Bioinformatičke metode sastavljanja genoma de novo

Glavaš, Dunja

Undergraduate thesis / Završni rad

2014

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:611407

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**



Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb







SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

BIOLOŠKI ODSJEK

BIOINFORMATIČKE METODE SASTAVLJANJA GENOMA *DE NOVO*

BIOINFORMATICS METHODS OF *DE NOVO* GENOME ASSEMBLY

Dunja Glavaš

Preddiplomski studij molekularne biologije

(Undergraduate study of Molecular Biology)

Mentor: prof. dr.sc. Kristian Vlahoviček

Zagreb, 2014.

Sadržaj

1. Uvod 2

2. Problem sastavljanja genoma 3

3. Programi za sastavljanje genoma 4

3.1. Pohlepni algoritam 5

3.2. Metoda OLC 7

3.3. de Bruijnov graf 9

4. Sklapanje okvira 11

5. Zaključci 12

6. Literatura 13

7. Programi za sastavljanje spomenuti u radu 15

8. Sažetak 16

9. Summary 16

1. Uvod

U biologiji, odgovori na različita pitanja sve češće se traže na staničnoj razini, proučavanjem gena i genoma. Razvoj tehnike sekvenciranja omogućio je istraživačima da odrede slijed nukleotida u molekuli DNA, dok brojne računalne analize dobivenog slijeda pomažu u donošenju zaključaka u mnogim pravcima istraživanja, od porijekla organizma do funkcije proteina. Visoka znanstvena vrijednost dobivenih podataka vidljiva kroz brojna značajna otkrića potakla je daljnji napredak tehnologije sekvenciranja. Sekvenciranje sljedeće generacije (eng. *next generation sequencing*, NGS) pruža brz i jeftin način da se „pročita" molekula DNA.

Međutim, sekvenatori (uređaji za sekvenciranje), a posebno sekvenatori sljedeće generacije, nisu u stanju očitati cijelu molekulu DNA odjednom pa su „sirovi" rezultati takvih eksperimenata brojna očitanja tek kratkih fragmenata cijele sekvence (eng. *reads*). Sami po sebi, takvi fragmenti nude vrlo malo informacija. Bilo kakva kasnija analiza zahtijeva potpunu sekvencu gena ili genoma. Kako bi se rekonstruirala cijela sekvenca, očitani fragmenti moraju biti poredani pravilnim redoslijedom i u ispravnom broju kopija. Dakle, podatci dobiveni sekvenciranjem moraju biti podvrgnuti računalnoj obradi – postupku sastavljanja genoma (eng. *genome assembly*). Sve niža cijena i sve veća brzina sekvenciranja dovele su do toga da upravo postupak sastavljanja genoma postaje vremenski ograničavajući faktor u istraživanjima genoma. Razvoj učinkovitih programa za sastavljanje genoma (eng. *assemblers*) time dobiva na važnosti.

Cilj ovog rada je ponuditi pregled danas najznačajnijih programa za sastavljanje, te ukratko objasniti način na koji funkcioniraju.

2. Problem sastavljanja genoma

Sastavljanje genoma možemo definirati kao rekonstrukciju slijeda DNA iz kolekcije nasumično uzorkovanih fragmenata (Narzisi i Mishra 2011). Ukoliko je genomski slijed srodne vrste otprije poznat, zadatak sastavljanja je olakšan pretpostavkom da su sljedovi dva genoma slični. Tada otprije poznati srodni slijed može poslužiti kao kalup za sastavljanje nepoznatog genoma. Ovakav postupak naziva se komparativno sastavljanje genoma (eng. *comparative genome assembly*). Međutim, ukoliko referentni slijed nije na raspolaganju, zadatak postaje mnogo teži i potrebno je genom sastaviti postupkom *de novo*. Važno je napomenuti da oba pristupa (komparativni i *de novo*) nisu međusobno isključiva. Čak i kod vrlo srodnih organizama, razlike između genoma, ili njihovih pojedinih dijelova, mogu biti takve da zahtijevaju sastavljanje *de novo*.

