SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2321

SWIG - Poravnanje struktura korištenjem iterativne primjene Smith-Waterman algoritma

Bruno Rahle

Zagreb, svibanj 2012.

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

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

1. Uvod

Problemi s kojima se znanost danas susreće često prelaze granice isključivo jedne discipline.

- pričaj o: bioinformatici, koje probleme ona rješava, zašto je problem koji si rješavao bitan, kome koristi, kako se može koristiti i potencijalno za što se sve može koristiti. - napiši zašto je ovo što si napravio jebeno i kako se može još poboljšati. - cca 2 stranice

2. Poravanavanje sturktura

- detaljan opis problema - cca 1-2 stranice

3. Smith-Watermanov algoritam

Smith-Watermanov algoritam služi nam da bismo pronašli lokalno poravnanje. Osmislili su ga Temple F. Smith i Michael S. Waterman 1981. Smith (1981). Temelji se na Needleman-Wunschevom algoritmu (Needleman (1970)) te i sam spada u kategoriju algoritama dinamičkog programiranja. Glavna razlika između ta dva algoritma jest što Needleman-Wunschov algoritam prolazi globalno poravananje.

- napisi jos teksta ovdje.

3.1. Needleman-Wunschov algoritam

Algoritam, kao što je već rečeno, traži globalno poravnanje. To znači da se svi članovi ulaznih nizova moraju poravnati. Dopuštene operacije kada tražimo poravnanje su preklapanje s elementom iz suprotnog niza i ubacivanje praznina u neki od nizova. Sve parove elemenata u dobivenom preklapanju bodujemo i na osnovu te ocjene određujemo sličnost nizova. Konačan rezultat ovog algoritma jest poravnanje koje maksimizira takvu ocjenu, tj. daje maksimalno globalno poravnanje.

Zanimljivost je da je to prvi algoritam dinamičkog programiranja ikada primjenjen u bioinformatici.

3.1.1. Ulazni podaci

1. Dva niza (A i B) proteina ili nukleotida. Zbog jednostavnosti, pretpostavit ćemo da su to nukleotidi iz DNK - adenin (A), timin (T), gvanin (G) i citozin (C). U primjeru ćemo koristiti $A = "ATGCCGTA"$ i $B = "TGCACTA"$. Dužinu niza A označit ćemo s N, a dužinu niza B s M.

2. Supstitucijska matrica S, koja nam daje bodove koje dobijemo kada jedan nukletoid preklopimo s drugim. U našem će slučaju imati dimenzije 4×4, budući da ćemo razmatrati slučaj kada imamo samo četiri nukleotida. U principu će na dijagonali imati pozitivne brojeve, a na ostalim poljima negativne. To znači da

nam se najviše isplati preklapati nukletide istog tipa, jer za to dobivamo bodove, a inače ih gubimo. Primjer jedne takve matrice koju ćemo koristiti i u primjeru:

	A	C	G	T
A	10	-3	-9	-1
C	-5	8	-8	-7
G	-5	-4	7	-5
T	-4	-11	-8	9

3. Negativan broj d, koji označava bodove koje dobijemo (tj. izgubimo) kada nukleotid preklopimo s prazninom. U primjeru ćemo koristiti $d = -5$.

3.1.2. Izlazni podaci

1. Broj H, ocjena najboljeg globalnog poravnanja.

2. Dva nova niza jednake dužine, A' i B', nastala ubacivanjem praznina (označenih najčešće sa '-') u nizove A i B koja predstavljaju najbolje pronađeno poravnanje.

3.1.3. Algoritam

Neka nam matrica F služi za računanje poravnanja. Tada će nam $F_{i,j}$ označavati maksimalan broj bodova koje možemo dobiti kada poravnamo prvih i članova niza A i prvih j članova niza B. $F_{i,j}$ možemo računati rekurzijom na slijedeći način:

$$F_{i,j} = \begin{cases} 0 & \text{ako je } i = 0 \text{ i } j = 0 \\ F_{i-1,j} + d & \text{ako je } i > 0 \text{ i } j = 0 \\ F_{i,j-1} + d & \text{ako je } i = 0 \text{ i } j > 0 \\ max \begin{pmatrix} F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ F_{i-1,j} + d \\ F_{i,j-1} + d \end{pmatrix} & \text{ako je } i > 0 \text{ i } j > 0 \end{cases}$$

U $F_{N,M}$ će nam stoga pisati maksimalno globalno poravnanje. Primjetite da u niti jednom slučaju nećemo poravnati dvije praznine. Ako bismo to učinili, samo bismo izgubili bodove, budući da je d nužno negativan broj.

