Sveučilište u Zagrebu
Fakultet Elektrotehnike i Računarstva

Seminarski rad

Usporedba DNA sekvenci: Smith-Waterman algoritam

Goran Žužić

Voditelj: *Doc.dr.sc. Mile Šikić*

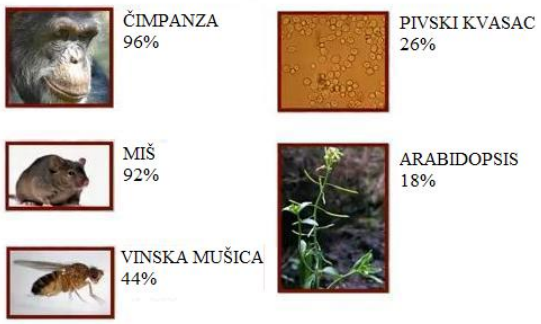
Zagreb, travanj, 2011.

Sadržaj

1 Uvod

Od kada je čovjek otključao tajnu genoma, otkrio je da sva živa bića (točnije, njihove stanice) posjeduju nekolicinu vrlo dugačkih lanaca molekula koji su među raznim organizmima zaprepaštujuće slični s obzirom da njihovi domaćini na prvi pogled nemaju nikakvog doticaja. Danas znamo da ti lanci određuju nasljedna svojstva svih živih bića i zovemo je DNA. Osnovni gradivni element DNA je molekula zvana nukleotid. Za potrebe ovog rada potrebno je napomenuti da se u DNA može naći točno četiri vrste nukleotida: *adenin* (A), *guanin* (G), *timin* (T) i *citozin* (C), kao što se može naći u [1].

DNA prosječne osobe sadrži otprilike 3 milijarde nukleotidnih parova organiziranih u 46 lanaca koje zovemo kromosomima. Nakon što se odrede vrste nukleotida u jednom od tih lanaca, njihove oznake (A, G, T ili C) mogu jednoznačno zapisati u sekvencu (sljed) i pohraniti na računalu [1]. Usporedbom takvih sekvenci DNA dobivenih od čovjeka sa sekvencama dobivenih od raznih drugih organizama dobivaju se iznenađujuće velike sličnosti. Slika 1 ilustrira rezultate do kojih je došao Institut za istraživanje ljudskog genoma iz SAD-a. [2]



Slika 1: Sličnost s ljudskom DNA

Postavlja se pitanje kako su gore iskazane sličnosti izračunate. No odgovor nije jednostavan. Štoviše, glavni cilj ovoga rada će biti da razradi kako matematički modelirati te potom kako najefikasnije izračunati sličnost između dvije DNA sekvence.

1.1 Mutacije

Da bi bolje razumjeli pitanje koliko su dvije DNA sekvence slične potrebno je proučiti kako se DNA može mijenjati kroz vrijeme. Evolucija unosi stalne i slučajne promjene u genetski kod pomoću djelovanja raznih fizikalnih i kemijskih procesa koje zovemo *mutacije*. Biolozi su u [3] izoliralli 5 vrsta mutacija:

1. Supstitucija
2. Ispušanje

[1] Bitno je primjetiti da bi navedene količine podataka koje se redovito mjere u milijunima predstavljale nesavladivu prepreku za bilo kakvu analizu bez pomoći računala.

3. Umetanje
4. Udvostručavanje
5. Inverzija

1.1.1 Supstitucija

Uvjerljivo najčešća mutacija je upravo supstitucija. Uzrokovana je kemikalijama i greškama u replikaciji DNA i očituje se u zamjeni jedne vrste nukleotida s drugim. Najčešće su bezazlene. Ukoliko se na nekom mjestu u DNA dogodila supstitucija, vjerojatnost da na to mjesto dođe točno određena vrsta novog nukleotida je gotovo jednaka.

A C G T T G A C
A C G A T G A C

Slika 2: Primjer substitucije

1.1.2 Ispuštanje i umetanje

Događa se kada se iz DNA ukloni ili doda *uzastopni* sljed nuklotida. Mala je vjerojatnost da će dotični dio DNA pravilno raditi ukoliko se dogodi ova vrsta mutacija zato što one promijene način na koji mehanizmi odgovorni za replikaciju čitaju podatke s DNA. Nasreću, događaju se osjetno rijeđe nego supstitucije.

A C G T T G A C
A C G A C

Slika 3: Primjer ispuštanja

1.1.3 Udvostručavanje i inverzija

Iako relativno bezazlene, ove mutacije su iznimno rijetke do te mjere da ih se u usporedbama sličnosti DNA sekvenci zanemaruje[1]. Unatoč tome, zanimljivo je napomenuti kako biolozi smatraju udvostručavanje kao jednu od najvažnijih metoda povećanja količine genetskih informacija u današnjim živim bićima. [4]

[1] Iako su obje mutacije rijetke, empirijski je utvrđeno da su udvostručavanja ipak malo češća od inverzija.