Matematički gledano, sastavljanje genoma metodama *de novo* pripada u tzv. NP-tešku klasu problema – za njih nije učinkovito moguće pronaći jednoznačno računalno rješenje. Zbog toga se programi za sastavljanje moraju oslanjati na heuristiku i aproksimacije. Takav pristup zahtijeva visoku specifičnost programa. Izbor algoritama i protokola koji će ispravno sastaviti genom ovisi o velikom broju faktora: izvoru „sirovih“ podataka dobivenih sekvenciranjem (različite platforme za sekvenciranje proizvode očitane fragmente različitih duljina i pouzdanosti); duljini genoma; broju i učestalosti ponavljajućih sljedova; kasnijoj primjeni dobivenih podataka (različite analize imaju različitu osjetljivost na određene pogreške) i tako dalje. Sve to otežava razvoj univerzalnih alata za sastavljanje genoma *de novo* te zahtijeva veću stručnost korisnika u polju bioinformatike.

Nadalje, nijedna eksperimentalna tehnika nije nepogrešiva, pa tako ni sekvenciranje nije iznimka. Svaka platforma za sekvenciranje ima specifičan obrazac eksperimentalnih pogrešaka, koje moraju biti prepoznate i uklonjene da bi genom mogao biti uspješno sastavljen. Različite vrste ponavljanja prisutne u genomu predstavljaju dodatnu prepreku, kao i individualne razlike u slijedu DNA između jedinki koje pripadaju istoj vrsti (haplotipovi).

I najučinkovitiji programi za sastavljanje imaju svoja ograničenja. Neovisno o veličini genoma, rezultat slaganja kratkih očitanih fragmenata uvijek je isprekidani slijed, odnosno skup više odvojenih dijelova sekvence. Dobivene sljedove potrebno je poredati ispravnim redoslijedom i procijeniti veličinu prekida među njima. Proces kojim se to postiže naziva se

sklapanje okvira (eng. *scaffolding*). Nakon toga provode se dodatni eksperimenti sekvenciranja kako bi se popunili prekidi i time dovršila potpuna genomska sekvenca.

3. Programi za sastavljanje genoma

Programi za sastavljanje moraju imati određenu specifičnost u ovisnosti o porijeklu podataka. Zbog toga je paralelno s razvojem većeg broja tehnologija sekvenciranja razvijen i veći broj programa za sastavljanje sekvenciranih sljedova. Neovisno o tome jesu li prilagođeni za obradu podataka dobivenih metodom sekvenciranja po Sangeru (koji se sastoje od manjeg broja očitanih fragmenata veće duljine) ili sekvenciranjem slijedeće generacije (koje proizvodi kraće očitane fragmente u većem broju), programi za sastavljanje se generalno mogu podijeliti u dvije skupine s obzirom na to kako predstavljaju i obrađuju očitane sljedove: kao nizove znakova (eng. *string-based*) ili kao usmjereni graf (eng. *graph-based*) (Zhang i sur. 2011).

Programi za sastavljanje zasnovani na nizovima znakova sljedove prikazuju kao nizove znakova (četiri različita slova, A, T, C i G), pri čemu svaki znak predstavlja bazu jednog nukleotida u sekvenci DNA. Nizovi se zatim međusobno uspoređuju i sravnjuju, a preklapajući nizovi se spajaju. Na taj je način moguće rekonstruirati slijed dulji od pojedinačnog niza tj. očitanog fragmenta. Tako sastavljena sekvenca naziva se *contig*. Najznačajnija implementacija *string-based* pristupa je pohlepni (eng. *greedy*) algoritam.

