

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4189

**Generiranje konsenzusnog slijeda
iz grafova djelomično uređenih
višestrukih poravnanja**

Adriano Baćac

Zagreb, lipanj 2015.

*Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.*

Zahvaljujem se Mili Šikiću na vodstvu i trudu oko prakse. Zahvaljujem se Ani Bulović na pruženoj pomoći i strpljenju, bez nje ovaj rad bi još bio u izradi.

SADRŽAJ

Popis slika	vi
1. Uvod	1
2. Biološke osnove	3
2.1. Poravnanje sljedova	3
2.2. Konsenzus	4
2.3. Graf djelomično uređenih višestrukih poravnanja	5
3. Generiranje grafa	7
3.1. Poravnanje slijeda na slijed	7
3.2. Poravnanje slijeda na graf	9
4. Dobivanje konsenzusa	10
4.1. Stvaranje grafa	11
4.2. Stvaranje pravila	11
4.3. Pronalaženje konsenzusnog slijeda	13
4.3.1. Obilazak grafa	13
4.3.2. Obrada čvora	15
4.4. Grupiranje sljedova	17
4.5. Generiranje izlaza	18
4.6. Vremenska i memorijska složenost	19
5. Testni primjeri	20
5.1. Ovisnost o duljini referentnog slijeda	21
5.2. Ovisnost o pokrivenosti nukleotida	22
6. Analiza rezultata	24
7. Zaključak	25

Dodatak	26
A. Format zapisa grafa	27
A.1. Podaci o alatu	27
A.2. Podaci o sljedovima	27
A.3. Podaci o čvorovima	28
Literatura	29

POPIS SLIKA

1.1. Primjer zapisa višestrukog poravnanja sljedova	1
2.1. Primjer zapisa konsenzusnog slijeda	4
2.2. Poravnanja različitih sljedova koja imaju jednak profil	5
2.3. Više mogućnosti poravnanja dva slijeda	5
2.4. Poravnanje pomoću usmjerenog grafa	6
3.1. Izračun vrijednosti jednog polja matrice pri poravnanju slijeda na slijed	7
3.2. Određivanje poravnanja praćenjem prijelaza	8
3.3. Izračun vrijednosti jednog polja matrice pri poravnanju slijeda na graf	9
4.1. Osnovni tok rada alata Conpoa	10
4.2. Stvaranje grafa	11
4.3. Stvaranje pravila	12
4.4. Više jednako valjanih prolazaka kroz graf	14
4.5. Moguće situacije pri određivanju najboljeg prethodnika	16
4.6. Poseban slučaj gdje prethodnik nije čvor s najvećom vrijednošću što može značiti da čvor s najvećom vrijednošću ima sljedbenika	16
4.7. Odnos komponenti za stvaranje izlaza	18
5.1. Vrijeme provedeno na procesoru ovisno o duljini referentnog slijeda .	21
5.2. Memorijska potrošnja ovisno o duljini referentnog slijeda	22
5.3. Vrijeme provedeno na procesoru ovisno o duljini manjih sljedova . . .	23
5.4. Memorijska potrošnja ovisno o duljini manjih sljedova	23
6.1. Usporedba poravnanja konsenzusa i referentnog slijeda	24

1. Uvod

Konsenzus je genetski (ili proteinski) slijed koji predstavlja najvjerojatniji slijed kojeg je moguće rekonstruirati iz višestrukog poravnanja sljedova. Generiranje konsenzusnog slijeda ili profila iz višestrukog poravnanja sljedova značajan je korak u velikom broju analize sljedova. Ima veliki raspon primjene, kao što je na primjer sastavljanje genoma iz kraćih očitavanja.

Konsenzus se obično dobiva prebrojavanjem nukleotida na pojedinoj poziciji u višestrukom poravnanju. Svaki znak u konsenzusnom slijedu odgovara znaku koji se najčešće javlja na tom položaju.

```
Q5E940 BOVIN -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_HUMAN -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_MOUSE -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_BAT -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_CHICK -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_RAMSE -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
Q72UG3 BRARE -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_TCTPU -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_DROME -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
RLAO_DICDI -----MREDRATWKSNSYELKLIQLDDVFKCFIVGADNVGKQMQIIMSLSLGGK-AVVLGKNTMMRKAIRGHLENN--PALE 76
Q54LP0 DICDI -----MSEAE-SRKNVFEKTKLFTTDMKIVAEADVFGSGLQKIRKSLRGI-GAVLGGKNTMIRKVVIRDLADSK--PELD 75
RLAO_PLAFB -----MAKLSKQOKMYTEKLSLELQQSKLLVHYVNVGSMASVKSLSLGGK-AVVLGKNTMIRKVVIRDLADSK--PELD 76
RLAO_SULAC -----MELAVTTTKERKRVDAEETKTKTKLITLIEANIEGFPADKLDHDKKMRGM-ALIKVKTLLFKIAAKNAG----LQV 79
RLAO_SULTO -----MRINAVITQRKIAKKEIEEKLEKREKRENTLIIANIEGFPADKLDHDKKMRGM-ALIKVKTLLFKIAAKNAG----LQV 80
RLAO_SULSO -----MKRIALALQKRVASWKEEVEKELTELKNSNTLLIENIEGFPADKLDHDKKMRGM-ALIKVKTLLFKIAAKNAG----LQV 80
RLAO_ARBE HSVSLVQHYKREKLEPKETLMLRELELEKSRVLEADLIGEPFVYRVRKLLKWK--PHMFAKRLILRANKAGLE--LDDH 86
RLAO_PYAE -BMLALEGSRVYKQVAKRVTYSEETLQKQVYVLDAGLSRLEHREKLEKRY-VYKELTIDREPTAFTVYGG--LQV 89
RLAO_METAC -----MSEERHTEHIDPKKDEEENKELQSHKVFQVYEGLEGLATKMKIRDLKDV-AVLKVSNTLLEKALNQLG----ETFD 78
RLAO_METMA -----MSEERHTEHIDPKKDEEENKELQSHKVFQVYEGLEGLATKMKIRDLKDV-AVLKVSNTLLEKALNQLG----ETFD 78
RLAO_ARCFU -----MRAVLEG--DPEETVAVTEKQMSSEVYVYVSEVYVFAQMKKIRREYQK-RLTKVYKELLEKEDALEG----DQV 75
RLAO_METKA MAVKAKGQDSCVYKPKVAEKRRREKELKEMDEENYGLVDLEGIPAPQLDQIRAKLREDDITMRNTLMRIALEKEDER--PELD 88
RLAO_METH -----MAHVAEKKKKVEQLNDLKEEVEVVGIANIADTARLQKMSQTLRDS-ALLRMSKRLTSLALKKRREL--BNVD 74
RLAO_METL -----MTEAESEKELAVWIEENKELDKMQLVALVDHMEVVARLQEDDKIN--EHLRMSNTLLEKALNQLG----ETFD 82
RLAO_METJA -----MIDAKSEHTAVKIEEYMAKELKSNVYALIDMMEVAVLQETKDKTE-DQMLRMSNTLLEKALNQLG----ETFD 82
RLAO_PYRAB -----METKVAHVADKTEEWKTLKGLKSKPVVATVDMDVFPALQETDKIN--DKVLRMSNTLLEKALNQLG----ETFD 81
RLAO_PYRFO -----MAHVAEKKKKVEELANLKSFPVVALVDVSSMAYPLSQMRLIENNGLLVSRNTLLEKALNQLG----ETFD 77
RLAO_PYRKO -----MAHVAEKKKKVEELANLKSFPVVALVDVAGVAVYLSKMSDKLE--KALLVSRNTLLEKALNQLG----ETFD 76
RLAO_HALMA -----MSESEVQTEVIEQKREVEEVDVDFEESVYGVYVAGIPRLQSLMSRELEHS-AVLRMSNTLLEKALNQLG----ETFD 79
RLAO_HALVO -----MSESEVQTEVIEQKREVEEVDVDFEESVYGVYVAGIPRLQSLMSRELEHS-AVLRMSNTLLEKALNQLG----ETFD 79
RLAO_HALSA -----MSESEVQTEVIEQKREVEEVDVDFEESVYGVYVAGIPRLQSLMSRELEHS-AVLRMSNTLLEKALNQLG----ETFD 79
RLAO_THAC -----MKEVSOQKELMNETIRKASRSVAIVDAGLRRTIPIQINRMSK--ENLKVIRKLLFKALENLDG----EKLS 72
RLAO_THYD -----MKEVSOQKELMNETIRKASRSVAIVDAGLRRTIPIQINRMSK--ENLKVIRKLLFKALENLDG----EKLS 72
RLAO_PICTO -----MTEPQOKIDFVKNLENEENSRKVAIVSIEKLRNMEFKIKNSLSDK-ARIKVSRALLRLAENLQK--NNTV 72
tuler 1.....10.....20.....30.....40.....50.....60.....70.....80.....90
```