A C G T T G A C
A C G T T G A T T G A C

Slika 4: Primjer udvostručavanja

A C T C A A G G
A C A A C T G G

Slika 5: Primjer inverzije

2 Poravnavanje dvije DNA sekvence

Sada ćemo izgraditi matematički model za usporedbu dvije DNA sekvence. Dati ćemo ocjenu na količinu mutacija koje su potrebne kako bi jednu sekvencu preveli u drugu sekvencu. No mi iz prethodnog razmatranja znamo da nisu sve mutacije jednako vjerojatne, pa ćemo pretpostaviti da su dvije sekvence međusobno sličnije ukoliko je moguće jednu pretvoriti u drugu upotrebom mutacija koje se češće događaju u prirodi.

Kako bi pojednostavili analizu, u obzir ćemo uzeti samo substituciju, ispuštanje i umetanje nukleotida. Udvostručenja i inverzije su vrlo rijetke mutacije koje se mogu relativno precizno reprezentirati kao umetanje odnosno niz substitucija.

Formalni postupak usporedbe dvije DNA sekvence nazivamo *poravnavanje*. Termin podrazumijeva proizvoljno umetanje praznina u obe sekvence tako da po završetku dodavanja oni imaju jednaku duljinu. Potom se prva sekvenca pozicionira iznad druge tako da svakom simbolu prvog niza pridružimo točno jedan simbol drugog. [3] Ukoliko se u nekom stupcu nalaze identični nukleotidi, tada dodajemo $+5$ bod za sličnost, dok za različite dodajemo -3. Ukoliko se neki nukleotid našao iznad/ispod praznine (dakle dogodilo se izbacivanje/umetanje) u ovom ćemo trenutku dodavati -8 bodova po svakoj praznini, ali ćemo u nastavku rada poboljšati našu ocjenu na jedan realniji model. Slika 6 ilustrira jedan od mogućih načina poravnavanja nizova „TTACGTACAATTA" i „TGGAACAGTA" kojeg po ovom modelu ocjenjujemo sa -13 bodova.

T	T	A	C	G	T	A	C	A	–	A	T	T	A	
T	–	–	G	G	A	A	C	A	G	–	–	T	A	
+5	−8	−8	−3	+5	−3	+5	+5	+5	−8	−8	−8	+5	+5	= −13

Slika 6: Primjer poravnavanja

Ako izlistamo i ocijenimo sva moguća poravnavanja dvaju slijedova, najveći broj bodova koje smo uspjeli pronaći definiramo kao njihovu *sličnost*.[2] Nadalje, pod problemom poravnavanja dvaju sljedova podrazumijevamo nalaženje (maksimalne) sličnosti te neko poravnavanje kojim se postiže ta sličnost.

Ovako definirana usporedba dvaju DNA sekvenci se pokazala da mnogo bolje

[2]Korisno je primjetiti da tih poravnavanja ima konačno mnogo nakon što uvidimo kako nam se nikada neće isplatiti stavljati razmak iznad razmaka jer njihovim uklanjanjem vrlo jednostavno dolazimo većeg broja bodova. Kao rezultat, sličnost između dvaju slijedova je dobro definirana.

prati zakone prirode nego njezini prethodnici i njeno efikasno rješavanje se danas smatra jednim od najvažnijih problema bioinformatike.[10] Svoju primjenu je našla u mnogim granama znanosti, poput:

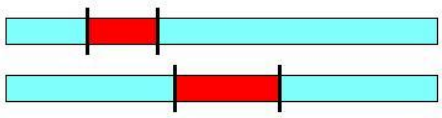
1. **Gradnja evolucijskog stabla:** sva današnja raznolikost vrsta je nastala mnogobrojnim genetskim mutacijama. Usporedbom sekvenci molekula DNA dviju vrsta možemo procijeniti prije koliko je vremena živio njihov zajednički predak, te odrediti evolucijske odnose među njima. [5]

2. **Procjena funkcije i strukture novootkrivenih proteina:** Sličnosti u nizu aminokiselina između novootkrivenog i poznatog proteina omogućuju znanstveniku da bolje procjeni funkciju i strukturu i prije nego što započne sa dugotrajnim postupkom njihova utvrđivanja. [5]

3. **Primjene izvan biologije:** Poravnavanje sljedova je našlo svoju primjenu i u poljima poput analize prirodnih jezika i u društvenim znanostima. Ekonomisti su također znali iskoristiti algoritme razvijene za ovaj problem kako bi analizirali niz transakcija na burzi. [6]

* * *

Pogledajmo jednu malu modifikaciju dosadašnjeg problema. Definirajmo *lokalno poravnavanje sekvenci* kao problem pronalaženja podnizova (uzastopnih znakova) obaju sljedova takvih da sličnost između njih bude maksimalna moguća (Slika 7). Dakako, zanima nas i pronalaženje nekog konkretnog poravnavanja tih podnizova koje će dovesti do maksimalnog broja bodova za sličnost.