Drugi općeniti pristup zasniva se na primjeni grafa kao oblika u kojem se podatci prikazuju i analiziraju (*graph-based* slagači). Graf je matematička apstrakcija koja je našla široku primjenu u računalnoj znanosti, pa tako i u bioinformatici. Sastoji se od skupa točaka (čvorova) i bridova između tih točaka. Čvorovima i bridovima mogu biti pridružene dodatne informacije – u slučaju slaganja genoma, polarnost lanca i duljina očitanog fragmenta, duljina i tip preklapanja, komplementarnost lanaca itd. Prikazivanjem podataka u dvodimenzionalnom sustavu grafa omogućene su analize i heurističke transformacije koje u jednodimenzionalnom sustavu nizova nisu moguće, te je problem sastavljanja genoma sveden na pojednostavljivanje grafa i traženje puta kroz čvorove i bridove koji zadovoljava određene uvjete. Većina današnjih programa za sastavljanje oslanja se na ovaj princip. U ovisnosti o tome kako su podatci pridruženi čvorovima i bridovima (čvorovi mogu predstavljati očitane fragmente a bridovi preklapanja između njih i obrnuto), možemo razdvojiti programe za

sastavljanje zasnovane na usmjerenom grafu u dvije grupe: oni koji koriste metodu OLC i oni koji koriste de Bruijnov graf.

U nastavku rada slijedi pregled tri najznačajnije metode (pohlepni algoritam, OLC i de Bruijn), te kratki osvrt na proces sklapanja okvira.

3.1. Pohlepni algoritam

Pohlepni algoritam je najjednostavnije, intuitivno rješenje problema sastavljanja genoma. Nad skupom nizova (očitanih fragmenata) provode se operacije koje za cilj imaju pronalazak najkraćeg zajedničkog superniza (eng. *shortest common superstring*, SCS) koji sadrži sve moguće preklopljene očitane fragmente i u teoriji predstavlja cijeli slijed DNA.

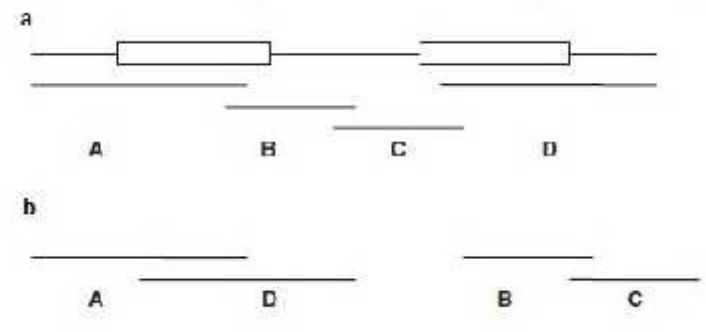
Ključni dio ovog algoritma je modul nazvan *overlapper*. On sravnjuje očitane fragmente u parovima, svakog sa svakim, i boduje sravnjenja. Shema bodovanja ovisi o konkretnoj implementaciji algoritma, a najčešće se boduju duljina preklapanja i stupanj identičnosti baza u preklopljenim dijelovima sekvence. Ovaj početni stupanj obrade podataka ujedno je i najzahtjevniji u smislu vremena i potrebne radne memorije računala. Nastavak procesa može se podijeliti u tri osnovna koraka:

- Izabiru se preklapanja s najvećim brojem bodova.
- Preklopljeni fragmenti se spajaju u dulju sekvencu i u tom obliku vraćaju u početni skup fragmenata.
- Izabire se sljedeće najbolje preklapanje i koraci se iterativno ponavljaju dokle god su moguća pouzdana preklapanja i spajanja.

Dodatno, područje preklapanja se nakon svakog spajanja heuristički ispravlja. Regije u kojima to nije moguće (npr. regija je sekvencirana samo jednom tj. predstavljena je samo jednim očitanim fragmentom) u izlaznim podatcima prikazuju se kao prekidi (eng. *gaps*). Program vraća samo jedno moguće rješenje u obliku isprekidanog slijeda, odnosno skupa *contig*-a.

