SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 293

POOPĆENJE ALGORITMA ZA PORAVNANJE PARCIJALNOG UREĐAJA

Marin Rašić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 293

POOPĆENJE ALGORITMA ZA PORAVNANJE PARCIJALNOG UREĐAJA

Marin Rašić

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 293

Pristupnik:	**Marin Rašić (0036514528)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Mile Šikić

Zadatak: **Poopćenje algoritma za poravnanje parcijalnog uređaja**

Opis zadatka:

Višestruko poravnanje sekvenci jedan je od važnijih problema u bioinformatici. Od nekolicine progresivnih metoda, poravnanje parcijalnog uređaja spada u skupinu najpopularnijih te poopćuje poravnanje dviju sekvenci na poravnanje sekvence i grafa parcijalnog uređaja. Da bi pronašli višestruko poravnanje skupa sekvenci potrebno je iterativno poravnati i uključiti svaku sekvencu u graf, što onemogućuje paralelizaciju na razini skupa sekvenci. Jedno od mogućih rješenja je poopćiti poravnanje sekvence i grafa parcijalnog uređaja na poravnanje dva grafa te potom poravnavati sekvence i grafove u parovima istovremeno. Potrebno je implementirati alat koji poravnava i sekvence i grafove parcijalnog uređaja s linearnim procjepima te usporediti rješenje s alatom Spoa. Rješenje mora biti napisano koristeći programski jezik C++. Programski kod je potrebno komentirati i pri pisanju pratiti neki od standardnih stilova. Napisati iscrpne upute za instalaciju i izvođenje. Kompletno programsko rješenje postaviti na GitHub pod jednom od OSI-odobrenih licenci.

Rok za predaju rada: 11. lipnja 2021.

SADRŽAJ

1. Uvod

Poravnanje nizova proces je kojim se uspoređuju dva niza i traže sličnosti među njima. Ovisno o cilju, različiti algoritmi poravnanja pronalaze različite sličnosti među nizovima. Poravnanje bioloških slijedova jedan je od glavnih izazova područja bioinformatike. Postupak poravnanja provodi se zbog brojnih razloga: pronalaska evolucijske povezanosti među vrstama, analize genetski uzrokovanih bolesti, kreiranje obiteljskih stabala, razvoja novih lijekova i mnogih drugih.

Razlikujemo dvije vrste poravnanja: poravnanje parova slijedova (engl. *pairwise alignment*), te poravnanje više slijedova (engl. *multiple sequence alignment*).

Postoje mnogi algoritmi za poravnanje parova slijedova, a neki od najpoznatijih su Smith-Waterman algoritam za lokalno poravnanje i Needleman-Wunch algoritam za globalno poravnanje. Smith-Waterman algoritam pronalazi slične regije između dva slijeda dok Needleman-Wunch algoritam poravnava oba slijeda u potpunosti.

Višestruko poravnanje je poravanje tri ili više slijedova. Jedna od najpopularnijih metoda višestrukog poravnanja je poravnanje korištenjem grafa parcijalnog uređaja - POA (Partial Order Alignment) grafa. Poravnanjem dvaju slijedova nekim od algoritama poput Smith-Watermanov nastaje POA graf. Daljni slijedovi poravnavaju se s POA grafom pri ćemu se svaki slijed poravnanjem uključuje u sam graf.

Problem koji se javlja kod poravnanja korištenjem grafa parcijalnog uređaja je nemogućnost paralelizacije tog postupka. U ovom radu se izlaže algoritam koji pokušava zaobići taj problem poravnanjem više grafova parcijalnog uređaja. To omogućuva da se paralelno izgradi više POA grafova i na kraju svi tako dobiveni grafovi poravnaju.

2. Pregled područja

U ovom poglavlju proći ću kroz teorijsku pozadinu potrebnu za definiranje grafa parcijalnog uređaja i njegova poravnanja sa drugim slijedovima, te kako on izgleda i kako se gradi.

2.1. Hammingova i Levenshteinova udaljenost

Poravnanje dvaju nizova zahtjeva njihovu usporedbu, te je zbog toga potrebno definirati kako uopće usporediti dva niza.

Američki znanstvenik Richard Hamming 1950. godine definirao je mjeru udaljenosti između dva niza jednakih duljina, još poznatu kao Hammingova udaljenost. To je minimalan broj znakova koje je potrebno supstituirati u jednom nizu kako bi on postao jednak drugom nizu. Na primjer, mjera udaljenosti nizova $x = AGTA$ i $y = ACTA$ jednaka je jedan zato što je potrebno zamijeniti samo jedna znak u nizu x ili y kako bi oni postali isti. No ovakva definicija nije prikladna za područje bioinformatike zato što ona zahtjeva da su nizovi koji se uspoređuju jednakih duljina, što bi bilo preveliko ograničenje za usporedbu bioloških slijedova.

Definiciju usporedbe dvaju nizova proizvoljne duljine donio je 1965. godine sovjetski matematičar Vladimir Levenshtein. On je definirao udaljenost uređivanja između dva niza, poznatu kao Levenshteinova udaljenost, kao minimalan broj potrebnih jednoznakovnih operacija kako bi nizovi postali jednaki. Moguće jednoznakovne operacije su:

- umetanje znaka
- brisanje znaka
- supstitucija znaka

Cijena svih operacija je jednaka i iznosi jedan.