Slika 7: Lokalno poravnavanje

2.1 Ocjena procjepa

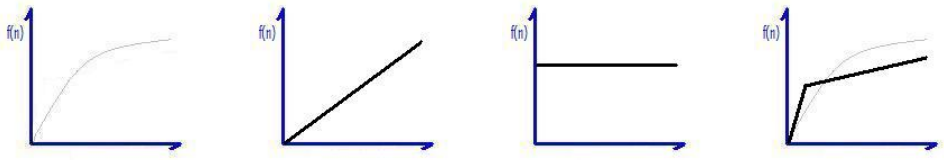
Prije nego što prezentiramo rješenje gore danog problema, potrebno je dodatno razraditi matematički model poravnavanja kako bi bolje opisivao empirijski dobivene podatke. U dosadašnjem smo modelu procjep (uzastopnih praznina) duljine n bodovali sa ocjenom $-8n$. Ovaj model nosi naziv linearan. U njemu će, na primjer, jednako bodova dobiti poravnavanje u kojem su napravljena dva procjepa duljine 2 i 3, kao i poravnavanje koje će napraviti samo jedan procjep duljine 5. Međutim, biološki je vjerojatnije da će se ostvariti ona mogućnost u kojoj je broj različitih procjepa minimiziran. Zbog toga se uvode novi modeli ocjene procjepa koji bolje aproksimiraju idealnu vjerjoatnost stvaranja procjepa veličine n. Za nas će najvažniji modeli biti upravo linearni, zbog jednostavnosti, te afini, jer najbolje aproksimira idealnu ocjenu od svih navedenih jednostavnih modela. Pogledajmo, zbog kompletnosti, sve popularnije modele:

1. **Idealna ocjena procjepa:** je empirijski dobivena konkavna funkcija koja je presložena da bi se koristila u bilo kakvoj praktičnoj primjeni pri usporedbi DNA sekvenci.

2. **Linearna ocjena procjepa:** Ako je k broj bodova koji dodajemo za otvaranje procjepa veličine 1, tada je ocjena procjepa veličine n upravo $n \cdot k$.

3. **Konstantna ocjena procjepa:** Bez obzira na veličinu, uvijek dodajemo isti broj bodova za bilo koji procjep.

4. **Afina ocjena procjepa:** uvode se dva parametra: d kao ocjena otvaranja novog procjepa i e kao ocjena svakog dodatnog proširenja procjepa nakon što je jednom otvoren. Ocjena procjepa duljine n je po ovom modelu $d + (n-1) \cdot e$.



Idealna ocjena Linearna ocjena Konstantna ocjena Afina ocjena

Slika 8: Osnovni modeli ocjene procjepa

3 Lokalno poravnavanje: Smith-Waterman algoritam

Problem lokalnog poravnavanja su prvi postavili i riješili T. F. Smith i M. S. Waterman u [7] još 1981. godine. Kako bismo lakše analizirali složenost, označimo duljinu sekvenci koje se uspoređuju sa N i M. Algoritam prezentiran u [7] je uvijek radio u $\Theta(NM)$ memorijskoj i vremenskoj složenosti za linearan model ocjene procjepa, dok mu je za afini model bilo potrebno povećanje vremena na $\Theta\big(NM(N+M)\big)$.

Za početak ćemo riješiti problem lokalnog poravnavanja za linearni model ocjene procjepa. Zbog elegancije zapisa, uvodimo substitucijsku funkciju

$$S : \{-, A, G, T, C\} \times \{-, A, G, T, C\} \longrightarrow \mathbb{Z}$$

koja utjelovljuje cijenu mutacije jednog nukletodia u drugi, a definirana je na sljedeći način:

Opis	**Primjer**
Ukoliko je x bilo koji nukleotid, tada $S(x,x)$ označava broj bodova koji dodajemo kada u poravnavanju međusobno pridružimo dva jednaka nukleotida. Vrijediti će da $S(x,x) > 0$.	$S(A,A) = 5$
Ukoliko su x i y različiti nukleotidi, oznakom $S(x,y)$ označavamo bodove za njihovo međusobno pridruživanje. Uvijek vrijedi da $S(x,y) < 0$.	$S(A,T) = -3$
Budući da dodavanja i proširivanja svih procjepa ocjenjujemo s jednakim brojem bodova, možemo tu ocjenu prikazati kao cijenu pretvorbe bilo kojeg nukleotida u prazninu. Dakle ukoliko je x proizvoljni nukleotid, sa $S(x,-) = S(-,x) < 0$ označavamo cijenu dodavanja praznine u bilo koju sekvencu.	$S(-,T) = -8$
Vrijednost $S(-,-)$ je nedefinirana jer nikada nećemo ni provjeravati poravnavanja u kojoj se ovo pridruživanje pojavljuje. Ovo opravdavamo činjenicom da poravnavanje sa ovim pridruživanjem nikada nije optimalno.	