Pohlepni algoritam svoje ime duguje činjenici da zbog načina rada daje prednost lokalnim maksimumima, koji ne vode nužno do globalno optimalnog rješenja. Stoga je pohlepni algoritam prikladniji kod sastavljanja manjih genoma. Također, u osnovnom obliku *overlapper* modula, vrijeme potrebno za obradu podataka eksponencijalno se produljuje s povećanjem broja očitanih fragmenata. Većina programa koji se oslanjaju na pohlepni

algoritam tako je razvijena za sastavljanje manjeg broja duljih fragmenata dobivenih sekvenciranjem metodom po Sangeru. Primjeri su phrap, TIGR Assembler i CAP3.



Slika 1. (**a**) Slijed koji sadrži ponavljanja ispravno složen iz očitanih fragmenata A, B, C i D. Pravokutnici predstavljaju ponavljanja. (**b**) Slijed koji sadrži ponavljanja neispravno složen primjenom pohlepnog algoritma (daje se prednost lokalnom optimumu). Preuzeto i prerađeno iz Pop, 2009.

Uz „klasičan“ pohlepni algoritam, razvijeni su i programi za sastavljanje očitanih fragmenata dobivenih sekvenciranjem sljedeće generacije koji koriste istu strategiju. *Overlapper* je prilagođen kako bi se poboljšala efikasnost procesa. To se postiže paralelizacijom procesa u multiprocesorskim računalima ili računalnoj mreži i/ili tehnikama indeksiranja koje prethode sravnjivanju fragmenata (Fragmenti se razlome na kraće dijelove određene duljine (*k*-meri). Ukoliko dva fragmenta ne dijele ni jedan zajednički *k*-mer, smatra se da ih nije potrebno sravnjivati.). Dodatnom prilagodbom programa moguće je dobiti nefragmentirane sljedove dulje od onih koje sastavlja običan pohlepni algoritam. Očitani se fragmenti preklapaju i produljuju prvo na jednom kraju, 3’ ili 5’, a potom na drugom koristeći reverzni komplement preklopljenih fragmenata kao polazište. Primjeri takvih prilagođenih programa za sastavljanje su SSAKE (Warren i sur. 2007), VCAKE (Jeck i sur. 2007) i SHARCGS (Dohm i sur. 2007). Na kraju, iako se pohlepni algoritam tipično koristi u programima zasnovanima na nizovima znakova, razvijene su i implementacije tog algoritma koje podatke prikazuju i obrađuju u obliku grafa.