Da bismo znali rekonstruirati rješenje, koristit ćemo matricu R. U polju $R_{i,j}$ pisat će koje smo polje matrice F koristili da bi došli u polje $F_{i,j}$. Kako su jedine mogućnosti $F_{i-1,j}$, $F_{i,j-1}$, $F_{i-1,j-1}$ i da nismo došli iz nikojeg polja (to vrijedi jedino za polje $F_{0,0}$), koristit ćemo redom oznake A, B, O i X.

$$R_{i,j} = \begin{cases} X & \text{ako je } \begin{pmatrix} i = 0 \\ j = 0 \end{pmatrix} \\ O & \text{ako je } \begin{pmatrix} i > 0 \\ j > 0 \\ F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \geq F_{i-1,j} + d \\ F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \geq F_{i,j-1} + d \end{pmatrix} \\ A & \text{ako je } \begin{pmatrix} i > 0 \\ j = 0 \end{pmatrix} \text{ili } \begin{pmatrix} i > 0 \\ j > 0 \\ F_{i-1,j} + d > F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ F_{i-1,j} + d \geq F_{i,j-1} + d \end{pmatrix} \\ B & \text{ako je } \begin{pmatrix} i = 0 \\ j > 0 \end{pmatrix} \text{ili } \begin{pmatrix} i > 0 \\ j > 0 \\ F_{i,j-1} + d > F_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ F_{i,j-1} + d > F_{i-1,j} + d \end{pmatrix} \end{cases}$$

Rekonstrukciju provodimo tako krenemo iz polja $R_{N,M}$ i krećemo se po matrici unazad dok ne dođemo do polja na kojem piše X, tj. $R_{0,0}$. Ako na polju pročitamo O, pomičemo se po dijagonili, tj. u izlazni niz spremimo par (A_{i-1}, B_{j-1}) te smanjimo i i i j. Ako pročitamo A, spremamo par $(A_{i-1}, -)$ te smanjimo samo i za jedan. U slučaju da pročitamo B, spremamo par $(-, B_{j-1})$ te smanjujemo j za jedan. Ako smo pročitali X, došli smo do kraja i generirali smo izlazni niz, ali u obrnutom redosljedu.

3.1.4. Primjer

Za prethodno navedene ulazne podatke, matrice F i R izgledat će ovako:

$F =$

	-	T	G	C	A	C	T	A
-	0	-5	-10	-15	-20	-25	-30	-35
A	-5	-1	-6	-11	-5	-10	-15	-20
T	-10	4	-1	-6	-10	-15	-1	-6
G	-15	-1	11	6	1	-4	-6	-6
C	-20	-6	6	19	14	9	4	-1
C	-25	-11	1	14	14	22	17	12
G	-30	-16	-4	9	9	17	17	12
T	-35	-21	-9	4	5	12	26	21
A	-40	-26	-14	-1	14	9	21	36

$R =$

	-	T	G	C	A	C	T	A
-	X	B	B	B	B	B	B	B
A	A	O	B	B	O	B	B	O
T	A	O	B	B	A	A	O	B
G	A	A	O	B	B	B	A	O
C	A	A	A	O	B	O	B	B
C	A	A	A	O	O	O	B	B
G	A	A	O	A	O	A	O	O
T	A	O	A	A	O	A	O	B
A	A	A	A	A	O	B	A	O

Iz tih podataka lagano je napraviti rekonstrukciju (polja označena sivom bojom). Stoga zaključujemo da je traženo globlano poravnanje $A' =$ "ATGCCGTA" i $B' =$ "-TGCACTA".

3.1.5. Analiza složenosti

Trivijalno je vidljivo da su memorijska i vremenska složenost opisanog algoritma jednake $O(NM)$.