Lokalno poravnavanje je originalno bilo rješeno metodom dinamičkog programiranja koju ovdje prezentiramo. Pretpostavimo da želimo naći sličnost sekvence A (duljine N) sa sekvencom B (duljine M). Postupno ćemo izgraditi matricu DP, gdje $DP(x,y)$ za $0 \leq x \leq N, 0 \leq y \leq M$ označava maksimalnu sličnost *nekog sufiksa* prvih x elemenata sekvence A i nekog sufiksa prvih y elemenata sekvence B. Jasno je da će izgradnja navedene matrice biti dovoljna da riješimo problem zato jer je $\max\{DP(x,y)\}$ upravo konačna lokalna sličnost koja nas zanima. Za početak pokažimo kako inicijalizirati matricu:

$$DP(0,y) = 0, \quad 0 \leq y \leq M$$
$$DP(x,0) = 0, \quad 0 \leq x \leq N$$

Inicijalizacija je opravdana jer najbolju lokalnu sličnost dobivamo ako poravnamo dvije prazne sekvence. Bilo koji drugi slučaj bi uključivao dodavanje praznina u inicijalno praznu sekvencu i, time, negativnu sličnost.

	$\emptyset$	A	C	G	T	C
$\emptyset$	0	0	0	0	0	0
G	0					
T	0					
A	0					
C	0					

Slika 9: Inicijalizacija matrice DP za sekvence „ACGTC" i „GTAC"

Pretpostavimo da izračunavamo matricu red po red (od gore prema dolje), te unutar retka od lijevo prema desno. Nadalje, recimo da je na red za izračunavanje došlo polje $DP(x, y)$. Bitno je primjetiti da su u tom trenutku polja $DP(x-1, y), DP(x, y-1)$ i $DP(x-1, y-1)$ sva izračunata. Možemo učiniti sljedeće:

a) Mutirati nukleotid A_x u nukleotid B_y (moguće je da su oni isti) sa cijenom $S(A_x, B_y)$. Potom preostaje da poravnamo prvih $x-1$ nukleotida sekvence A, te $y-1$ nukleotida sekvence B.

b) Provjerimo mogućnost da je znak A_x izbačen tijekom evolucije. Izbacivanje se boduje sa $S(A_x, -)$, te sad preostaje da poravnamo prvih $x-1$ nukleotida sekvence A, te y nukleotida sekvence B.

c) Preostaje da provjerimo je li znak B_y umetnut. Samo umetanje ima cijenu $S(-, B_y)$, te preostaje da poravnamo x nukleotida sekvence A, te $y-1$ nukleotida sekvence B.

d) Budući da tražimo lokalna poravnavanja, moguće je da odlučimo odbaciti sve prethodne znakove i započeti sa praznim nizovima. U tom slučaju poravnavanje se boduje sa 0 bodova.

Zbog navedenog vrijedi:

$$DP(x, y) = \max \begin{cases} S(A_x, B_y) + DP(x-1, y-1) \\ S(A_x, -) + DP(x-1, y) \\ S(-, B_y) + DP(x, y-1) \\ 0 \end{cases}$$

Navedeni algoritam omogućava izračunavanje cijele matrice DP u $\Theta(NM)$ vremena i memorije. Donja slika prikazuje stanje matrice DP nakon završetka algoritma. Ocjene u navedenoj slici odgovaraju modelu opisanom na stranici 4.

Ne smijemo zaboraviti kako mi moramo naći neko poravnavanje koje rezultira maksimalnom lokalnom sličnosti. Mi smo dosad opisali algoritam kojim dolazimo samo do njezina iznosa. Nasreću, nakon što smo izgradili matricu DP, *rekonstrukcija* samog poravnavanja je jednostavna. Pronađemo najveći elemente matrice DP i označimo ga sa $DP(x_m, y_m)$ i pitamo se kako pronaći poravnavanje koje rezultira sa vrijednosti $DP(x_m, y_m)$. Ukoliko je moguće tu vrijednost postići mutacijom elementa A_{x_m} u B_{x_y} (tj. ako $DP(x_m, y_m) = DP(x_m -$

$1, y_m - 1) + S(A_{x_m}, B_{y_m}))$, tada originalno pitanje svodimo na pitanje kako rekonstruirati vrijednost $DP(x_m - 1, y_m - 1)$. Slično danom razmatranju, ako $DP(x_m, y_m) = DP(x_m - 1, y_m) + S(A_{x_m}, -)$, pitanje svodimo na rekonstrukciju $DP(x_m - 1, y_m)$, te ukoliko $DP(x_m, y_m) = DP(x_m, y_m - 1) + S(-, B_{y_m})$, pitanje svodimo na rekonstrukciju $DP(x_m, y_m - 1)$. Opisanu rekurzivnu rekurzivnu rekonstrukciju izvršavamo sve dok ne dođemo do polja na kojem piše 0. Donja slika crvenom bojom prikazuje korake rekonstrukcijskog postupka.

	$\emptyset$	A	C	G	T	C
$\emptyset$	0	0	0	0	0	0
G	0	0	0	5	0	0
T	0	0	0	0	10	2
A	0	5	0	0	2	0
C	0	0	5	0	0	7

Slika 10: Konačno stanje matrice DP za sekvence „ACGTC" i „GTAC"

$* \ * \ *$

Dosadašnja su algoritamska razmatranja proučavala isključivo linearni model ocjene procjepa. Preostaje nam da vidimo kako bismo riješili teži problem afinog modela. Kako se sada praznine više ne boduju jednoliko, više ih ne možemo ocjenjivati substitucijskom funkcijom S, već uvodimo parametre d i e analogne onima prikazanim na stranici 6. Kao konkretne vrijednosti tih parametara koristimo $d = -10$, $e = -3$.