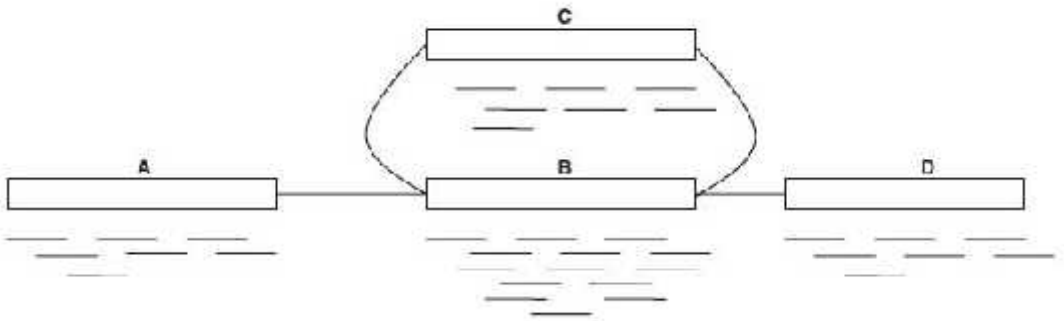
3.2. Metoda OLC

OLC (eng. *overlap-layout-consensus*) je metoda koja omogućava globalnu analizu odnosa između očitanih fragmenata, za razliku od lokaliziranog pristupa pohlepnog algoritma. Iako su temeljene na različitim principima, dvije metode imaju nešto zajedničko – *overlapper* modul. Kao i u pohlepnom algoritmu, početna faza kod metode OLC je sravnjivanje i formiranje liste najboljih preklapanja. Pomoću liste se zatim oblikuje graf na način da je svaki očitani fragment predstavljen jednim čvorom, a bridovi spajaju sve čvorove između kojih je utvrđeno preklapanje. Takav prikaz svodi problem sastavljanja genoma na pronalazak puta kroz graf koji prolazi svakim čvorom točno jednom (Hamiltonov put). Općeniti slučaj traženja Hamiltonovog puta pripada u klasu NP-teških problema, pa je potrebno pojednostaviti graf putem heurističkih transformacija i aproksimacija.

Putovi kroz čvorove i bridove inicijalnog grafa mogu i ne moraju biti jednoznačni. Kad čvor s obje strane, ulazno i izlazno, ima samo po jedan brid, radi se o jednoznačnom putu koji predstavlja dio sekvence oko čijeg slaganja nema dvojbi. Drugim riječima, jednoznačni put kroz čvorove odgovara neprekinutom genomskom slijedu koji je sigurno prisutan u konačnoj sekvenci. Takav put naziva se *unitig* (eng. *uniquely assemblable contig*). Međutim, u grafu se javljaju grananja na mjestima gdje se jedan kraj nekog očitanog fragmenta preklapa s krajevima dva ili više drugih očitanih fragmenata, dok na suprotnom kraju tih drugih fragmenata nema preklapanja. Rezultat je čvor sa višestrukim bridovima. Put kroz čvor koji ima više ulaznih ili izlaznih bridova nije jednoznačan, odnosno moguće je proći takav čvor s više različitih putova. Grananja najčešće predstavljaju granicu između ponavljanja i genomskog slijeda susjednog ponavljanju ili su uzrokovana pogreškama pri sekvenciranju.

Program analizira graf hijerarhijski na način da prvo pronalazi sve *unitig*-ove. U trenutku kad algoritam susretne prvo grananje, produljivanje *unitig*-a se zaustavlja. Iako su složeni *unitig*-ovi vrlo pouzdani i pogreške unutar njih su rijetke, za većinu primjena nisu dovoljni. Grananja u grafu su česta pa su *unitig*-ovi u pravilu kratki. Produljenje nefragmentiranih sljedova postiže se primjenom heurističkih metoda. Velik broj očitanih fragmenata kod sekvenciranja sljedeće generacije omogućava otkrivanje pogrešaka kod sekvenciranja primjenom statistike. Ako je neki dio sekvence predstavljen mnogobrojnim očitanim fragmentima, većina tih fragmenata neće sadržavati pogrešku, pa je one u kojima je došlo do pogreške moguće detektirati i zatim ukloniti. Time se uklanja i pripadajuće grananje

u grafu te se *unitig* produljuje. Što se drugih grananja tiče, nerazmjerno velik broj očitanih fragmenata koji odgovaraju nekom slijedu DNA je indikator ponavljanja – dotični slijed je u genomu vjerojatno prisutan u više kopija.



Slika 2. Graf preklapanja genomske regije oblika ABCBD koja sadrži dvije kopije slijeda B. Treba primijetiti veći broj očitanih fragmenata koji odgovaraju području slijeda B. Preuzeto iz Pop, 2009.

Većina programa koji koriste metodu OLC u kasnijoj fazi ugrađuju i podatke dobivene drugim tipovima eksperimenata, npr. sekvenciranjem uparenih krajeva (eng. *mate-pair*). Na taj način se *unitig*-ovi svrstavaju u skupove koji se mogu spojiti u veće strukture. Primjeri su Celera (Myers i sur. 2000) i Arachne Assembler (Batzoglou i sur. 2002).

Po nekima, programi za sastavljanje koji se oslanjaju na OLC danas su najzastupljeniji i najuspješniji (Pop 2009). Glavna prednost metode je fleksibilnost. Programi su modularni, prikladni za više tipova analiza i dobro prilagođeni na slaganje genoma iz vrlo kratkih očitanih fragmenata.