Formalno, Levenshteinovu udaljenost između dva niza x i y možemo definirati kao

vrijednost sljedeće funkcije:

$$lev(x,y) = \begin{cases} n, & \text{ako je } m = 0, \\ m, & \text{ako je } n = 0, \\ lev(rep(x), rep(y)) & \text{ako je } x[0] = y[0], \\ 1 + min \begin{cases} lev(rep(x), y), \\ lev(x, rep(y)), \\ lev(rep(x), rep(y)) \end{cases} & \text{inače} \end{cases}$$

gdje je n duljina niza x, m duljina niza y, a $rep(z)$ je funkcija koja vraća sve znakove niza z osim prvog.

2.2. Algortimi poravnanja parova nizova

Poravnanje dvaju nizova možemo predstaviti kao tablicu s dva retka. Prvi redak sadrži znakove prvog niza u njihovom orginalnom redoslijedu, a drugi redak sadrži znakove drugog niza isto u očuvanom poretku. Promotrimo dva niza $x = AGTGCA$ i $y = GTGGAA$. Moguća tablica poravnanja ta dva niza prikazana je tablicom 2.1.

Tablica 2.1: Moguća tablica poravnanja nizova $x = AGTGCA$ i $y = GTGGAA$

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

Slučaj u kojoj su oba slova u stupcu tablice jednaka nazivamo slaganje (engl. *match*), a slučaj u kojoj su različita nazivamo neslaganje (engl. *mismatch*). Uvodimo poseban znak '-' koji predstavlja umetanje znakova u prvi niz ili brisanje znakova drugog niza (engl. *gap*). U ovom primjeru ukupno je potrebno provesti tri jednoznakovne operacije (jednu operaciju umetanja, jednu operaciju brisanja i jedno operaciju supstitucije) da bi nizovi x i y postali jednaki te je zbog toga Levenshteinova udaljenost jednaka 3.

2.2.1. Needleman-Wunsch algoritam

1970. godine znanstvenici Saul B. Needleman i Christian D. Wunsch razvili su jedan od prvih algoritama za usporedbu bioloških slijedova koji koristi načelo dinamičkog programiranja, poznat još kao i algoritam za globalno poravnanje [6]. Ovaj algoritam

računa sličnosti dvaju nizova na sličan način poput funckije za izračun Levenshteinove udaljenosti, no uz dvije bitne razlike:

- ubacivanje i brisanje kažnjavaju se negativnom vrijednošću d
- vrijednost slaganja ili neslaganja temelji se na funkciji $T(a, b)$

Algoritam provodimo građenjem dvodimenzionalne matrice veličine $(n+1)*(m+1)$ gdje su n i m duljine nizova x i y. Relacija za računanje vrijednosti svake pojedine ćelije matrice (i, j) izgleda ovako:

$$NW(i,j) = \begin{cases} 0, & \text{ako je } i = 0 \text{ i } j = 0, \\ j * d, & \text{ako je } i = 0, \\ i * d, & \text{ako je } j = 0, \\ max \begin{cases} NW(i-1,j) + d, \\ NW(i,j-1) + d, \\ NW(i-1,j-1) + T(x[i-1], y[j-1]) \end{cases} & \text{inače} \end{cases}$$

Konačna mjera sličnosti nizova x i y bit će vrijednost ćelije (n, m).

U svom radu Needleman i Wunsch vrijednost funkcije $T(a, b)$ definirali su točno za svaki pojedini par aminokiselina. No, općenito funkciju $T(a, b)$ možemo zamisliti u sljedećem obliku:

$$T(a,b) = \begin{cases} match, & \text{ako je } a = b \\ mismatch, & \text{ako je } a \neq b \end{cases}$$

Ako su znakovi a i b jednaki tada funkcija vraća vrijednost *match*, a ako su različiti vraća vrijednost *mismatch*.

Za računanje vrijednosti pojedine ćelije (i, j) potrebne su nam vrijednosti samo ćelija $(i-1, j-1)$, $(i, j-1)$ i $(i-1, j)$. Time je veliki problem poravnanja nizova sveden na veći broj manjih problema računanja vrijednosti pojedine ćelije matrice, što je temelj dinamičkog programiranja.

Ako za svaku ćeliju zapamtimo na temelju koje susjedne ćelije smo njenu vrijednost izračunali možemo izgraditi put poravnanja krenuvši od posljednje do prve ćelije matrice. Takvih putova može biti više što znači da postoji više optimalnih poravnanja. Slika 2.1 prikazuje četiri moguća poravnanje nizova $x = AGTGCA$ i $y = GTGGAA$, koja možemo prikazati tablicama 2.2.

		0	1	2	3	4	5	6
			G	T	G	G	A	A
0		0	-1	-2	-3	-4	-5	-6
1	A	-1	-1	-2	-3	-4	-3	-4
2	G	-2	0	-1	-1	-2	-3	-4
3	T	-3	-1	1	0	-1	-2	-3
4	G	-4	-2	0	2	1	0	-1
5	C	-5	-3	-1	1	1	0	-1
6	A	-6	-4	-2	0	0	2	1

Slika 2.1: Matrica izgrađena na temelju algoritma za globalno poravnanje za nizove $x = AGTGCA$ i $y = GTGGAA$ gdje je $d = -1$, a funkcija $T(a, b)$ vraća 1 ako su znakovi a i b jednaki, odnosno -1 ako su različiti.