Slika 1.1: Primjer zapisa višestrukog poravnanja sljedova

Slika preuzeta 10.6.2015. sa

en.wikipedia.org/wiki/Multiple_sequence_alignment#/media/File:RPLP0_90_ClustalW_aln.gif,

autor Miguel Andrade

Osnovna pretpostavka pri generiranju konsenzusa je da iza višestrukog poravnanja stoji jedan referentni slijed, i većina algoritama polazi od te pretpostavke. No, kako generirati konsenzus ako je temeljna pretpostavka, da poravnati sljedovi formiraju samo jedan konsenzus, neispravna?

Konsenzus tipično sadrži dvije vrste promjena koje se mogu dogoditi: supstituciju i dodavanje/brisanje. Ove izmjene su lokalne i ne mogu promijeniti strukturu na glo-

balnoj razini. Iz tog razloga svi će sljedovi koji nastanu imati samo lokalne razlike, a gledajući globalnu strukturu, bit će jednaki. U svijetu genetike usporedbe sljedova mogu postati dosta složenije. Sa izmjenama kao što su razmještanje domene, duplikacija i rekombinacija u skup sljedova uvodi se složeni sustav grananja koji se protivi ideji da se može predstaviti samo jednim konsenzusom. Gledajući jednostavan primjer, dva višedomenska slijeda koja se preklapaju u jednoj domeni ne mogu biti predstavljena jednim konsenzusom.

Umjesto tabličnog prikaza tražimo prirodniji način da prikažemo grananje sljedova - usmjereni graf. Svaki čvor u grafu predstavlja jedan nukleotid, s tim da su zajednički nukleotidi spojeni u jedan čvor. Za razliku od tabličnog prikaza, razlike u sljedovima prikazane su različitim putevima kroz graf.

Ovim radom predstavlja se metoda dobivanja konsenzusnih sljedova iz višestrukih poravnanja sljedova predstavljenih grafom.

U nastavku rada dane su osnovne informacije potrebne za razumijevanje rada (poglavlje 2), predložena struktura podataka koja rješava problem prikaza višestrukih poravnanja (poglavlje 2.3) te način kako ju stvoriti (poglavlje 3) i način kako iz te iste strukture dobiti više konsenzusnih sljedova (poglavlje 4).

2. Biološke osnove

Bioinformatika je grana znanosti koja se bavi primjenom računalne tehnologije u svrhu obrade i razumijevanja bioloških podataka. Obuhvaća područja računarske znanosti, statistike i inženjerstva.

Procvat bioinformatike dogodio se pojeftinjenjem i sve većom dostupnosti strojeva za sekvenciranje (Illumina, PacBio, Ion Torrent, Roche 454, Oxford Nanopore), mjerenje razina ekspresije gena (microarray), količine proteina (masena spektrometrija) te interakcije proteina s DNA (Chip-Seq). Ove su tehnologije uzrokovale ubrzan razvoj algoritama za analizu bioloških podataka, te zbog konstantno rastuće veličine tih podataka, kao i njihovog većeg razumijevanja, sve se više ulaže u razvoj novih računalnih metoda koje bi omogućile njihovu pohranu, obradu, analizu i prikaz.

2.1. Poravnanje sljedova

Višestruko poravnanje sljedova, dalje spominjano kao MSA (*eng. Multiple Sequence Alignment*), je poravnanje više sljedova koji uglavnom predstavljaju proteine, RNA ili DNA. U većini slučajeva pretpostavlja se da sljedovi koji se poravnavaju potječu od zajedničkog pretka.

Globalno poravnanje dva sljedova moguće je napraviti u složenosti $O(L^2)$ koristeći dobro poznati Needleman-Wunsch algoritam [6], što je proširivo na N sljedova sa složenosti $O(L^N)$ (što u realnim slučajevima nije primjenjivo). Način rada algoritma prikazan je u detaljnije u poglavlju 3.1, nakon čega je njegova funkcionalnost dodatno proširena u poglavlju 3.2.

Najšire korišten heuristički pristup poravnanju višestrukih sljedova korištenje je progresivno poravnanje (alat *ClustalW*¹). Progresivno poravnanje gradi MSA kombinirajući globalna poravnanja sljedova, počevši od najsličnijih prema različitim.

¹Najčešće korištena metoda progresivnog poravnanja za koju postoji veći broj internet stranica koje nude mogućnost poravnanja, tako da se ne mora lokalno instalirati program.

Metoda progresivnog poravnanja može se podijeliti u dvije faze.

1. izgradnja pomoćnog stabla temeljenog na mjeri sličnosti između parova sljedova
2. dodavanje sljedova u poravnanje prateći pomoćno stablo

Algoritam počinje poravnavanjem svih parova sljedova. U svakom se koraku odabire novi slijed po kriteriju najveće sličnosti s referentnim slijedom (ili referentnim poravnavanjem). Novi se slijed poravnava na poravnanje generirano u prethodnom koraku. Algoritam staje kada su svi sljedovi poravnati.

2.2. Konsenzus

Konsenzus je genetski (ili proteinski) slijed je najvjerojatniji slijed kojeg je moguće rekonstruirati iz višestrukog poravnanja sljedova. Svaki znak u konsenzusnom slijedu odgovara znaku koji se najčešće javlja na tom položaju u višestruko poravnatim sljedovima. Znak u konsenzusu ne mora nužno odgovarati samo jednom znaku, već može biti skup znakova, isključenje znakova te pripadnost određenom biološkom skupu (npr. pirimidin ili purin).

Na primjeru sa Slike 2.1 dan je jedan od mogućih zapisa konsenzusnog slijeda. U ovom zapisu GC znači da je na početku slijeda uvijek gvanin nakon kojeg slijedi citozin. {T} predstavlja dušičnu bazu koja nije timin (adenin, gvanin ili citozin). Y predstavlja neki pirimidin (citozin ili timin), nakon kojega slijedi adenin (A). [CA] je skup baza, može biti ili citozin ili adenin. Jednako kao što Y predstavlja neki pirimidin, R predstavlja neki purin (adenin ili gvanin).