3.3. de Bruijnov graf

Strategija de Bruijnovog grafa nekad se naziva i metodom Eulerovog puta. Konceptualno, algoritam je sličan danas zastarjeloj metodi sekvenciranja hibridizacijom (eng. *sequencing by hibridization*, SBH). SBH je napušten zbog velike eksperimentalne zahtjevnosti i visoke cijene pokusa. Strategija de Bruijnovog grafa računalnim metodama proizvodi skup podataka sličan rezultatu SBH pokusa koristeći jeftinije i brže platforme sekvenciranja sljedeće generacije.

Program razlama skup očitanih fragmenata na kratke sljedove točno određene duljine *k*, koja je kraća od duljine pojedinačnog očitanog fragmenta. Takvi kratki sljedovi nazivaju se *k*-meri. Dobiveni spektar *k*-mera je lista svih mogućih oligomera duljine *k*, prisutnih u genomu od interesa. Na neki način, spektar *k*-mera je nalik podatcima koje bismo dobili teoretski savršenim sekvenciranjam: svi fragmenti su jednake duljine i savršeno uzorkuju genom (svaka baza u sekvenci odgovara početku točno jednog *k*-mera). Dodatna prednost ovakvog pristupa je ušteda memorije - svaki različiti *k*-mer se sprema samo jednom, a uz njega se pohranjuje informacija o tome koliko puta je prisutan u genomu - dakle, program ne troši memoriju na spremanje duplikata.

Iz dobivenog seta podataka zatim se konstruira de Bruijnov graf. Čvorovi predstavljaju prefikse i sufikse svih *k*-mera duljine (*k*-1). Brid između dva čvora postoji ako dotični (*k*-1)-meri imaju savršeno preklapanje duljine (*k*-2). Na taj način svaki je *k*-mer prikazan jednim bridom u grafu i problem sastavljanja genoma se svodi na problem pronalaska puta koji prolazi svakim bridom točno jednom (Eulerov put). Ovakav algoritam zaobilazi dio obrade koji je u druge dvije metode (pohlepnom algoritmu i metodi OLC) vremenski najzahtjevniji. Preklapanja između očitanih fragmenata se nikad eksplicitno ne računaju već su implicitno prikazana grafom kao bridovi između *k*-mera jednog očitanog fragmenta i *k*-mera drugog očitanog fragmenta.

Poznati su efikasni algoritmi za pronalaženje Eulerovog puta kroz graf, što nije slučaj kod Hamiltonovog puta. Međutim, broj mogućih Eulerovih putova eksponencijalno raste s brojem čvorova u grafu pa ni ovaj problem nije trivijalan i zahtijeva primjenu heuristike. Potrebno je definirati brojna ograničenja i dodatne uvjete kako bi se algoritam naveo na pravi put - što znatno otežava izračunavanje. Također, razlamanje očitanih fragmenata na *k*-mere koje prethodi konstrukciji de Bruijnovog grafa rezultira gubitkom informacija o povezanosti

udaljenijih područja. Zbog toga je prvi korak nakon konstrukcije grafa vraćanje podataka o originalnim očitanim fragmentima. Program prvo povezuje *k*-mere porijeklom iz istog očitanog fragmenta. Svaki takav rekonstruirani fragment tvori kraći Eulerov put, koji ne prolazi cijelim grafom već samo njegovim dijelom. Problem izračunavanja se sada može definirati kao pronalazak Eulerovog superputa - kompletnog Eulerovog puta koji prolazi kroz graf koristeći sve Eulerove pod-putove (koji predstavljaju očitane fragmente).