Tablica 2.2: Tablice poravnanja nizova $x = AGTGCA$ i $y = GTGGAA$ na temelju matrice poravnanja sa Slike 2.1.

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

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

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

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

2.2.2. Smith-Waterman algoritam

Smith-Waterman algoritam su 1981. godine predložili znastvenici Temple F. Smith i Michael S. Waterman [7]. Ovaj algoritam je varijacija na Needleman-Wunsch algoritam, ali za razliku od Needleman-Wunsch algoritma koji traži sličnosti između nizova u njihovoj cijelosti, Smith-Waterman algoritam sličnosti traži u njihovim dijelovima. Zbog toga se još naziva i algoritam za lokalno poravnanje.

Kao i Needleman-Wunsch algoritam, Smith-Waterman algoritma za poravnanje nizova x i y provodimo građenjem dvodimenzionalne matrice veličine $(n+1)*(m+1)$ gdje su n i m duljine nizova x i y. Ćelije te matrice računamo na temelju sljedeće relacije:

$$SW(i,j) = \begin{cases} 0, & \text{ako je } i = 0 \text{ ili } j = 0 \\ max \begin{cases} 0, \\ SW(i-1,j) + d, \\ SW(i,j-1) + d, \\ SW(i-1,j-1) + T(x[i-1], y[j-1]) \end{cases} & \text{inače} \end{cases}$$

U Smith-Waterman algoritmu sve ćelije imaju minimalnu vrijednost 0 i ćelija koja sadrži konačnu mjeru sličnosti je ćelija sa najvećom vrijednošću. Put poravnanja kreće od ćelije sa najvećom vrijednošću sve do ćelije čija je vrijednost 0. Kao i u Needlman-Wunsch algoritmu može se pojaviti više mogućih putova, a time i više mogućih poravnanja.

Tablica 2.3: Tablice poravnanja nizova $x = AAGTCA$ i $y = TCTCG$ na temelju matrice poravnanja sa Slike 2.2. Mala slova označuju neporavnate dijelove nizova.

aag	T	C	a
tc	T	C	g
a	T	C	tca
	T	C	tcg

		0	1	2	3	4	5
			T	C	T	C	G
0		0	0	0	0	0	0
1	A	0	0	0	0	0	1
2	A	0	0	0	0	0	1
3	G	0	0	0	0	0	0
4	T	0	1	0	1	0	0
5	C	0	0	2	1	2	1
6	A	0	0	1	1	1	1

Slika 2.2: Matrica izgrađena na temelju algoritma za lokalno poravnanje za nizove $x = AAGTCG$ i $y = ACGTCA$ gdje je $d = -1$, a funkcija $T(a, b)$ vraća 1 ako su znakovi a i b jednaki, odnosno -1 ako su različiti.

2.3. Poravnanje više slijedova

Poravnanje više slijedova (engl. *Multiple Sequence Alignment*) je poravnanje tri ili više bioloških slijedova sličnih duljina. Poravnanje više slijedova omogućuje pronalaženje sličnih regija između većeg broja slijedova, te je kao takvo neophodno u području biologije i bioinformatike [1].

Algoritmi dinamičkog programiranja parova slijedova predstavljeni u prethodnom poglavlju efikasni su za poravnanje dva slijeda, te se mogu proširiti za poravnanje više slijedova, ali njihovo korištenje na više od dva slijeda vrlo brzo postaje vremenski i prostorno presloženo. Razlog tome je što broj usporedba raste eksponencionalno s rastom broja slijedova.

Još jedan problem koji se javlja kod proširanja algoritama za poravnanje parova slijedova na poravnanje više slijedova je taj što poravnanje dvaju slijedova ne mora biti jedinstveno. Slika 2.1 prikazuje jedan takav slučaj gdje postoje čak četiri različita optimalna poravnanja. Ako želimo dalje poravnavati te slijedova s nekim trećim moramo izabrati jedno od mogućih poravnanja, te se javlja problem koje od tih poravnanja trebamo izabrati.

Veliki napredak u poravnanju više slijedova donose progresivne metode poravnanja koje se temelje na izgradnji stabala poravnanja. Jedna od najpopularnijih progresivnih metoda poravnanja je poravnanje grafa parcijalnog uređaja.

Graf parcijalnog uređaja i algoritam poravnanja graf parcijalnog uređaja i slijeda predstavili su u svom radu 2002. godine znanstvenici Christopher Lee, Catherine Grasso i Mark F. Sharlow [5].

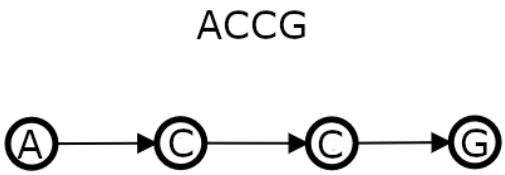
2.3.1. Graf parcijalnog uređaja

Graf parcijalnog uređaja (POA graf) je usmjeren povezan aciklički graf u kojem čvorovi predstavljaju znakove nekog slijeda, a usmjerene veze u grafu povezuju dva uzastopna znaka unutar tog slijeda. Slika 2.3 prikazuje način pretvorbe slijeda u graf parcijalnog uređaja.