GC { T } YA [CA] R

Slika 2.1: Primjer zapisa konsenzusnog slijeda

2.3. Graf djelomično uređenih višestrukih poravnanja

Ako želimo generirati konsenzus iz matrice MSA, pojavljuju se neki problemi tipični za takav tablični prikaz. Za generiranje konsenzusa koriste se svi stupci jer bi inače odmah došlo do gubitaka podataka. Čak i ako je u konsenzusu za svaki položaj zapamćeno koliko je kojih baza i rupa bilo, još uvijek nedostaje podatak koja baza pripada kojem slijedu. Na primjer, poravnanja sa Slike 2.2 imaju jednak konsenzus.

```

Slijed_1 -----MVSEGRVVRVNGPLVIADGMREAQMFEVVYVSDLKLVGE
Slijed_2 -----MGRIRVVTGPLVVADGMKGAKMYEVVRVVGEMGLIGE
Slijed_3 -----E----SKAKEGDYGSIKKVGSPVVADNMGGGSAMYELVRVGTGELIGE
Slijed_4 MPSVYGDRLTTFM----DSEKESEYGYVRKVGSPVVADGMGGAAMYELVRVGHDLNIGE

Slijed_5 -----EGDMGSVVRVNGPLVIADGMGGAAMYEVVVYVSDLELIGE
Slijed_6 -----EGYIRVVTGPLVVADGMKGAKMYEVVRVVGEMGLIGE
Slijed_7 -----M----DKEKMSYGRIKKVGSPVVADNMGGGSAMYELVRVGTGKLVGE
Slijed_8 MPSVYGDRLTTFE----SSAKEVEYGRVVRKVGSPVVADGMREAQMFELVRVGHDLNIGE

```

Slika 2.2: Poravnanja različitih slijedova koja imaju jednak profil

Također, za poravnanje dva niza postoji više ekvivalentnih načina, što uvodi dodatne greške u krajnjem MSA koristeći progresivno poravnanje. Koji je način najbolji ovisi o metodi bodovanja rupa u poravnanju. Uzmimo kao primjer dva slijeda:

```

ACGATTCAGT
ACGAAAGAGT

```

Slijedovi su jednaki uz iznimku središnjeg dijela, gdje se razlikuju za tri nukleotida. Poravnanje ovih slijedova može se prikazati na više načina (Slika 2.3), iako su, biološki gledano, svi jednaki. Kako se sve više slijedova dodaje u MSA konačno se rješenje malo po malo iskrivljuje upravo zbog tih rupa.

```

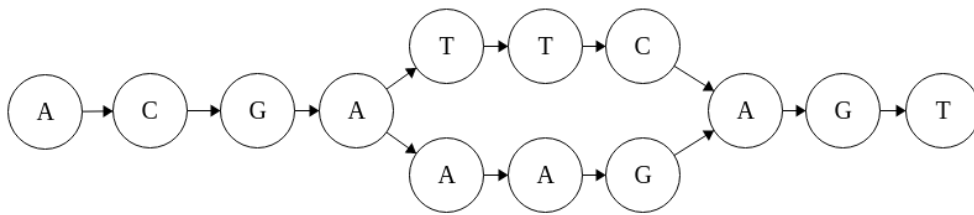
ACGATTC--AGT   ACGA---TTCAGT   ACGAT-TC--AGT
ACGA--AAGAGT   ACGAAAG--AGT   ACGA-A--AGAGT

ACGATT-C--AGT   ACGA--TT-CAGT   ACGAT-T-C-AGT
ACGA--A-AGAGT   ACGAAA--G-AGT   ACGA-A-A-GAGT

```

Slika 2.3: Više mogućnosti poravnanja dva slijeda

Umjesto tabličnog prikaza tražimo prirodniji način da prikažemo grananje sljedova - usmjereni graf. Za razliku od tabličnog prikaza, razlike u sljedovima prikazane su različitim putevima kroz graf. Svi poravnati elementi spojeni su u jedan čvor. U tabličnom prikazu svako slovo imalo je samo jednog prethodnika (prethodni element u istom slijedu), dok kod usmjerenog grafa jedno slovo može imati više prethodnika. Na slici 2.4 nukleotid A nakon spajanja ima dva prethodnika: C i G, svaki iz svojeg niza. Svi prethodni različiti primjeri poravnanja se mogu prikazati na samo jedan način pomoću grafa.



Slika 2.4: Poravnanje pomoću usmjerenog grafa

3. Generiranje grafa

Za generiranje prethodno opisanog grafa djelomično uređenih višestrukih poravnanja korišten je programski alat *POA* ([1]). U nastavku je detaljnije objašnjen algoritam korišten pri stvaranju grafa.

3.1. Poravnanje slijeda na slijed

Da bi se moglo razumjeti na koji način *POA* poravnava sljedove potrebno je razumjeti kako poravnati dva slijeda pomoću Needleman-Wunsch (ili iz njega izvedenog) algoritma.

Iz dva slijeda generira se matrica gdje je jedan slijed predstavljen retcima, a drugi stupcima. Za svako polje matrice računaju se vrijednosti mogućih prijelaza te se pamti koji je prijelaz najbolji.

Moguća su tri prijelaza (na slici 3.1 svaki označen svojom bojom):

1. Sljedovi su poravnati
2. Rupa u prvom slijedu
3. Rupa u drugom slijedu

		A	C	T	T	
	prvi slijed drugi slijed	0	-10	-20	-30	-40
G	-10	1	?			
T	-20					
T	-40					

Slika 3.1: Izračun vrijednosti jednog polja matrice pri poravnanju slijeda na slijed

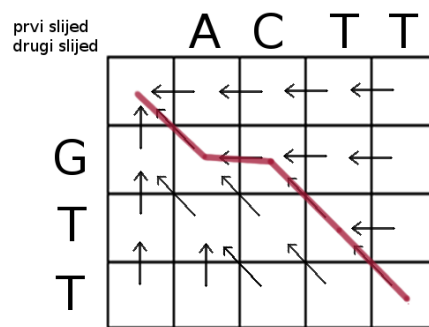
Vrijednost svakog polja matrice može se izračunati kao

$$vrijednost(i, j) = \max \begin{cases} vrijednost(i - 1, j - 1) + s(x_i, y_j) \\ vrijednost(i, j - 1) + g \\ vrijednost(i - 1, j) + g \end{cases} \quad (3.1)$$

gdje je s vrijednost za par baza x_i i y_j , a g cijena rupe.

Polja se mogu prolaziti proizvoljnim redosljedom, pod uvjetom da su polja potrebna za izračun vrijednosti trenutnog polja već obrađena. Po završetku se, prateći prijelaze od krajnjeg polja, rekonstruira poravnanje. Po rekonstruiranom puta sa Slike 3.2 zaključujemo da je poravnanje sljedova ACTT i GTT:

ACTT
-GTT



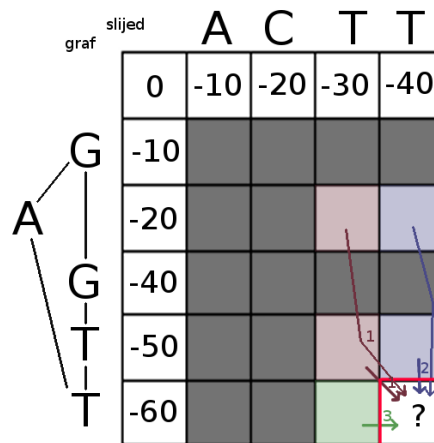
Slika 3.2: Određivanje poravnanja praćenjem prijelaza

3.2. Poravnanje slijeda na graf

Algoritam poravnanja slijeda na slijed može se jednostavno proširiti na poravnanje slijeda na graf. Jedina je razlika u tome što sada jedan element može imati više prethodnika. Matematički gledano, vrijednost svakog polja matrice može se izračunati kao:

$$vrijednost(i, j) = \max \begin{cases} vrijednost(p, j - 1) + s(x_i, y_j) \\ vrijednost(i, j - 1) + g \\ vrijednost(p, j) + g \end{cases} \quad (3.2)$$

gdje je s vrijednost za par baza x_i i y_j , a g cijena rupe. Jedina razlika od prethodno navedene formule je uvođenje varijable p koja predstavlja sve prethodnike trenutnog elementa, a ne samo $i - 1$ element.