Bitno je napomenuti da svaka pogreška u sekvenciranju stvara veći broj „lažnih“ *k*-mera. Zbog toga je metoda de Bruijnovog grafa prikladnija za obradu podataka dobivenih sekvenciranjem sljedeće generacije, gdje je zbog velikog broja očitanih fragmenata moguće statistički ukloniti pogreške. Primjeri programa koji se oslanjaju na de Bruijnov graf su Euler (Pevzner i sur. 2001), Velvet (Zerbino i Birney 2008), ABySS (Simpson i sur. 2009), AllPaths (Butler i sur. 2008) i SOAPdenovo (Li i sur. 2009).

4. Sklapanje okvira

Kao što je prije spomenuto, izlazni podatci programa za sastavljanje uvijek su u obliku fragmentiranog slijeda, odnosno skupa nezavisnih *contig*-a. Pošto je cilj slaganja genoma dobiti što potpuniju i što cjelovitiju sekvencu, izlazni podatci se podvrgavaju i dodatnoj obradi. Sklapanje okvira je proces kojim se određuje relativna pozicija nezavisnih *contig*-a unutar genoma, odnosno utvrđuje se položaj *contig*-a s obzirom na druge *contig*-e u smislu udaljenosti i redoslijeda.

Sklapanje okvira se najčešće oslanja na podatke dobivene pokusima sekvenciranja uparenih krajeva (*mate-pair*). Za dva *contig*-a se može pretpostaviti da su susjedni ako se jedna strana uparenog kraja preklapa s jednim *contig*-om a druga strana s drugim *contig*-om. U praksi se za potvrdu odnosa traže barem dvije takve veze kako bi se smanjio utjecaj eventualne eksperimentalne pogreške. Alternativno, sklapanje okvira se može provoditi pomoću mapiranja čitavog genoma odjednom, primjerice optičkim mapiranjem. Dva pristupa se nadopunjuju - upareni krajevi nude visoku rezoluciju ali su lokalno ograničeni, dok optičko mapiranje nudi pregled nad čitavim genomom u niskoj rezoluciji. Set *contig*-a može se pomoću optičke mape „usidriti“ na ispravno mjesto u genomu, a detaljniji odnosi unutar takve strukture mogu se odrediti preko podataka o uparenim krajevima.

Algoritmi za sklapanje okvira često rade na pohlepnom principu. Povezivanje *contig*-a u veće strukture prvo se provodi na temelju najpouzdanijih informacija; ostali podatci se kasnije iterativno uklapaju dok ne dođu u konflikt s već konstruiranim slijedom. Većina modernih programa za sastavljanje genoma ima ugrađen i modul za sklapanje okvira, a razvijeni su i samostalni programi za sklapanje okvira.

5. Zaključci

Sastavljanje genoma *de novo* je složen i nimalo trivijalan zadatak. Napredak u tehnikama sastavljanja potaknut je potrebom za većom efikasnošću procesa zbog rastućeg broja istraživanja genoma. Od brojnih razvijenih programa za sastavljanje, većina ih se oslanja na jedno od tri načela: pohlepni algoritam, metodu OLC ili strategiju de Bruijnovog grafa. Zbog karakteristika različitih algoritama koje koriste, programi su bolje prilagođeni na određene setove ulaznih podataka. Pohlepni algoritam daje prednost lokalnim maksimumima pa je stoga prikladniji za sastavljanje kraćih genoma. Programi koji koriste pohlepni algoritam razvijeni su za obradu podataka dobivenih sekvenciranjem metodom po Sangeru i najčešće su zasnovani na nizovima znakova. Metoda OLC i strategija de Bruijnovog grafa koriste se u programima za sastavljanje zasnovanima na usmjerenom grafu. Razlikuju se po načinu prikaza očitanih fragmenata i preklapanja unutar grafa. Kod metode OLC čvorovi u grafu predstavljaju očitane fragmente a bridovi preklapanja između fragmenata. U strategiji de Bruijnovog grafa je obrnuto - bridovi odgovaraju sljedovima i čvorovi preklapanjima. Obje metode zasnovane na usmjerenom grafu razvijene su kao alati za obradu izlaznih podataka sekvenciranja sljedeće generacije, s tim da je po nekima metoda OLC bolje iskoristiva zbog modularnosti i fleksibilnosti.