Osnovna se ideja pretjerano ne razlikuje od one prethodno prikazane. Jedina je razlika što će nam biti potrebne 3 matrice umjesto jedne. Definirajmo:

$DP_{ab}(x, y)$ Označava najveću sličnost bilo kojeg sufiksa zadnjih x, y znakova sekvenci A, B ukoliko se pridruživanje praznine bilo A_x bilo B_y boduje sa d bodova.

$DP_a(x, y)$ Označava najveću sličnost bilo kojeg sufiksa zadnjih x, y znakova sekvenci A, B ukoliko se pridruživanje praznine nukleotidu A_x boduje sa e bodova, dok se nukleotidu B_y boduje sa d bodova.

$DP_b(x, y)$ Označava najveću sličnost bilo kojeg sufiksa zadnjih x, y znakova sekvenci A, B ukoliko se pridruživanje praznine nukleotidu A_x boduje sa d bodova, dok se nukleotidu B_y boduje sa e bodova.

Svaku od dane tri matrice inicijaliziramo identično kako smo inicijalizirali matricu DP u linearnom modelu. Izračun samih vrijednosti matrice obavlja se po sljedećoj formuli, uz objašnjenja analogna (i preduga da bi ovdje detaljno bila opisana) linearnom modelu:

$$
\begin{aligned}
DP_{ab}(x,y) &= \max \begin{cases} S(A_x, B_y) + DP_{ab}(x-1, y-1) \\ d + DP_a(x-1, y) \\ d + DP_b(x, y-1) \\ 0 \end{cases} \\
DP_a(x,y) &= \max \begin{cases} S(A_x, B_y) + DP_{ab}(x-1, y-1) \\ e + DP_a(x-1, y) \\ d + DP_b(x, y-1) \\ 0 \end{cases} \\
DP_b(x,y) &= \max \begin{cases} S(A_x, B_y) + DP_{ab}(x-1, y-1) \\ d + DP_a(x-1, y) \\ e + DP_b(x, y-1) \\ 0 \end{cases}
\end{aligned}
$$

Konačna lokalna sličnost je upravo najveći element matrice DP_{ab}, a poravnavanje se može rekonstruirati direktnim analogom sa linearnim modelom. Preostaje još napomenuti kako i ovaj algoritam ima jednake karakteristike kao i onaj koji rješava linearni model zato jer koristi $\Theta(MN)$ memorije i vremena.

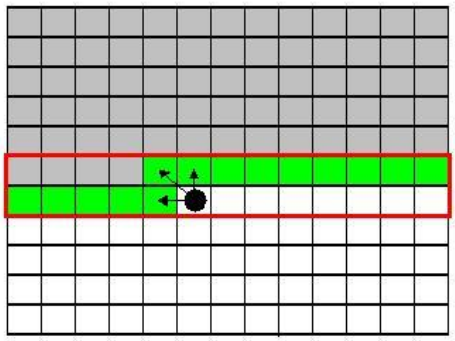
4 Memorijska optimizacija Smith-Waterman algoritma

Smith-Watermanov algoritam opisan u prethodnoj odjeljku predstavlja osnovni alat pri usporedbi DNA sekvenci. Primjera radi, pogledat ćemo grubu procjenu vremena i memorije koje algoritam zahtjeva pri izvršavanju za dvije ulazne sekvence jednakih duljina ($N = M$). [3]

N	Vrijeme	Memorija
100	1 ms	0.1 MB
1000	0.1 s	12 MB
100000	1000 s	120 GB
1000000	100000 s	12 TB

Posljednji redak tablice ugrubo odgovara usporedbi dvaju ljudskih kromosoma. Bitno je uočitii kako su na velikim ulazima potrebni memorijski resursi preveliki za praktične primjene. Pomalo iznenađujuće, ali memorijsku složenost je moguće drastično smanjiti uz samo konstantni gubitak na brzini izvršavanja. Naime, Hirschberg je 1975. godine u [8] opisao algoritam za rješavanje problema vrlo bliskog[4] usporedbi dvaju DNA sekvenci.

Zbog jednostavnosti ograničit ćemo se na linearni model, ali čitatelj bi trebao imati na umu da se prikazani algoritmi mogu jednostavno primjeniti i na afinom modelu. Za početak primjetimo: kada bi nas zanimala samo maksimalna sličnost (a ne i neko konkretno poravnavanje), tada bismo mogli jednostavno modificirati originalni algoritam tako da radi u linearnoj memorijskoj složenosti.



Slika 11: Matrica DP tijekom izvođenja Smith-Watermanovog algoritma

Slika 11 ilustrira ideju iza algoritma. Pretpostavljamo da smo izračunali sva polja koja su obojana u sivu ili zelenu boju, te još trebamo izračunati sva bijela polja. Sa slike je vidljivo da je za izračunavanje vrijednosti matrice DP

[3] Pretpostavljamo da je konstanta algoritma 100, da sekvencijalno računalo izvršava 10^9 operacija u sekundi, te da za spremanje matrica koristimo 32-bitni tip podataka.