Slika 3.3: Izračun vrijednosti jednog polja matrice pri poravnanju slijeda na graf

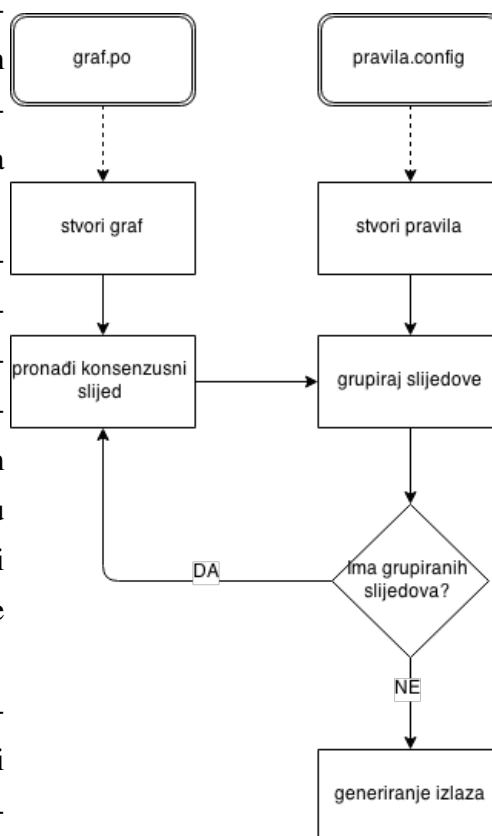
4. Dobivanje konsenzusa

Za analizu grafa u svrhu dobivanja jednog ili više konsenzusnih sljedova (profila) napravljen je program Conpoa (*eng. CONsensus from Partial Order Aligment*). U cijelosti je implementiran u programskom jeziku C++, prateći Google-ov standard. Opis konkretne implementacije nalazi se u samom kodu, dok je u radu opisan princip rada. Nakon opisa osnovnog principa rada alata svaki pojedini korak bit će detaljno analiziran, redosljedom kojim se komponente koriste pri analizi grafa.

Iz prethodno opisanog zapisa grafa kojeg generira alat *POA* stvara se graf djelomično uređenih višestrukih poravnanja. Graf se analizira algoritmom opisanim u poglavlju 4.3 te se dobiva konsenzusni sljed (ili sljedovi).

Generiranje konsenzusa je zapravo dodjeljivanje pojedinih sljedova konsenzusu. Da bi se sljed dodijelio konsenzusu, ne smije već biti dodijeljen postojećem konsenzusu te treba zadovoljiti prethodno definiran skup pravila. Svim grupiranim sljedovima smanjuje se doprinos u grafu da bi sljedeća iteracija generirala različiti konsenzus. Postupak se ponavlja dokle god se može grupirati barem jedan sljed.

Ukoliko se stvore dva jednaka konsenzusa prestaje se s radom, to implicira da su svi sljedovi koji se mogu grupirati s tim konsenzusom grupirani u prošloj iteraciji i uvjet za nastavak algoritma nije zadovoljen.



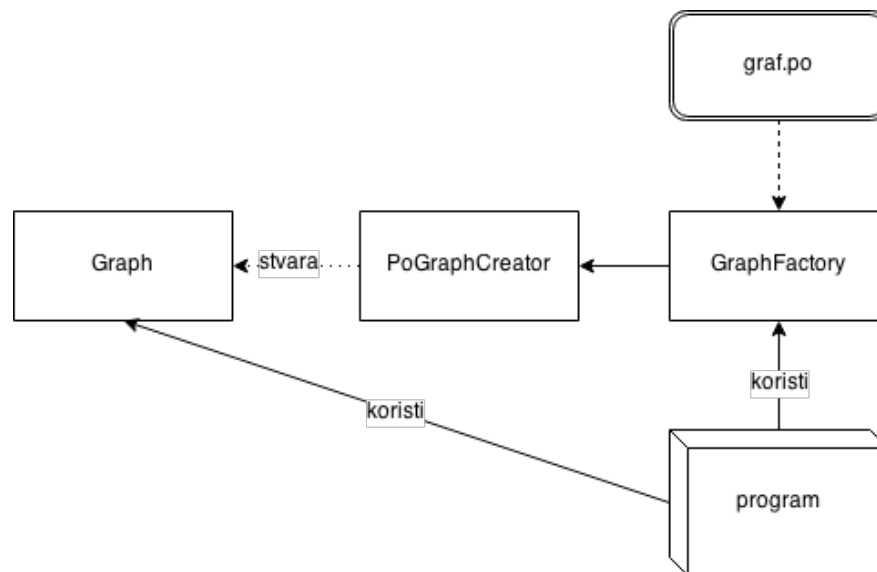
Slika 4.1: Osnovni tok rada alata Conpoa

4.1. Stvaranje grafa

Conpoa podržava samo jedan format zapisa grafa, onaj kojeg generira *POA*. No, novi podržani formati mogu se jednostavno dodati u *GraphFactory*. Sam graf sadrži tri pomoćne klase:

1. **Node** - Jedan čvor u grafu
2. **Seq** - Slijed sadržan u grafu
3. **Link** - Veza između čvorova

Graf je predstavljen i čvorovima i vezama zbog smanjenja potrebnih izračuna pri analizi samog grafa. Konkretno, pri određivanju konsenzusnog slijeda potrebna je informacija koje veze sadrže koje sljedove, a pri grupiranju slijeda i konsenzusa potrebna je informacija koji čvor sadržava koji slijed.



Slika 4.2: Stvaranje grafa

4.2. Stvaranje pravila

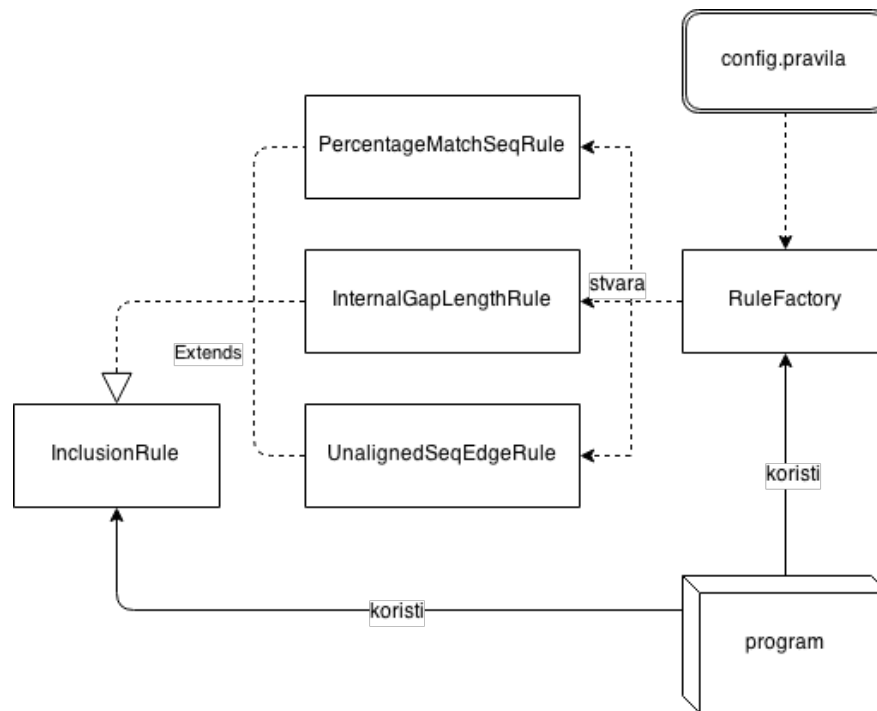
Da bi slijed bio grupiran uz konsenzus treba zadovoljiti sva pravila grupiranja. Izmjenom parametara pravila mijenjaju se i generirani konsenzusni sljedovi zbog toga što se mijenja odabir sljedova koji su predstavljeni pojedinim konsenzusom. Za vrijeme pisanja ovog rada podržana su tri pravila:

1. Postotak sadržajnosti slijeda u konsenzusu
2. Najveća dozvoljena rupa u poravnanju slijeda i konsenzusa
3. Najveći dozvoljeni rubni ne poravnati broj baza

Kao što je slučaj i za podržane formate zapisa grafa, nova se pravila jednostavno dodaju. Aktivna pravila zadaju se posebnom konfiguracijskom datotekom, gdje je svako pravilo sa svojim parametrima u zasebnom retku.

