedgewick R. (eds.). New Orleans, LA, SIAM: 135–149

Karp R. M. 1972. Reducibility among combinatorial problems. V: Complexity of Computer Computations. Raymond E. M., James W. T., Jean D. B. (eds.). Berlin, Springer: 85–103

Kerfeld C. A., Scott K. M. 2011. Using BLAST to teach "e-value-tionary" concepts. PLoS Biology, 9, 2: e1001014, doi: 10.1371/journal.pbio.1001014: 4 str.

Konc J., Janežič D. 2007. An improved branch and bound algorithm for the maximum clique problem. MATCH Communications in Mathematical and in Computer Chemistry, 50: 569–590

Lander E. S., Waterman M. S. 1988. Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics, 2, 3: 231–239

Li H., Homer N. 2010. A survey of sequence alignment algorithms for next-generation sequencing. Briefings in Bioinformatics, 11, 5: 473–483

Li Z., Chen Y., Mu D., Yuan J., Shi Y., Zhang H., Gan J., Li N., Hu X., Liu B., others 2012. Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph. Briefings in Functional Genomics, 11, 1: 25–37

MacArthur B. D., Sánchez-García R. J., Anderson J. W. 2008. Symmetry in complex networks. Discrete Applied Mathematics, 156, 18: 3525–3531

Matsunaga T., Yonemori C., Tomita E., Muramatsu M. 2009. Clique-based data mining for related genes in a biomedical database. BMC bioinformatics, 10, 1: 205, doi: 10.1186/1471–2105–10–205: 9 str.

McKay B. D., Piperno A. 2014. Practical graph isomorphism, {II}. Journal of Symbolic Computation, 60, 0: 94 – 112

Miller J. R., Koren S., Sutton G. 2010. Assembly algorithms for next-generation sequencing data. Genomics, 95, 6: 315–327

Myers E. W., Sutton G. G., Delcher A. L., Dew I. M., Fasulo D. P., Flanigan M. J., Kravitz S. A., Mobarry C. M., Reinert K. H., Remington K. A., Anson E. L., Bolanos R. A., Chou H.-H., Jordan C. M., Halpern A. L., Lonardi S., Beasley E. M., Brandon R. C., Chen L., Dunn P. J., Lai Z., Liang Y., Nusskern D. R., Zhan M., Zhang Q., Zheng X., Rubin G. M., Adams M. D., Craig V. J. 2000. A whole-genome assembly of Drosophila. Science, 287, 5461: 2196–2204

Östergård P. R. 2002. A fast algorithm for the maximum clique problem. Discrete Applied Mathematics, 120, 1: 197–207

Pevzner P. A., Tang H., Waterman M. S. 2001a. An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences, 98, 17: 9748–9753

Pevzner P. A., Tang H., Waterman M. S. 2001b. A new approach to fragment assembly in DNA sequencing. V: Proceedings of the fifth annual international conference on Computational biology, Montreal, April 22 – 25. New York, ACM: 256-267

Rokhlenko O., Wexler Y., Yakhini Z. 2007. Similarities and differences of gene expression in yeast stress conditions. Bioinformatics, 23, 2: e184–e190

Sedgewick R., Wayne K. 2011. Algorithms. 4th ed. Boston, Addison-Wesley Professional: 992 str.

Shendure J., Ji H. 2008. Next-generation DNA sequencing. Nature Biotechnology, 26, 10: 1135–1145

Sievers F., Wilm A., Dineen D., Gibson T. J., Karplus K., Li W., Lopez R., McWilliam H., Remmert M., Söding J., Thomson J. D., Higgins D. G. 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Molecular Systems Biology, 7, 1: 539, doi: 10.1038/msb.2011.75: 6 str.

Stephens Z. D., Lee S. Y., Faghri F., Campbell R. H., Zhai C., Efron M. J., Iyer R., Schatz M. C., Sinha S., Robinson G. E. 2015. Big data: astronomical or genomical? PLoS Biology, 13, 7: e1002195, doi: 10.1371/journal.pbio.1002195: 11 str.

Tomita E., Seki T. 2003. An efficient branch-and-bound algorithm for finding a maximum clique. V: Discrete mathematics and theoretical computer science. Cristian S. C., Michael J. D., Vincent V. (eds.). Berlin, Springer: 278–289

Tomita E., Sutani Y., Higashi T., Takahashi S., Wakatsuki M. 2010. A simple and faster branch-and-bound algorithm for finding a maximum clique. V: WALCOM: Algorithms and computation. Saidur R., Satoshi F. (eds.). Berlin, Springer: 191–203

Töpfer A., Marschall T., Bull R. A., Luciani F., Schönhuth A., Beerenwinkel N. 2014. Viral quasispecies assembly via maximal clique enumeration. PLOS Computational Biology, 10, 3: e1003515, doi:10.1371/journal.pcbi.1003515: 10 str.

Wallis W. D. 2010. A beginner's guide to graph theory. 2nd ed. Boston, Birkhäuser: 260 str.

Wang Z., Gerstein M., Snyder M. 2009. RNA-Seq: a revolutionary tool for transcriptomics. Nature Reviews Genetics, 10, 1: 57–63

Wheeler D., Bhagwat M. 2007. BLAST QuickStart. V: Comparative Genomics. Vol. 1. Bergman N. H. (ed.). Totowa, Humana Press: 149- 176

Wilbur W. J., Lipman D. J. 1983. Rapid similarity searches of nucleic acid and protein data banks. Proceedings of the National Academy of Sciences, 80, 3: 726–730

# ZAHVALA

Zahvaljujem se mentorju doc. dr. Tomažu Curku za pomoč pri izdelavi magistrske naloge.

Zahvala tudi Vasji Progarju iz Biotehniške fakultete za pomoč pri pripravi podatkov in razprave o analizi dobljenih rezultatov.

Posebna zahvala gre Jerneju Azariji, ki je mi je zadnja leta stal ob strani in mi bil mentor pri učenju programiranja. Zahvaljujem se mu tudi za strokovne pripombe pri izdelavi magistrske naloge.

Hvala Tini Vouk za pregled angleškega besedila.

Hvala mami.

Hvala Smiljanki in Dušanu.

In nenazadnje:

Jelena, hvala!