Svako poravnanja dvaju slijedova možemo prikazati kao graf parcijalnog uređaja koji izgrađujemo na sljedeći način (Slika 2.4):

1. Oba slijeda pretvorimo u graf parcijalnog uređaja.

2. Za svaki par znakova u matrici poravnanja:
   - Ako su znakovi jednaki, onda čvorove spajamo

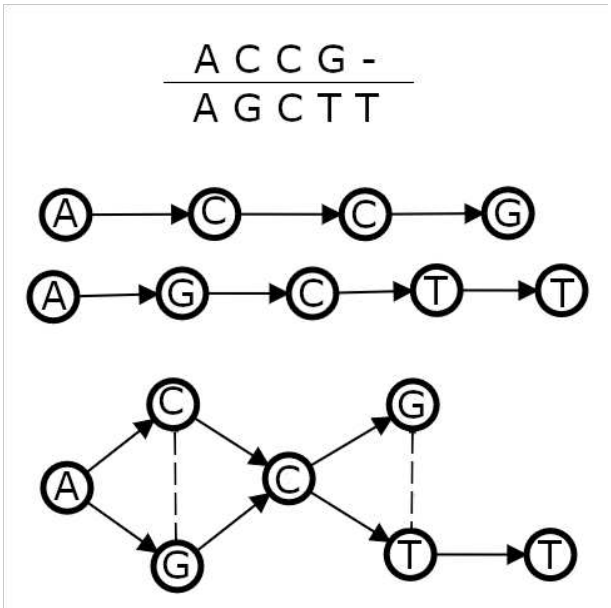


Slika 2.3: Pretvorba slijeda $x = ACCG$ u graf parcijalnog uređaja

– Ako su znakovi različiti, onda naznačimo da su ta dva čvora poravnata

Kako bi se izbjegao gubitak informacija o orginalnim slijedovima koji čine POA graf svaki čvor grafa sadrži podatake o slijedovima iz kojeg dolazi i o položaju koji ima unutar tih slijedova. Zbog toga možemo rekonstruirati sve slijedove koji ga čine.

Grafom parcijalnog uređaja moguće je prikazati više različitih poravnanja (ali ne sve) čime se riješava problem odabira između više mogućih optimalnih poravnanja.



Slika 2.4: Pretvorba poravnanja nizova $x = ACCG$ i $y = AGCTT$ u graf parcijalnog uređaja. Isprekidane linije predstavljaju poravananje dvaju čvorova.

2.3.2. Poravnanje grafa parcijalnog uređaja i slijeda

Na temelju poravnanja dvaju slijedova možemo izgraditi graf parcijalnog uređaja. Kako bi ostvarili poravnanje više slijedova potrebno je definirati kako poravnati graf parcijalnog uređaja i slijed.

Poravnanja grafa parcijalnog uređaja i slijeda provodi se pomoću nekog od opisanih algoritama za poravnanje parova slijedova. U ovom primjeru koristit ćemo algoritam globalnog poravnanja. Postupak poravnanja provodimo izgradnjom dvodimenzionalne matrice veličine $(n+1)*(m+1)$ gdje je n broj čvorova POA grafa, a m duljina niza y. Relacija za računanje pojedine ćelije matrice izgleda dana je sljedećom formulom:

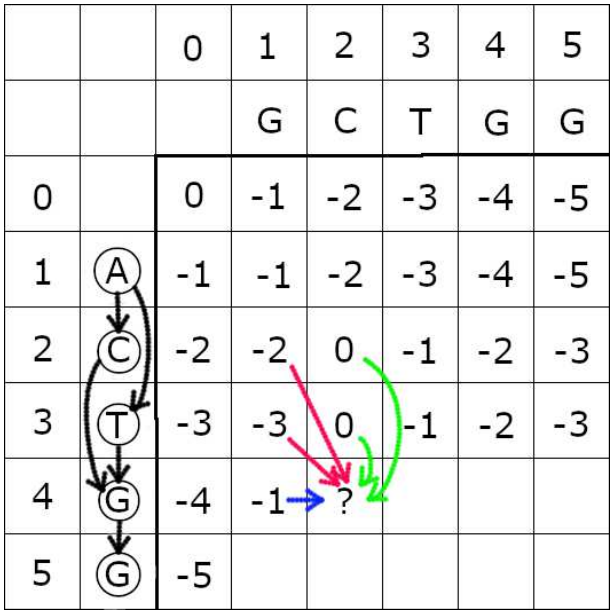
$$S(i,j) = \begin{cases} 0, & \text{ako je } i = 0 \text{ i } j = 0, \\ j * d, & \text{ako je } i = 0, \\ i * d, & \text{ako je } j = 0, \\ max \begin{cases} S(p, j-1) + T(i,j), \\ S(p,j) + d, \\ S(i, j-1) + d, \end{cases} & \text{inače} \end{cases}$$

gdje je p index svih čvorova prethodnika čvoru na trenutnom indeksu i. Konačna mjera sličnosti više neće biti nužno vrijednost posljednje ćelije matrice, već ćelije za koju je čvor koji promatramo posljedni čvor grafa, te znak posljedni znak slijeda. Takvih ćelija može biti više te za mjeru sličnosti uzimamo najvišu vrijednost među tim ćelijama.