6. Literatura

Batzoglou S. i sur., 2002. ARACHNE: a whole-genome shotgun assembler. *Genome* Res. **12**, 177-189.

Butler J. i sur., 2008. ALLPATHS: de novo assembly of whole-genome shotgun microreads. *Genome* Res. **18**, 810-820.

Dohm J.C. i sur., 2007. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome* Res. **17**, 1697-1706.

Jeck W.R. i sur., 2007. Extending assembly of short DNA sequences to handle error. *Bioinformatics* **23**, 2942-2944.

Li R. i sur., 2009. De novo assembly of human genomes with massively parallel short read sequencing. *Genome* Res. **20**, 265-272.

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

Myers E.W. i sur., 2000. A whole-genome assembly of Drosophila. *Science* **287**, 2196-2204.

Narzisi G, Mishra B, 2011. Comparing De Novo Genome Assembly: The Long and Short of It. *PLoS ONE* **6**(4): e19175.

Pevzner P.A., Tang H., Waterman M.S., 2001. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A.* **98**, 9748-9753.

Pop M., 2009. Genome assembly reborn: recent computational challenges. *Briefings in bioinformatics*, **10**, 354-366.

Simpson J.T. i sur., 2009. ABySS: A parallel assembler for short read sequence data. *Genome* Res. **19**, 1117-1123.

Warren R.L. i sur., 2007. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* **23**, 500-501.

Ye C. i sur., 2012. Exploiting sparseness in *de novo* genome assembly. *BMC Bioinformatics* **13**(Suppl 6):S1.

Zerbino D.R., Birney E., 2008. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*. **18**, 821-829.

Zhang W. i sur., 2011. A Practical Comparison of *De Novo* Genome Assembly Software Tools for Next-Generation Sequencing Technologies. *PLoS ONE* **6**(3): e17915.

7. Programi za sastavljanje spomenuti u radu

AllPaths: ftp://ftp.broadinstitute.org/pub/crd/ALLPATHS/Release-LG/

CAP3: http://deepc2.psi.iastate.edu/aat/cap/capdoc.html

Euler: http://euler-assembler.ucsd.edu/portal/

SHARCGS: http://sharcgs.molgen.mpg.de/

SOAPdenovo: http://soap.genomics.org.cn/soapdenovo.html

VCAKE: http://sourceforge.net/projects/vcake/

Celera: http://sourceforge.net/projects/wgs-assembler/files/wgs-assembler/wgs-8.1/

ABySS: http://www.bcgsc.ca/platform/bioinfo/software/abyss

SSAKE: http://www.bcgsc.ca/platform/bioinfo/software/ssake

Arachne: http://www.broadinstitute.org/crd/wiki/index.php/Arachne_Main_Page

Velvet: http://www.ebi.ac.uk/~zerbino/velvet/

8. Sažetak

Tehnologije sljedeće generacije omogućile su brzo i jeftino sekvenciranje genoma. Programi za sastavljanje obrađuju izlazne podatke takvih eksperimenata. Pri sastavljanju cjelovite sekvence iz fragmentiranih sljedova oslanjaju se na neki od tri pristupa: pohlepni algoritam, metodu OLC ili de Bruijnov graf. Rad ukratko opisuje svaki od tri algoritma te proces sklapanja okvira, koji je sljedeći korak kod slaganja genoma.

9. Summary

Next-generation technologies have made genome sequencing cheap and fast. Genome assemblers are programs which process output data from such experiments. When computing a complete genomic sequence from many smaller fragments, they take one of the three main approaches: *greedy*, OLC or de Bruijn graph. This paper describes the main points in each of the three algorithms, as well as the *scaffolding* process, which is the next step in genome assembly.