Na primjer, kad bismo htjeli grupirati samo sljedove koji su 90% sadržani u konsenzusu, nemaju rupu u poravnanju veću od 10 baza i ne poravnati rub nije veći od 15 baza, predali bi konfiguracijsku datoteku oblika:

```
seq_percentage_in_consensus 0.9 1
internal_gap_length 10
unaligned_seq_edge 15
```



Slika 4.3: Stvaranje pravila

4.3. Pronalaženje konsenzusnog slijeda

U poglavlju 2.3 prikazani su problemi kod dobivanja konsenzusnog slijeda pomoću najčešće baze za svaki stupac. Problem traženja konsenzusnog slijeda u grafu pretvara se u pronalaženje najboljeg puta kroz graf. Algoritam je, kao i generiranje samog grafa, temeljen na dinamičkom programiranju. Krećući se po čvorovima slijeva nadesno računa se najbolja ulazna veza za svaki čvor i i vrijednost samog čvora.

Nakon obilaska grafa bira se čvor s najboljim rezultatom i , prateći zapamćene veze, rekonstruira se konsenzus. Kako postoji više mogućih puteva kroz graf, dobiveni najbolji put predstavlja samo dio sljedova u poravnanju.

```
Za svaki čvor  $i$  u grafu slijeva nadesno:  
 $p \leftarrow$  najbolji_prethodnik( $i$ )  
ako postoji  $p$ :  
    spremi_najboljeg_prethodnika( $i$ ,  $p$ )  
    vrijednost( $i$ )  $\leftarrow$  vrijednost( $p$ ) + vrijednost( $i$ ,  $p$ )  
vrati  $r$  gdje je  $vrijednost_r = \max_i\{vrijednost_i\}$ 
```

4.3.1. Obilazak grafa

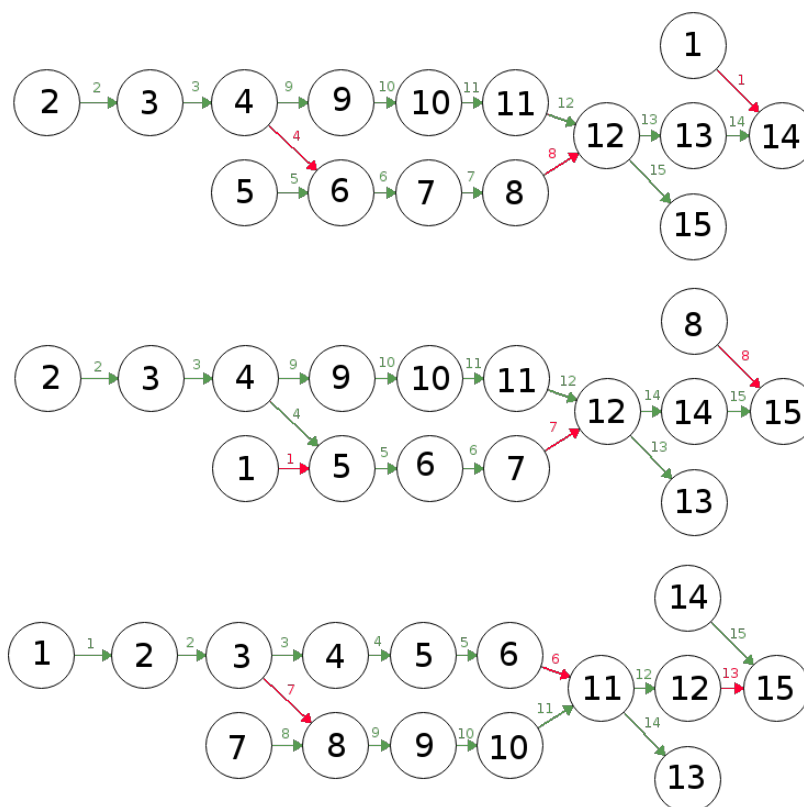
```
Za svaki čvor  $i$  u grafu slijeva nadesno:
```

Iako *.po* format kojeg daje alat *POA* ima čvorove zapisane u redosljedu obilaska, u implementaciji za ovaj rad obilazak ne ovisi o prethodno sortiranim čvorovima, čime se nudi veća mogućnost drugih ulaznih formata.

Graf je potrebno obići čvor po čvor, s tim da se svaki čvor može analizirati tek nakon što su analizirani njegovi prethodnici. Prije same analize, svi početni čvorovi (oni bez ulaznih veza) dodaju se u listu za obradu. Za svaki čvor iz liste započinje se obrada grane grafa. Dokle god čvor nema više sljedbenika izračuna se njegova vrijednost i obrada se nastavlja njegovim sljedbenikom. Ukoliko čvor ima više sljedbenika, jednog se odabire za obradu, dok se svi ostali dodaju u listu za obradu. Obrada se nastavlja s odabranim čvorom. Ukoliko trenutni čvor ima više prethodnika, potrebno je provjeriti koliko se puta čvoru pristupilo i, tek nakon što je utvrđeno da su obrađeni svi ostali prethodnici, s obradom se nastavlja kao da čvor ima samo jednog prethodnika. Ukoliko nisu obrađeni svi prethodnici ili nema više sljedbenika, obrada grane prestaje te se uzima sljedeći čvor iz liste.

Na slici 4.4 prikazana su tri (od više mogućih) puta kroz graf. Čvorovi su označeni rednim brojem koji označava redoslijed obrade, a veze su označene rednim brojem pristupa vezi. Zelenom bojom prikazane su veze kojima je analiza uspješno nastavljena, a crvenom veze do čvorova koje u tom trenutku nije bilo moguće obraditi.

U prvom i drugom slučaju slijedno se započinje obrada grana grafa (jedina razlika je u redoslijedu početnih čvorova), dok se u trećem primjeru nakon prekida obrade (*veza₆* i *veza₁₃*) prvo obrađuju nadodani čvorovi (*veza₇* i *veza₁₄*), a tek onda početni čvorovi.



Slika 4.4: Više jednako valjanih prolazaka kroz graf

Programski isječak 4.1: Pronalaženje konsenzusnog slijeda

```
1 list <Node *> FindTopScoringPath () {
2   for (Node *start : graph_ -> starts ()) {
3     this -> to_process_ . push ( start );
4   }
5   while (! to_process_ . empty ()) {
6     Node *local_start = to_process_ . front ();
7     to_process_ . pop ();
8     ProcessBranch ( local_start );
9   }
10  return BranchCompletion ( top_scoring_node_ ) -> Traceback ();
11 }
```

4.3.2. Obrada čvora

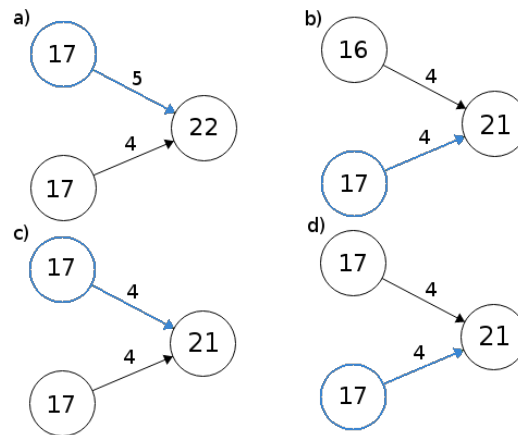
```
p <- najbolji_prethodnik (i)
ako postoji p:
    spremi_najboljeg_prethodnika (i, p)
    vrijednost (i) <- vrijednost (p) + vrijednost (i, p)
```

U početku svi čvorovi imaju vrijednost 0. Od svih ulaznih veza u čvoru bira se ona s najvećom težinom. Težina veze suma je težina svakog slijeda na njoj, gdje je pretpostavljena početna vrijednost za težinu svakog slijeda jednaka 1.

$$vrijednost(i, j) = \sum_k^{S_k: i \rightarrow j} težina(S_k) \quad (4.1)$$

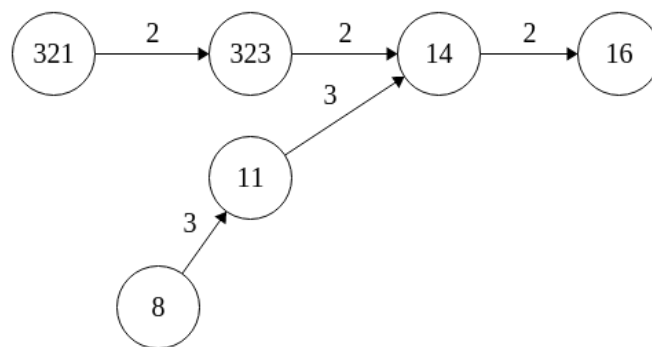
Ukoliko do čvora vode veze s jednakim težinama, u obzir se uzimaju vrijednosti čvorova iz kojih dolaze. Ukoliko i prethodni čvorovi imaju jednaku vrijednost, veza se bira proizvoljno (ne ide se rekurzivno do prve razlike). Primjer svih situacija prikazan je na slici 4.5.