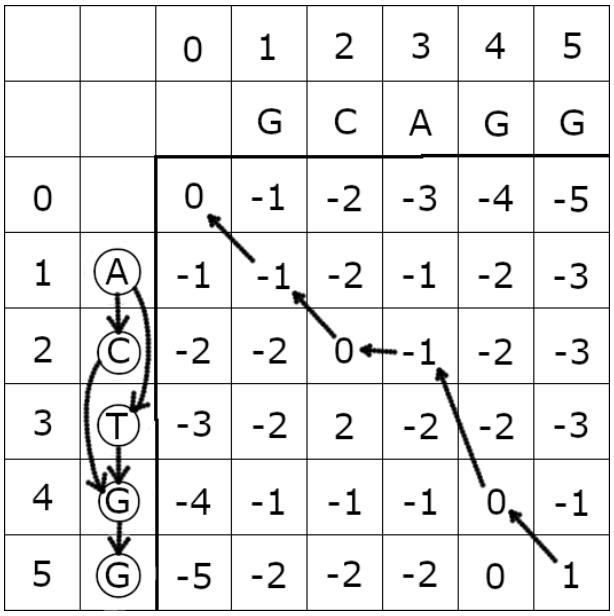
Glavna razlika između poravnanja dva slijeda i poravnanja grafa parcijalnog uređaja i slijeda je što svaki znak unutar slijeda ima točno jednog prethodnika, dok u grafu jedan znak (tj. čvor) može imati više prethodnika. Zbog toga je za izračun vrijednosti neke ćelije potrebno proći kroz sve čvorove prethodnike (Slika 2.6). Kako bi te vrijednosti mogli izračunati moramo osigurati da su ćelije svih prethodnika trenutnog čvora izračunate u ranijim koracima te zato moramo POA graf topološki sortirati. Topološkim sortiranjem POA grafa dobivamo niz svih njegovih čvorova poredanih tako da se svi sljedbenici nekog čvora uvijek nalaze iza tog čvora.

Za izračun vrijednosti pojedine ćelije matrice prilikom poravnanja dva slijeda bilo je potrebno ukupno 3 usporedbe, u slučaju poravnanja grafa i slijeda biti će nam potrebno sveukupno $(2 * p) + 1$ usporedba gdje je p prosječan broj čvorova prethodnika u grafu.

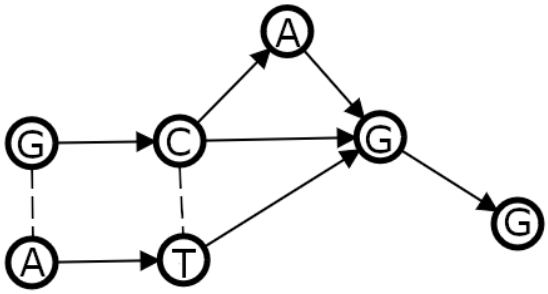
Ako i dalje za svaku ćeliju pamtimo na temelju koje ćelije smo izračunali njenu vrijednost, krenuvši od ćelije čiju vrijednost smo proglasili konačnom mjerom sličnosti možemo slijed uključiti u graf parcijalnog uređaja (Slike 2.6 i 2.7).



Slika 2.5: Izračun vrijednosti ćelije prilikom poravnanje POA grafa i slijeda zahtjeva prolazak kroz ćelije svih prethodnika čvora grafa.



Slika 2.6: Matrica poravnanja POA grafa i slijeda izgrađena na temelju algoritma za globalno poravnanje gdje je d jednak -1, a funkcija $T(a, b)$ vraća 1 ako su znakovi a i b jednaki, odnosno -1 ako su različiti



Slika 2.7: Graf parcijalnog uređaja nastao na temelju poravnanja na slici 2.6. Isprekidane linije predstavljaju poravananje dvaju čvorova.

2.4. Konstrukcija konsenzusa grafa parcijalnog uređaja

Konsenzus je statistički profil poravnanja više slijedova. Postoje različiti načini konstrukcije konsenzusa. U ovom poglavlju predstavit će se algoritam dinamičkog programiranja za konstrukciju konsenzusa grafa parcijalnog uređaja kojeg je predložio Christopher Lee u svom radu 2003. godine [4]. Problem konstrukcije konsenzusa svodi na problem pronalaska najboljeg mogućeg puta kroz graf parcijalnog uređaja.

Algoritam za izgradnju konsenzusa provodi se u nekoliko koraka:

1. Topološki sortiramo graf parcijalnog uređaja.

2. Svim čvorovima početno pridjelimo vrijednost 0.

3. Prolazeći kroz topološki sortiranu listu čvorova, za svaki čvor odaberemo ulaznu vezu najveće težine. Težinu veze definiramo kao broj slijedova koji njome prolaze. Ako je više ulaznih veza iste težine, izabiremo onu čiji početni čvor ima najveću vrijednost.

4. Za svaki čvor zabilježimo odabranu vezu i tom čvoru pridjelimo vrijednost jednaku težini odabrane veze + vrijednosti početnog čvora te veze.

5. Nakon što prođemo kroz sve čvorove, krenuvši od čvora s najvećom vrijednošću sljedimo odabrane ulazne veze sve dok ne dođemo do jednog od početnih čvorova. Staza od čvora s najvećom vrijednošću do početnog čvora čini konsenzus grafa parcijalnog uređaja.