3.2. Smith-Watermanov algoritam

Razlika njega i prethodno opisanog Needleman-Wunschevog algoritma jest u tome što ovaj algoritam traži najbolje lokalno poravnanje. To znači da ne koristi nužno cijele nizove proteina ili nukleotida već samo najsličnije uzastopne podnizove. U praksi se koriste nešto poboljšane verzije ovog algoritma.

3.2.1. Ulazni podaci

Vidi odlomak 3.1.1.

3.2.2. Izlazni podaci

Vidi odlomak 3.1.2.

3.2.3. Algoritam

U ovom ćemo algoritmu koristiti matricu H za računanje poravnanja. Za razliku od Needleman-Wunschevog algoritma, $H_{i,j}$ ovaj će put označavati rezultat najboljeg lokalnog poravnanja koje koristi A_{i-1} i B_{j-1}. Matricu ćemo popuniti na slijedeći način:

$$H_{i,j} = \begin{cases} 0 & \text{ako je } i = 0 \text{ ili } j = 0 \\ max \begin{pmatrix} 0 \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H_{i-1,j} + d \\ H_{i,j-1} + d \end{pmatrix} & \text{ako je } i > 0 \text{ i } j > 0 \end{cases}$$

Primjetite bitnu razliku u odnosu na Needleman-Wunschev algoritam - u ovom slučaju matrica H neće sadržavati negativne brojeve. Rješenje, tj. najbolje lokalno poravnanje više neće pisati na polju $H_{N,M}$. Ono će biti na polju (t, u) na kojem se nalazi najveći broj u matrici.

Da bismo znali rekonstruirati takvo lokalno poravnanje, koristit ćemo matricu R koju gradimo na sličan način kao i kod prethodnog algoritma:

$$R_{i,j} = \begin{cases} X & \text{ako je} H_{i,j} = 0 \\ O & \text{ako je} \begin{pmatrix} i > 0 \\ j > 0 \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \geq H_{i-1,j} + d \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \geq H_{i,j-1} + d \end{pmatrix} \\ A & \text{ako je} \begin{pmatrix} i > 0 \\ j = 0 \end{pmatrix} \text{ili} \begin{pmatrix} i > 0 \\ j > 0 \\ H_{i-1,j} + d > H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H_{i-1,j} + d \geq H_{i,j-1} + d \end{pmatrix} \\ B & \text{ako je} \begin{pmatrix} i = 0 \\ j > 0 \end{pmatrix} \text{ili} \begin{pmatrix} i > 0 \\ j > 0 \\ H_{i,j-1} + d > H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H_{i,j-1} + d > H_{i-1,j} + d \end{pmatrix} \end{cases}$$

Rekonstrukcija se, slično kao i kod Needleman-Wunschevog algoritma, izvodi tako da krenemo iz polja (t, u) i krećemo se po matrici R dok ne dođemo do polja na kojem piše X koje ovaj put ne mora nužno biti polje $R_{0,0}$. Dobiveni niz okrenemo i dobit ćemo traženo poravnanje.

3.2.4. Primjer

Za prethodno navedene ulazne podatke, matrice H i R izgledat će ovako:

$H =$

	-	T	G	C	A	C	T	A
-	0	0	0	0	0	0	0	0
A	0	0	0	0	10	5	0	10
T	0	9	4	0	5	0	14	9
G	0	4	16	11	6	1	9	9
C	0	0	11	24	19	14	9	4
C	0	0	6	19	19	27	22	17
G	0	0	7	14	14	22	22	17
T	0	9	4	9	10	17	31	26
A	0	4	0	4	19	14	26	41

$R =$

	-	T	G	C	A	C	T	A
-	X	X	X	X	X	X	X	X
A	X	X	X	X	O	B	B	O
T	X	O	B	X	A	A	O	B
G	X	A	O	B	B	O	A	O
C	X	X	A	O	B	O	B	O
C	X	X	A	O	O	O	B	B
G	X	X	O	A	O	A	O	O
T	X	O	B	A	O	A	O	B
A	X	A	O	A	O	B	A	O

Prema tome, traženo poravnanje jest $A' =$ "TGC-CGTA" i $B' =$ "TGCAC-TA".