Cilj je da konačno generirani put kroz graf predstavlja potpuni put, odnosno da započinje u čvoru bez prethodnika i završava u čvoru bez sljedbenika. Iako na prvi pogled pomoću prethodno opisanog algoritma vrijednost čvorova slijeva nadesno stalno raste, to nije slučaj. Postoji mogućnost da čvor među svojim prethodnicima ima jednoga s vrlo visokom vrijednošću, ali preko njihove zajedničke veze prolazi samo jedan slijed. U tom će slučaju kao prethodnik prije biti odabran čvor s malom vrijednošću, ali većom težinom veze, kao što je prikazano na slici 4.6.



Slika 4.5: Moguće situacije pri određivanju najboljeg prethodnika

Pod *a)* najbolji prethodnik određuje se pomoću težine veze ($5 > 4$). Pod *b)* težine veza jednake su, pa se najbolji prethodnik bira prema vrijednosti prethodnih čvorova ($17 > 16$). Primjeri *c)* i *d)* prikazuju jednaku situaciju gdje su i vrijednosti najvećih težina ($4 = 4$) i vrijednosti pripadnih čvorova ($17 = 17$) jednaki.



Slika 4.6: Poseban slučaj gdje prethodnik nije čvor s najvećom vrijednošću što može značiti da čvor s najvećom vrijednošću ima sljedbenika

Pošto se za rekonstrukciju puta kroz graf koristi čvor s globalno najvećom vrijednošću, moguće je da taj čvor ima sljedbenika. Problem se rješava ponovnim provođenjem algoritma traženja najboljeg puta, s tim da se svim ostalim prethodnicima vrijednost postavi na -1 čime se osiguravamo da neće biti odabrani. Ovime se osigurava da će preskočeni najbolji čvor biti odabran, no ponovno, moguće je da novi najbolji čvor nije krajnji. Algoritam je potrebno ponovno provoditi sve dok najbolji čvor nije krajnji. U samoj implementaciji ovaj problem riješen je jednostavnim odabirom najboljeg krajnjeg čvora, zbog toga što je prethodno opisana situacija rijetka i preskočeni čvor bit će na najboljem putu već u sljedećoj iteraciji kada se promijene težine grafa.

4.4. Grupiranje sljedova

Dobiveni najbolji put može predstavljati samo dio sljedova u poravnanju. Potrebno je odrediti koji su to sljedovi i, ako preostane sljedova koji nisu predstavljeni nijednim konsenzusnim slijedom, ponoviti postupak generiranja konsenzusnog slijeda.

Svaki slijed koji nema pridijeljeni konsenzus koji ga predstavlja uspoređuje se s novodobivenim konsenzusom. Za usporedbu se koriste pravila čije je stvaranje opisano u poglavlju 4.2.

```
I <- lista[]
Za svaki slijed  $S_k$  u grafu:
  ako  $S_{k,consensus} = \text{NULL}$  i zadovoljena_pravila( $S_k$ ,  $C$ )
     $S_{k,consensus} <- C$ 
  dodaj  $S_k$  u listu  $I$ 
vrati  $I$ 
```

Gdje je S_k jedan slijed u grafu, a C dobiveni konsenzusni slijed.

Svim sljedovima u skupu I sa pseudokoda težine su smanjene za određeni faktor, kako bi njihov doprinos pri određivanju konsenzusa bio smanjen (Formula 4.1). U programskom ostvarenju težina je bila smanjena na 0, čime je osigurano da grupirani sljedovi ne utječu na generiranje novog konsenzusa.

Programski isječak 4.2: Grupiranje sljedova sa konsenzusom

```
1  int SequenceBundler::AddSequencesToBundle(vector<Seq *> seqs ,
2  Seq *consensus , vector<Seq *> *bundled) {
3    assert(bundled != nullptr);
4    bundled->clear();
5    int cnt = 0;
6    vector<future<int> > results;
7
8    mutex *mylock = new mutex;
9    for (Seq *seq : seqs) {
10     if (!seq->HasConsensus()) {
11       results.emplace_back(
12         pool_>enqueue([ this , seq , consensus , bundled , mylock] {
13           if (ApplyInsertionRules(seq , consensus)) {
14             seq->set_consensus(consensus);
15             mylock->lock();
16             bundled->push_back(seq);
```

```

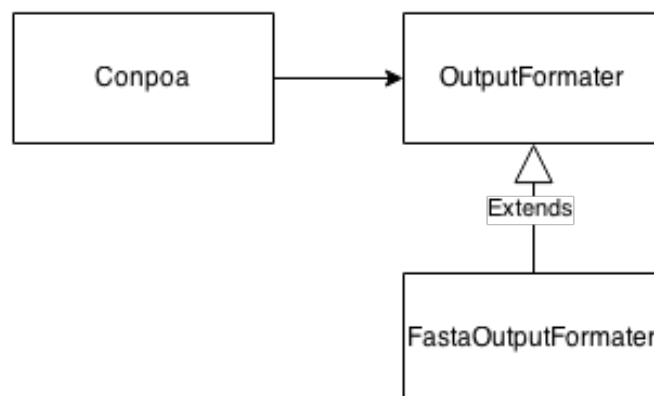
17         mylock->unlock ();
18         return 1;
19     }
20     return 0;
21     }));
22 }
23 }
24 delete mylock;
25 for (auto &&result: results) {
26     cnt += result.get ();
27 }
28 return cnt;
29 }

```

4.5. Generiranje izlaza

Kao izlazni format odabran je FASTA dokument sa svim konsenzusnim sljedovima. Poredani su onim redoslijedom kojim su nastajali i u naslovu je označeno koliko sljedova je grupirano uz svaki konsenzusni slijed.

Mogućnost dodatnog proširivanja funkcionalnosti bila je glavna ideja pri izradi programske implementacija. Da bi se dodao novi izlazni format potrebno je dodati klasu koja ga predstavlja i implementira sučelje *OutputFormater*. Odnos klasa koje se koriste za stvaranje izlaza prikazan je na Slici 4.7.



Slika 4.7: Odnos komponenti za stvaranje izlaza

Programski isječak 4.3: Generiranje FASTA izlaza

```
1  std::ofstream outfile(file_path);
2  for (Seq *cons : poMsa->consensuses()) {
3      outfile << ">" << cons->name() << "_" << cons->title() << std::endl;
4      std::list<Node *> nodes;
5      cons->GetNodes(&nodes);
6      for (Node *node: nodes) {
7          outfile << node->nucl();
8      }
9      outfile << std::endl;
10 }
```

4.6. Vremenska i memorijska složenost

Za dobivanje konsenzusnog slijeda najznačajnija je operacija izračuna vrijednosti svakog čvora i pripadnih najboljih prethodnika. Prolazak kroz graf ima složenost $O(N)$, gdje je N broj čvorova u grafu, pošto se svaki čvor treba posjetiti točno jednom. Treba uočiti da povećanje dužine pojedinačnih sljedova u poravnanju ne mora nužno povećati vrijeme izvođenja, pošto je moguće da sve nove baze budu uključene u već postojeće čvorove.

Svaki čvor koji obiđemo potrebno je obraditi, čime se složenost povećava na $O(N, n_{avg})$, gdje je n_{avg} prosječni broj ulaznih veza u čvor. Za samo dobivanje konsenzusa svaki čvor treba sadržavati podatak o njegovom najboljem prethodniku, no zbog grupiranja sljedova s konsenzusom također treba sadržavati informacije o sadržanim sljedovima i čvorovima s kojima je poravnat.