U rijetkim slučajevima može se dogoditi da čvor s najvećom vrijednošću nije posljednji čvor grafa tj. da čvor ima izlazne veze. U tom slučaju svim čvorovima osim čvora s najvećom vrijednošću postavimo vrijednost na neki negativan broj. Zatim ponovimo prethodno opisan algoritam uz uvjet da za neki čvor nikada ne odabiremo vezu ako je izvor te veze čvor s negativnom vrijednošću. Ovime će se osigurati da novi put koji stvorimo kreće od prethodno odabranog čvora jer on jedini ima nenegativnu vrijednost. Ovaj postupak ponavaljamo sve dok odabrani čvor nije posljedni čvor grafa.

3. Poravnanje dva grafa parcijalnog uređaja

U prošlom poglavlju opisano je kako iz poravnanja dvaju slijedova nastaje POA graf, te kako dalje uspoređivati taj graf s drugim slijedovima pritom proširujući sam graf.

Problem koji se javlja s višestrukim poravnanjem korištenjem POA grafa je nemogućnot paralelizacije, a time i ubrzanja, postupka poravnanja. Slijedove moramo iterativno jedan po jedan poravnavati s grafom. Jedno od riješenja tog problema je razviti algoritam poravnanja dva grafa parcijalnog uređaja. To bi omogućilo paralelizaciju postupka izgradnje više POA grafova. U ovom poglavlju izlaže se jedan takav algoritam.

3.1. Poravnanje grafova

Algoritam poravnanja dva grafa parcijalnog uređaja predstavljen u ovom radu temelji se na proširenju algoritma za poravnanje grafa i slijeda. U algoritmu za poravnanje dva grafa biti će potrebno provjeravati za svaku ćeliju matrice sve prethodnike čvorova oba grafa. Kao i poravnanje grafa i slijeda, poravnanje dva grafa temeljit će se na jednom od algoritama poravnanja parova slijedova (u daljnjim primjerima koristit ćemo Needlman-Wunsch algoritam za globalno poravnanje).

Poravnanje POA grafova G i F provodimo izgradnjom dvodimenzionalne matrice veličine $(n + 1) * (m + 1)$ gdje je n broj čvorova grafa G, a m broj čvorova grafa F.

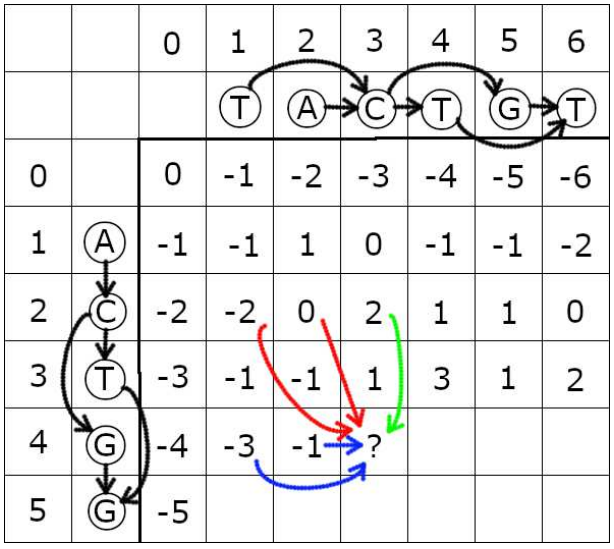
Računanje pojedine ćelije matrice dano je sljedećom relacijom:

$$S(i,j) = \begin{cases} 0, & \text{ako je } i = 0 \text{ i } j = 0, \\ j * d, & \text{ako je } i = 0, \\ i * d, & \text{ako je } j = 0, \\ max \begin{cases} S(p,q) + T(i,j), \\ S(p,j) + d, \\ S(i,q) + d, \end{cases} & \text{inače} \end{cases}$$

gdje je p index svih čvorova prethodnika čvoru grafa G na trenutnom indeksu i, a q index svih čvorova prethodnika čvoru grafa F na trenutnom indeksu j. Konačna mjera sličnosti bit će vrijednost ćelije za koju su oba čvora posljedni čvorovi svojeg grafa. Takvih ćelija može biti više, te odabiremo onu s najvišom vrijednošću.

Prilikom poravnanja POA grafa i slijeda bilo je potrebno provesti ukupno $(2*p)+1$ usporedba gdje je p prosječan broj čvorova prethodnika u POA grafu. Za poravnanje dva POA grafa G i F biti će potrebno provesti $2 * p + 2 * q - 1$ usporedba gdje je p prosječan broj čvorova prethodnika u grafu G, a q prosječan broj prethodnika u grafu F.

Kako bi mogli izračunati vrijednost svake ćelije matrice, za svaki čvor moramo imati izračunate ćelije svih njegovih prethodnika te zbog toga moramo topološki sortirati oba grafa prije poravnanja.