[4] Hirschberg je u [8] opisao pronalaženje najduljeg zajedničkog (ne nužno uzastopnog) podniza dvaju ulaznih nizova.

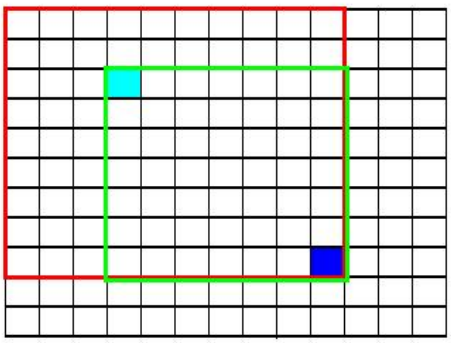
za bijela polja dovoljno pamtiti *isključivo* polja zelene boje (kojih ima $O(M)$). No, zbog jednostavnosti implementacije, obično ćemo pamtiti cijela zadnja dva retka podataka (polja uokvirena crvenim pravokutnikom).

Nadalje, pretpostavimo da računamo sličnost između neke dvije sekvence A i B, duljina N i M. Označimo sa A^R i B^R okrenute sekvence A i B. Jasno je, zbog prirode definicije lokalne sličnosti, da je sličnost između sekvenci A i B jednaka sličnosti između sekvenci A^R i B^R. Drugim riječima, mogli bismo originalni problem riješiti i popunjavajući matricu od kraja sekvenci krečući se prema početku. Da bismo formalizirali ovaj postupak, definirajmo matricu DP^R:

$$
\begin{aligned}
DP^R(N, y) &= 0, \quad 0 \leq y \leq M \\
DP^R(x, M) &= 0, \quad 0 \leq x \leq N \\
DP^R(x, y) &= \max \begin{cases} 0 \\ DP^R(x+1, y+1) + S(A_{x+1}, B_{y+1}) \\ DP^R(x+1, y) + S(A_{x+1}, -) \\ DP^R(x, y+1) + S(-, B_{y+1}) \end{cases}
\end{aligned}
$$

Kako se sva objašnjenja „normalnog" Smith-Waterman algoritma mogu preslikati i na ovu varijantu, jasno je da se i ova matrica može upotrebiti za nalaženje najboljeg lokalnog poravnavanja.

Zbog navedene observacije, moguće je problem rekonstrukcije lokalne sličnosti svesti na problem rekonstrukcije globalne sličnosti: usporedimo sekvence A i B normalnim Smith-Waterman algoritmom i označimo maksimalni element matrice DP sa koordinatama (x_k, y_k). Te koordinate nam označavaju zadnje iskorištene znakove u najboljem lokalnom poravnavanju. Potom pokrenemo „invertirani" Smith-Waterman algoritam na prvih x_k odnosno y_k znakova sekvenci A i B kako bismo našli prvi iskorišten znak u nekom najboljem lokalnom poravnavanju. Potom preostaje da pokrenemo (opisan u daljnjem tekstu) algoritam globalnog poravnavanja na sekvencama $(A_{x_p}, A_{x_p+1}, ..., A_{x_k})$ te $(B_{y_p}, B_{y_p+1}, ..., B_{y_k})$.



Slika 12: Svođenje lokalne sličnosti na globalnu sličnost

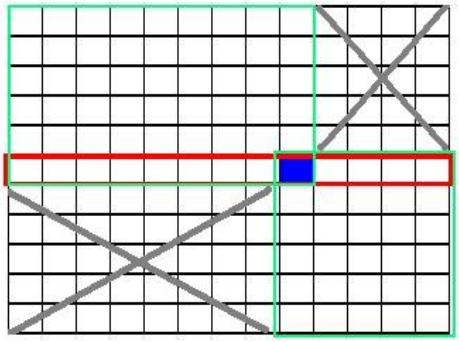
Slika 12 ilustrira navedeni algoritam: prvo se normalnim algoritmom nalazi

tamnoplavo polje, potom se u poljima označenim crvenim okvirom pokreće „invertirani" Smith-Waterman koji pronalazi svijetloplavo polje, te se potom u poljima označenim zelenim okvirom pronalazi najbolje globalo poravnavanje. Korisno je primjetiti da svodjenje sa lokalnog na globalni problem zahtjeva $O(NM)$ vremena i $O(M)$ memorije ako koristimo prethodno opisanu observaciju.