5. Testni primjeri

Iako vremenska i memorijska složenost ovise o broju čvorova u grafu, mnogo je zanimljivija ovisnost o karakteristikama samih sljedova. Ostvarena implementacija testirana je na većem skupu poravnanja, koja su se razlikovala po duljini referentnog slijeda i pokrivenosti pojedinih nukleotida. Da bi se postigla željena prosječna pokrivenost nukleotida povećavao se broj manjih sljedova. Svi manji sljedovi su bili duljine 100, što odgovara Illumina sljedovima. U nastavku su prikazani rezultati izvođenja gdje se samo jedan parametar mijenjao, čime se htjelo utvrditi ovisnost o tom parametru.

Kako bi podaci bili usporedivi, generiranje je ograničeno na samo dva konsenzusna slijeda (najznačajniji). Broj generiranih konsenzusa ne utječe na memoriju, barem ne primjetno jer se za svaki novi konsenzus dodaje samo onoliko pokazivača koliko je dugačak.

Svi testovi izvedeni su na računalu s 16GiB RAMa i 4 jezgrenom Intel® Core™ i7-4500U CPU @ 1.80GHz procesorom koristeći 2 dretve.

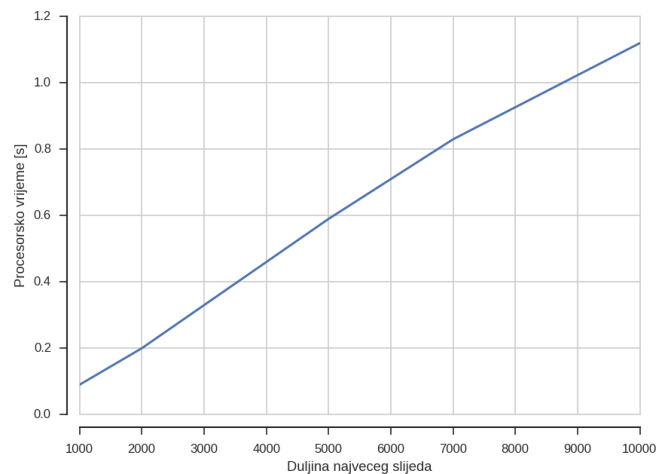
5.1. Ovisnost o duljini referentnog slijeda

Za provjeru performansi ovisno o duljini referentnog slijeda upotrebljene su duljine referentnog slijeda od 1000 do 10000 baza, dok je pokrivenost pojedinog nukleotida bila stalna. Da bi pokrivenost nukleotida bila jednaka broj manjih sljedova mijenjao se ovisno o duljini referentnog slijeda, kao što je vidljivo u Tablici 5.1.

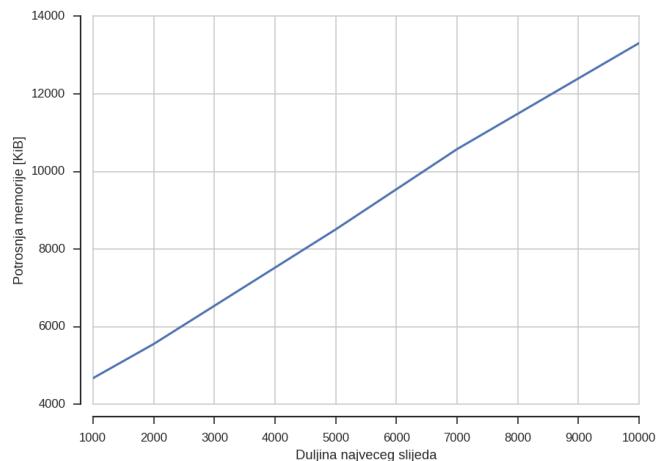
Tablica 5.1: Podaci korišteni u analizi ovisnosti o duljini slijeda

duljina referentnog slijeda	prosječan broj manjih sljedova	Pokrivenost nukleotida
1 000	50	5
2 000	100	5
5 000	250	5
7 000	350	5
10 000	500	5

Iz grafova sa Slika 5.1 i 5.2 može se zaključiti da povećanje duljine referentnog slijeda duljina izvođenja i memorijska potrošnja linearno raste. Razlog tome je što povećavanjem referentnog slijeda povećavamo i broj čvorova. Uz to, da bi se održala jednaka pokrivenost, potrebno je povećati broj manjih sljedova.



Slika 5.1: Vrijeme provedeno na procesoru ovisno o duljini referentnog slijeda



Slika 5.2: Memorijska potrošnja ovisno o duljini referentnog slijeda

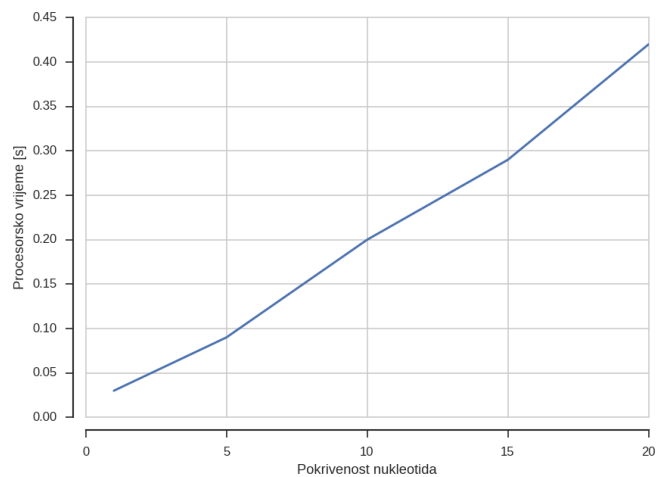
5.2. Ovisnost o pokrivenosti nukleotida

Za provjeru performansi ovisno o prosječnoj pokrivenosti nukleotida upotrebljena je duljina referentnog slijeda od 1000 baza, a pokrivenost je postepeno rasla od 1 do 20, kao što je vidljivo u Tablici 5.2.

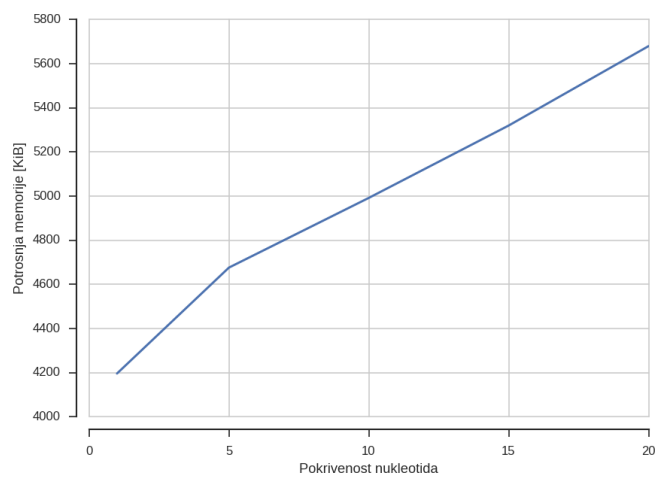
Tablica 5.2: Podaci korišteni u analizi ovisnosti o duljini slijeda

duljina referentnog slijeda	prosječan broj manjih sljedova	Pokrivenost nukleotida
1 000	10	1
1 000	50	5
1 000	100	10
1 000	150	15
1 000	200	20

Povećanjem prosječne pokrivenosti pojedinog nukleotida povećali smo prosječni broj ulaznih veza u pojedini čvor, što je uzrokovalo povećanje iskorištenog vremena i memorije, kao što je vidljivo iz grafova sa Slika 5.3 i 5.4 .