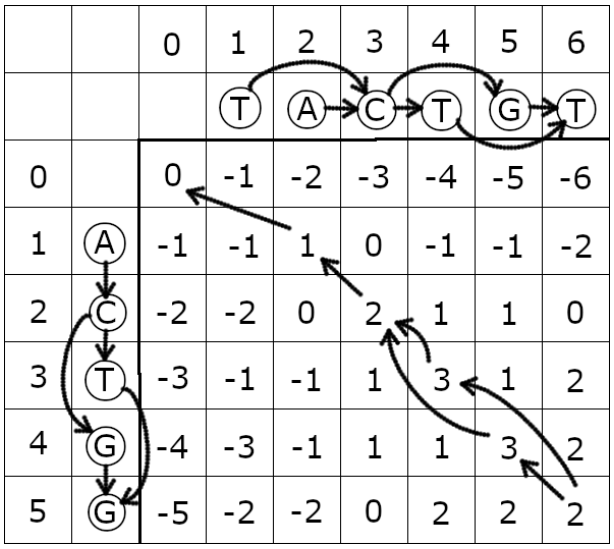


Slika 3.1: Izračun vrijednosti ćelije prilikom poravnanja dva POA grafa zahtjeva prolazak kroz ćelije svih prethodnika čvorova oba grafa.

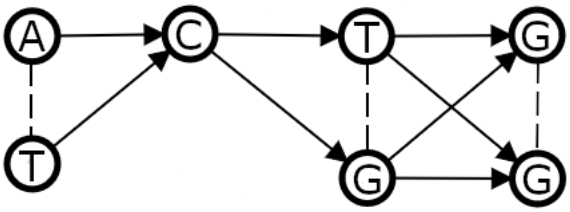
3.2. Izgradnja grafa na temelju poravnanja

Na temelju matrice poravnanja možemo izgraditi novi graf parcijalnog uređaja koji uključuje oba poravnata grafa. Kako bi spojili dva POA grafa na temelju njihova poravnanja moramo pamtiti za svaku ćeliju matrice na temelju koje druge ćelije je njena vrijednost izračunata - tu ćeliju ćemo nazivati roditeljska ćelija. Postupak spajanja provodimo na sljedeći način, krenuvši od ćelije čiju smo vrijednost odabrali kao konačnu mjeru sličnosti:

1. Gledamo vezu trenutne ćelije prema roditeljskoj ćeliji
   - Ako je veza prema roditeljskoj ćeliji dijagonalna:
     (a) Ako su znakovi unutar čvorova jednaki tada spajamo čvorove, a ako su znakovi unutar čvorova različiti tada zabilježimo da su ta dva čvora poravnata
     (b) Provjerimo sve čvorove sa kojima su trenutni čvorovi prethodno poravnati te ih spajamo ili poravnamo ovisno jesu li im znakovi unutar čvora jednaki ili ne
   - Ako je veza prema roditeljskoj ćeliji vertikalna ili horizontalna nastavljamo dalje s postupkom
2. Sljedimo vezu prema roditeljskoj ćeliji i ponavljamo postupak sve dok ne dođemo do početne ćelije matrice



Slika 3.2: Matrica poravnanja dva POA grafa izgrađena na temelju algoritma za globalno poravnanje gdje je $d = -1$, a funkcija $T(a, b)$ vraća 1 ako su znakovi a i b jednaki, odnosno -1 ako su različiti



Slika 3.3: Graf parcijalnog uređaja nastao na temelju poravnanja grafova sa Slike 3.2. Isprekidane linije predstavljaju poravananje dvaju čvorova.

4. Implementacija

U ovom poglavlju predstaviti će se kod koji implementira algoritam za poravnanje dva grafa parcijalnog uređaja opisan u prošlom poglavlju. Za implementaciju algoritma korišten je programski jezik C++. Potpun kod moguće je vidjeti na sljedećem repozitoriju: `https://github.com/marin-rasic/poa`.

4.1. Podatkovne strukture

Za implementaciju POA grafa koriste se tri podatkovne strukture:

- čvor (*Node*)
- veza (*Edge*)
- graf (*Graph*)

Čvor sadrži podatake o znaku koji ga čini, o svim ulaznim i izlaznim vezama, podatke o slijedovima iz kojih dolazi znak i indeksu znaka unutar tih slijedova, te listu svih drugih čvorova s kojima je ovaj čvor poravnat. Veze sadrže podatak o početnom i završnom čvoru, a graf listu početnih čvorova.

U nastavku je dan pojednostavljeni prikaz navedenih podatkovnih struktura:

```
class Node {
    char letter;
    vector<std::tuple<const char *, unsigned int>> origin_of_letter;
    vector<Edge *> incoming_edges;
    vector<Edge *> outgoing_edges;
    vector<Node *> aligned_nodes;
};

class Edge {
    Node *origin;
    Node *destination;
```

```
};

class Graph {
    vector<Node *> start_nodes;
};
```

4.2. Topološko sortiranje grafa

Prije nego što poravnamo dva grafa parcijalnog uređaja potrebno ih je topološki sortirati. Topološkim sortiranjem dobivamo niz svih čvorova grafa poredanih tako da se svi sljedbenici nekog čvora uvijek nalaze iza tog čvora. U implementaciji za topološko sortiranje se koristi modificirani Khanov algoritam [3].