Još je potrebno prezentirati rješenje problema globalnog poravnavanja koji koristi linearno memorije kako bismo ostvarili zacrtani cilj. Uz standardne oznake za sekvence i njihove duljine, označimo sa $p = \lfloor N/2 \rfloor$ polovicu duljine sekvence A, zaokruženu na dolje. Pomoću normalnog i invertiranog Smith-Waterman algoritma izračunajmo $p.$ redak matrica DP i DP^R. Zbrajajući vrijednosti koje se nalaze u istom stupcu definiramo niz

$$\Psi(y) = DP(p, y) + DP^R(p, y),\ 0 \leq y \leq M$$

te ćemo poziciju njegovog najvećeg elementa niza Ψ označiti sa y_p. Jasno je da postoji optimalno (globalno) poravnavanje koje tijekom rekonstrukcije prolazi kroz polje (p, y_p) matrice DP odnosno DP^R. Mi ćemo ga rekurzivno pronaći tako da pozovemo isti upravo opisan algoritam za prvih p odnosno y_p znakova sekvenci, te ga pozovemo za sve znakove od $p.$ odnosno y_p. Slika 13 ilustrira ovu ideju: crveno okvir označava $p.$ redak, dok crveno polje označava poziciju (p, y_p). Potom se isti algoritmi pozovu za oba svijetlo plava okvira. Polja matrice koja se nalaze u prekriženom dijelu slike se više nikada ne gledaju.



Slika 13: Osnovni korak Smith-Waterman algoritma koji koristi linearno memorije

Gore opisan algoritam rješava problem globalnog poravnavanja dvaju sekvenci. Da bismo riješili originalni problem lokalnog poravnavanja, prvo je potrebno pretvoriti ga u problem globalnog poravnavanja, za što je potrebno $O(MN)$ vremena i $O(M)$ memorije, te se potom upotrijebi gore opisan algoritam kojeg preostaje još analizirati za složenosti.

Bitno je primjetiti da će algoritam koji rješava globalno poravnavanje u svakom koraku prekrižiti približno (do na zanemariv ostatak) polovicu polja originalne $N \cdot M$ matrice koje više nikad neće biti promatrana. Zbog toga je vremenska složenost globalnog poravnavanja

$$MN + MN/2 + MN/4 + MN/8 + ... < 2MN \in O(MN).$$

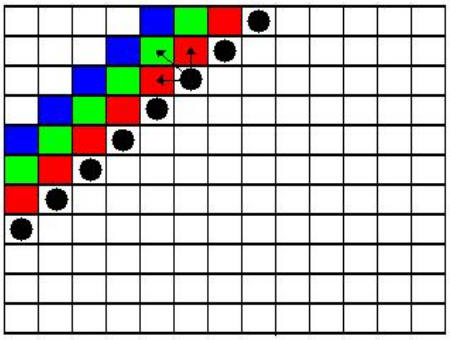
Sa druge strane, svaki korak algoritma za globalno poravnavanje zahtjeva linearno memorije, a budući da se cijeli program vrti na jednoprocesnom računalu, tu je memoriju moguće reciklirati pa zaključujemo da mu je memorijska složenost također $O(M)$.

Gornji rezultat dakako povlači da je i problem lokalne sličnosti moguće riješiti u memorijskoj složenosti $\max(O(M), O(M)) = O(M)$ te vremenskoj složenosti $O(MN) + O(MN) = O(MN)$, kao što je bilo i najavljivano.

5 Daljnji rad

U današnjem vremenu kada su višeprocesorska računala i višejezgrene grafičke kartice donijele osobnim računalima mogućnosti izvršavanja preko 10^12 paralelnih operacija po sekundi, prirodno je očekivati da će se sve više vremenski zahtjevnih aplikacija okretati paralelizaciji svojih procedura kako bi svoje vrijeme izvršavanja dovele u praktične proporcije. [9] Naravno, niti Smith-Watermanov algoritam nije iznimka. Kao jedan od osnovnih algoritama bioinformatike sa velikom vremenskom složenosti, efikasnom paralelizacijom bi se došlo do velike (vremenske i novčane) uštede.

Iako je tema paralelizacije Smith-Waterman algoritma vrlo široka i pomalo neistražena, ovdje ćemo istaknuti jednu od ideja koje vode prema efikasnoj implementaciji algoritma na višejezgrenoj grafičkoj kartici. Primjetimo kako smo u prijašnjim odijeljenjima računali dinamičke matrice (DP) red po red. Takav pristup nije primjeren za paralelno programiranje zato jer se neki element matrice neće moći izračunati prije nego što se element neposredno lijevo od njega ne izračuna. Nasreću, takav nedostatak je moguće izbjeći ako računamo elemente paralelno po svakoj sporednoj diagonali, kao što to ilustrira slika 14. [10]



Slika 14: Paralelizacija Smith-Waterman algoritma

Sva polja označena istom bojom bi se, idealno, trebala izračunavati istovremeno.

Zanimljivo je kako nigdje u autor nigdje u literaturi nije uspio pronaći opis paralelnog algoritma koji koristi linearnu memorijsku složenost prilikom izračunavanja maksimalne lokalne sličnosti. Unatoč tome, autor vjeruje kako je dotični algoritam moguće ostvariti proširenjem Hirschbergovog algoritma iskazanog u [8] i gornje ideje koja je opisana u [10].