3.2.5. Analiza složenosti

Trivijalno je vidljivo da su memorijska i vremenska složenost i ovog algoritma jednake $O(NM)$.

3.3. Procjepi

Do sada smo razmatrali samo slučaj kada je cijena otvaranja procjepa i njegova proširivanja jednaka (d). Taj slučaj nazivamo *linearnom ocjenom procjepa*, budući da, ako je

k dužina procjepa, dk je cijena za taj procjep. U praksi se primjenjuju još dva načina ocjenjivana procjepa.

Jedan je da za svaki procjep, bez obzira na njegovu dužinu, platimo fiksnu cijenu (c), a nazivamo ga *konstantnom ocjenom procjepa*.

Drugi je kombinacija prethodna dva načina: za svaki procjep plaćamo fiksnu cijenu (c), ali za njegovo produženje plaćamo neku drugu cijenu (d). Takvu funkciju ocjene procjepa nazivamo *Afinom ocjenom procjepa*. Prema tome, za procjep dužine k, platit ćem cijenu jednaku $c + (k-1)d$. Kako je empirijski pokazano da je ta funkcija u biti parobala, ovakva je ocjena ujedno i najpreciznija. Nažalost, ona otežava računanje matrice H (i R). Problem je najlakše riješiti tako da uvedemo dvije nove matrice H^A i H^B koje će nam pomoći u računanju cijene procjepa. Matrice ćemo onda računati na slijedeći način:

$$H_{i,j} = \begin{cases} 0 & \text{ako je } i = 0 \text{ ili } j = 0 \\ max \begin{pmatrix} 0 \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H^A_{i-1,j} + c \\ H^B_{i,j-1} + c \end{pmatrix} & \text{ako je } i > 0 \text{ i } j > 0 \end{cases}$$

$$H^A_{i,j} = \begin{cases} 0 & \text{ako je } i = 0 \text{ ili } j = 0 \\ max \begin{pmatrix} 0 \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H^A_{i-1,j} + d \\ H^B_{i,j-1} + c \end{pmatrix} & \text{ako je } i > 0 \text{ i } j > 0 \end{cases}$$

$$H^B_{i,j} = \begin{cases} 0 & \text{ako je } i = 0 \text{ ili } j = 0 \\ max \begin{pmatrix} 0 \\ H_{i-1,j-1} + S_{A_{i-1},B_{j-1}} \\ H^A_{i-1,j} + c \\ H^B_{i,j-1} + d \end{pmatrix} & \text{ako je } i > 0 \text{ i } j > 0 \end{cases}$$

Rekonstrukcijska matrica vrlo je slična onoj u izvorno opisanom algoritmu, a kako ćemo u uđućem potpoglavlju maknuti potrebu za njom, nećemo je posebno navoditi.

3.4. Memorijska optimizacija

3.5. Substitucija substitucijske matrice

- TODO: neko bolje ime za ovo potopoglavlje U ovom ćemo radu umjesto supstitucijske matrice za usporedbu dva elementa koristiti fizičku udaljenost između dva atoma. Stoga će nizovi A i B biti zapravo nizovi koordinata iz kojih ćemo moći za svaki par elemenata izračunati koordinate.

4. Algoritam simuliranog kaljenja

- objasni i zašto si uzeo više nizova odjednom, kako od njih biraš najboljeg i slično. - cca 4 starnice

5. CUDA tehnologija

- cca 2 stranice
    - prednosti i mane, problemi i rjesenja

6. Implementacija

- cca 5 stranica
    - kako je sve spojeno
    - detalji implementacije - koje funkcije ocjene sam koristio, kako sam došao do njih

7. Rezultati

- cca 1-2 starnice

8. Zaključak

Zaključak. - cca 1-2 stranice

LITERATURA

Christian D. Needleman, Saul B.; Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins, 1970.

Michael S. Smith, Temple F.; Waterman. Identification of common molecular subsequences, 1981.

SWIG - Poravnanje struktura korištenjem iterativne primjene Smith-Waterman algoritma

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.