Slika 5.3: Vrijeme provedeno na procesoru ovisno o duljini manjih sljedova

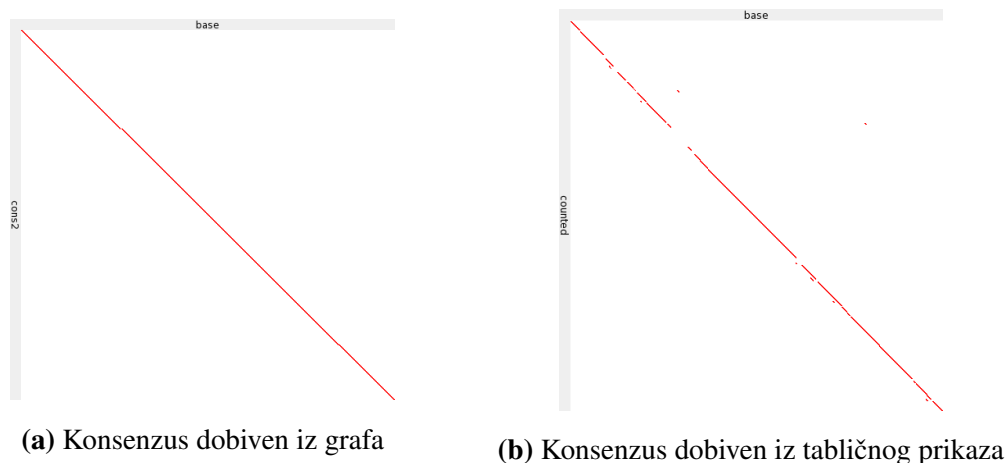


Slika 5.4: Memorijska potrošnja ovisno o duljini manjih sljedova

6. Analiza rezultata

Za provjeru valjanosti rezultata uzet je referentni slijed duljine 10000 sa prosječnom pokrivenošću nukleotida 5 (oko 500 manjih sljedova). Iz referentnog su slijeda sintetski generirana očitavanja duljine 100 po vjerojatnosnom modelu greške najbližijem Illumina sekvenceru. Kako su Illumina očitavanja poznata po velikoj točnosti, stvoreni bi konsenzus trebao u velikoj mjeri odgovarati referentnom slijedu, uz manja odstupanja na regijama gdje su manji sljedovi sa greškom prevladavali. Sa Slike 6.1a može se vidjeti da stvoreni konsenzus gotovo savršeno pokriva referentni slijed, kao što je i očekivano.

Sa jednakim podacima izvedeno je poravnanje i dobivanje konsenzusa iz tabličnog prikaza dobivenog alatom *ClustalW*. Kao što se vidi na slici 6.1b broj i veličina nepravilnih regija je veća. Ako se uzme u obzir da su u oba slučaja testni podaci bili jednaki, a konsenzus dobiven iz grafa nema većih odstupanja, može se zaključiti da greške u poravnanju nisu zbog grešaka u manjim sljedovima. Zaključujemo da je greška u poravnanju uzrokovana unošenjem rupa.



Slika 6.1: Usporedba poravnanja konsenzusa i referentnog slijeda

7. Zaključak

Opisani algoritam i ostvarena implementacija nude mogućnost stvaranja više konsenzusnih sljedova iz višestrukog poravnanja sljedova prikazanih usmjerenim grafom. No, predloženi algoritam ima svojih mana.

Sam rad algoritma ovisi o velikom broju parametara koje korisnik mora zadati, kao što su faktor smanjenja težine slijeda u poravnanju i parametri korišteni u grupiranju sljedova s konsenzusom. Bilo bi puno bolje kad bi se parametri, kao što su najveća moguća rupa i najveća moguća duljina ne poravnatog ruba, računali ovisno o ulaznim podacima, a ne ovisili o korisnikovom odabiru.

Unatoč tome, uz dobro zadane parametre, moguće je dobiti više informacija iz višestrukog poravnanja sljedova (i to bez gubitaka podataka) što je napredak nad uobičajeno korištenim metodama.

Dodatak

A. Format zapisa grafa

Kako bi se mogao dalje analizirati graf je potrebno spremi u datoteku. *POA* nudi ispis u format *.po* koji je zapravo zapis grafa. Sastoji se od tri cjeline:

1. Podaci o alatu
2. Podaci o sljedovima
3. Podaci o čvorovima

A.1. Podaci o alatu

Na početku datoteke nalaze se podaci o samom alatu i grafu. Ovdje se navode podaci o verziji alata, imenu i naslovu grafa i broju čvorova i sljedova u samom poravnanju. Svaki podatak navodi se u zasebnom redu. Ime i naslov poravnanja preuzimaju ime i naslov prvog slijeda, ako nije drugačije navedeno. Primjer zaglavlja bio bi:

```
VERSION=LPO.1.0  
NAME=ime poravnanja  
TITLE=naslov poravnanja  
LENGTH=32120  
SOURCECOUNT=126
```

A.2. Podaci o sljedovima

Za svaki slijed iz grafa navode se ime i detaljnije informacije. Redom su to : broj čvorova, indeks prvog čvora (vezano uz sljedeće potpoglavlje), težina slijeda, indeks pripadne grupe i naslov slijeda. Ukoliko vrijednost nekog indeksa nije poznata bit će zapisana vrijednost -1 . Primjer podataka o nizovima bio bi:

```
SOURCENAME=ime slijeda
SOURCEINFO=217 10 0 3 naslov slijeda
SOURCENAME=ime drugog slijeda
SOURCEINFO=432 24 4 -1 drugi naslov
```

A.3. Podaci o čvorovima

Za svaki čvor iz grafa navodi se pripadno slovo i svi podaci o prethodnom čvoru (L), sadržanim sljedovima (S) i poravnatim čvorovima (A), gdje su sljedovi i čvorovi prikazani indeksom, odnosno, rednim brojem u dokumentu (počevši od 0). Čvor ne smije biti naveden prije svih njegovih prethodnika. Primjer isječka podataka o čvorovima bio bi:

```
A:S14
A:L382S14
G:L381S21A388
A:L381S1S18A384
A:L381S27S48A385
```

LITERATURA

- [1] Mark F. Sharlow Christopher Lee, Catherine Grasso¹. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18:452, 464, October 2001.
- [2] Jonathan Dursi. Understanding partial order alignment for multiple sequence alignment, May 2015. URL <http://simpsonlab.github.io/2015/05/01/understanding-poa/>.
- [3] Christopher Lee. Generating consensus sequences from partial order multiple sequence alignment graphs. *Bioinformatics*, 19:999, 1008, December 2002.
- [4] Mirjana Domazet-Lošo Mile Šikić. *Bioinformatika*. Prateći materijal za predmet Bioinformatika, Fakultetu elektrotehnike i računarstva, 2013.
- [5] D. Stott Parker. Pairwise partial order alignment as a supergraph problem - aligning alignments revisited. *Journal of Bioinformatics and Computational Biology*, October 2003.
- [6] Saul B. Needleman, Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(7011):443–453, 1970.

Generiranje konsenzusnog slijeda iz grafova djelomično uređenih višestrukih poravnanja

Sažetak

Za razliku od uobičajenih metoda dobivanja konsenzusa (koja mogu prikazati akcije supstitucije i dodavanja/brisanja), koristeći usmjerene grafove moguće je prikazati složenije promjene u evoluciji slijeda, kao što su translacija domene, duplikacija i rekombinacija. Implementacija ovog rada nudi mogućnost stvaranja više konsenzusnih slijedova iz grafova djelomično uređenih višestrukih poravnanja koristeći format zapisa programskog alata *POA*. Stvoreni konsenzusi nisu ograničeni na lokalne promjene zbog veće mogućnosti prikaza informacija pomoću usmjerenih grafova.

Ključne riječi: usmjereni graf, višestruko poravnanje, *POA*, konsenzus

Generating consensus sequence using partially ordered multiple sequence alignment graphs

Abstract

Unlike conventional methods of obtaining consensus (which can only display changes like substitution and addition / deletion), the use of partial order multiple sequence alignment graphs allows more complex changes in the sequence evolution, such as translation of a domain , duplication and recombination. Implementation shown in this work offers the possibility to create multiple consensus sequences from graphs using the file format of the programming tool *POA*. The resulting consensus are not limited to local changes, due to greater level of information stored in graphs used to represent multiple sequence alignment.

Keywords: ordered graph, multiple alignment, *POA*, consensus