6 Zaključak

Problem lokalnog poravnavanja sekvenci je jedan od najvažnijih problema u bioinformatici. [10] Njegovo rješavanje ima dalekosežne posljedice kako u bioinformatici, tako i u raznim nebiološkim granama poput proučavanja prirodnih jezika ili analize ekonomskih transakcija. [6]

Ukoliko želimo naći optimalno lokalno poravnavanje ulaznih sekvenci, najpopularniji osmišljeni algoritam su 1981. godine razvili Smith i Waterman u [7]. Iako koristi metodu dinamičkog programiranja, algoritam radi u kvadratnoj memorijskoj i vremenskoj složenosti s obzirom na veličinu ulaznih podataka. Takva složenost je vrlo često prevelika da bi bila korisna u praktičnim primjenama, stoga na raspolaganju stoji nekoliko alternativa:

1. Iskoristiti neke od ne-optimalnih heuristika za rješavanje istog problema koji su superiorniji u memorijskoj i vremenskoj složenosti od optimalnih algoritama. Takve metode se nisu obrađivale u ovom radu, ali se mogu pronaći u [5].

2. Nadograditi originalni Smith-Watersonov algoritam pomoću Hirschbergove ideje opisane u [8] te dobiti algoritam koji ima kvadratnu vremensku i linearnu memorijsku složenost.

3. Paralelizirati algoritam na višeprocesorskom računalu ili na višejezgrenoj grafičkoj kartici. Reference i primjeri ovog pristupa su dani na strani 15.

Utjecaj razvoja informatike na biologiju samo je jedan od primjera kako je razvoj jedne grane znanosti utjecalo na proboj u drugima. Taj fenomen je posebno istaknut u računarskoj znanosti koja je u posljednjih nekoliko desetljeća potaknula razvoj i u većini drugih znanstvenih disciplina, na primjer:

- Zdravstvena informatika - disciplina u kojoj se informacijska tehnologija primjenjuje pri optimizaciji i analizi simptoma te pri pomoći u odabiru lijekova. Također uključuje izradu informacijskog sustava medicinskih klinika.

- Arheologija - računalni programi danas pomažu pri dešifriranju nekih starih pisma.

- Meteorologija - predviđanja vremenskih prognoza su trebala čekati nekoliko desetljeća do pojave prvih računala kako bi njihova teoretska primjena zaživjela.

- Ekonomija - postoje algoritmi koji se bave analizom kretanja tržišta te mikro- i makro- ekonomskim predviđanjima. [5]

Literatura

[1] *Nucleotide*, wikipedia, en.wikipedia.org/wiki/Nucleotide (dohvaćeno 10. travnja 2011.)

[2] *Marian Koshland Science Museum*, www.koshland-science-museum.org/exhibitdna/intro03.jsp (dohvaćeno 10. travnja 2011.)

[3] S. Ristov. *Usporedba sljedova*, www.irb.hr/hr/home/ristov/predavanja/Poravnanja07.ppt (dohvaćeno 10. travnja 2011.)

[4] *Gene duplication*, wikipedia, en.wikipedia.org/wiki/Gene_duplication (dohvaćeno 15. travnja 2011.)

[5] Čanadi, Igor (2009.) *Algoritmi za poravnavanje dviju sekvenci*, Seminar, complex.zesoi.fer.hr/Seminars.html

[6] *Sequence alignment*, wikipedia, en.wikipedia.org/wiki/Sequence_alignment (dohvaćeno 10. travnja 2011.)

[7] Smith, Temple F.; and Waterman, Michael S. (1981.) *Identification of Common Molecular Subsequences.* Journal of Molecular Biology 147: 195–197. doi:10.1016/0022-2836(81)90087-5

[8] D. S. Hirschberg (1975.) *A linear space algorithm for computing maximal common subsequences.* Comm. A.C.M. 18(6) p341-343

[9] D. B. Kirk, W. W. Hwu (2010.) *Programming Massively Parallel Processors: A Hands-on Approach*, 1^{st} edition, ISBN:0123814723 9780123814722

[10] E. F. O. Sandes, A. C. M. A. Melo (2010.) *CUDAlign: using GPU to accelerate the comparison of megabase genomic sequences*, doi.acm.org/10.1145/1693453.1693473

Sažetak

Bioinformatika je disciplina koja primjenjuje informacijsku tehnologiju na području molekularne biologije. [5] Jedno od najvažnijih područja bioinformatike je poravnavanje sekvenci, metoda usporedbe sličnosti dvije DNA sekvence. [10]

Rješenje problema poravnavanja sekvenci koristi se u mnogim znanstvenim disciplinama, poput gradnje evolucijskog stabla, usporedbe vrsta, identifikacije proteina i ostalim biološkim pa čak i nebiološkim primjenama. [5]

U ovom seminarskom radu prikazana je motivacija za definiciju problema lokalnog poravnavanja sekvenci te je prikazano njeno klasično rješenje: Smith-Watermanov algoritam koji ga rješava u $\Theta(MN)$ vremenskoj i memorijskoj složenosti. Potom je taj algoritam nadograđen tako da koristi samo linearnu memorijsku složenost bez značajne degradacije na vremenu.

U svjetlu povećanja računalne moći višejezgrenih grafičkih kartica prikazan je jedan od mogućih pristupa paralelizaciji samog algoritma, dane su pripadajuće reference, te je opisana smjernica za daljnje istraživanje teme.