U nastavku je prikazana pojednostavljena verzija implementacije algoritma za topološko sortiranje.

```
vector<Node *> Graph::TopologicalSort() {
    vector<Node *> sorted_nodes;
    queue<Node *> nodes;

    for (Node *n : this->start_nodes) {
        nodes.push(n);
    }

    while (!nodes.empty()) {
        Node *node = nodes.front();
        nodes.pop();
        sorted_nodes.push_back(node);
        for (Edge *e : node->outgoing_edges) {
            if (--e->destination->num_remain_edges == 0) {
                nodes.push(e->destination);
            }
        }
    }

    return sorted_nodes;
}
```

Topološko sortiranje grafa započinje dodavanjem svih njegovih početnih čvorova (za koje znamo da nemaju ulazih veza) u red $nodes$. Dok god red $nodes$ nije prazan iz njega uzimamo prvi čvor i stavljamo ga u listu sortiranih čvorova $sorted\_nodes$. Nakon toga provjeravamo sve izlazne veze tog čvora. Za svaku vezu, gledamo čvor destinaciju te veze, te smanjujemo varijablu num_remain_edges za 1 tom čvoru. Ako je ona tada jednaka 0, to znači da svi čvorovi koji imaju ulaznu vezu u taj čvor su već dodani u red $nodes$ te i sam čvor možemo dodati u $nodes$. Varijabla num_remain_edges početno ima vrijednost jednaku broju ulaznih veza čvora.

4.3. Implementacija poravnanja grafova parcijalnog uređaja

U nastavku ću prikazati pojednostavljeni kod implementacije poravnanja dva grafa parcijalnog uređaja. Kod u nastavku će biti prilagođen za globalno poravnanje.

```
tuple<int, int> Aligner::AlignTwoGraph(
    vector<vector<Cell>> &matrix,
    vector<Node *> &G,
    vector<Node *> &F) {

 Cell max_cell;
 tuple<int, int> index;


 for (int i = 1; i < G.size() + 1; i++) {
  for (int j = 1; j < F.size() + 1; j++) {
   int alig_value;
   if (G[i-1] == F[j-1]) {
    alig_value = match;
   } else {
    alig_value = mismatch;
   }

   for (Edge *F_edge : F[j - 1]->incoming_edges) {
    for (Edge *G_edge : G[i - 1]->incoming_edges) {
      diagonal_cell = compare_cells(
```

```
        diagonal_cell,
        matrix[G_edge->origin->index][F_edge->origin->index].value
        + alig_value);

       up_cell = compare_cells(
        up_cell,
        matrix[G_edge->origin->index][j].value
         + gap);

     }
     left_cell = compare_cells(
      left_cell,
      matrix[i][F_edge->origin->index].value + gap);
    }
   }

   matrix[i][j] = max({diagonal_cell, up_cell, left_cell});

   if (G_graph[i - 1]->outgoing_edges.empty()
    && F_graph[j - 1]->outgoing_edges.empty()
    && align_matrix[i][j].value >= max_cell.value) {
     max_cell = matrix[i][j];
     index = tuple<int, int>(i, j);
   }

  }

  return global_index;
}
```

Funckija za poravnanje dva grafa kao ulazne parametre prima matricu $matrix$ veličine $(n + 1) * (m + 1)$ gdje je n broj čvorova grafa G, a m broj čvorova grafa F. Grafovi G i F na ulazu su zapravo nizovi njihovih topološki sortiranih čvorova.

Glavna petlja prolazi kroz cijelu matricu te za svaku ćeliju matrice prolazi kroz sve ulazne veze ($incoming\_edges$) čvorova koji odgovaraju indeksu te ćelije. U tom prolazu zabliježe se vrijednost najveće dijagonlane, gornje i lijeve ćelije - $diagonal\_cell$,

up_cell i $left\_cell$. Na kraju svake iteracije glavne petlje se za konačnu vrijednost trenutne ćelije odabire najveća vrijednost zabilježenih ćelija. Ako su oba čvora posljedni čvor grafa i veći su od vrijednosti $max\_cell.value$ zapisuje se trenutna ćelija i njen indeks u varijable max_cell i $index$. Na kraju se vraća index odgovarajuće ćelije.

U prikazu ovog koda preskočen je dio koji zabilježava indeks roditeljske ćelije za svaku pojedinu ćeliju.

5. Rezultati

Jedan od ciljeva ovog završnog rada bilo je pokazati kako uvođenjem poravnanja dva POA grafa možemo ubrzati postupak višestrukog poravnanja, jer nam poravnanje dva POA grafa omogućuje paralelizaciju tog postupka.

Testiranje ubrzanja provelo se nad skupovima slijedova duljine oko 500 znakova. Za slučjano odabrane skupove postupak poravnanja proveden je na dva načina:

- Iterativno poravnanja slijeda po slijed s POA grafom koji se početno izgradi na temelju prva dva slijeda.
- Turnirsko stablo. Poravnanje parova slijedova, te stvaranje POA grafova na temelju svakog poravnanja. Zatim se provodi poravnanje dva po dva POA grafa sve dok ne ostanemo samo jedan graf. Ovaj postupak moguće je paralelizirati.

Rezultati ovog testiranja pokazuju da se paralelizacijom ubrzao postupak poravnanja. Za postupak paralelizacije korišten je alat $OpenMP$. Tablica 5.1 prikazuje par primjera. Ubrzanje definitvno postoji, no ono nije vrlo konstantno. Ovisno o skupu podataka ubrzanja će biti različitih veličina.

Tablica 5.1: Tablica usporedbe vremena izvođenja programa na nekoliko primjera

Bez paralelizacije	Korištenjem 2 dretve	Korištenjem 4 dretve
7.89 s	4.5 s	3.95 s
10.97 s	7.6 s	7.03 s
21.16 s	13.97 s	11.52 s
12.96 s	9.69 s	6.89 s

Dodatno, provedena je i usporedba mog rješenja s već postojećim programom *Spoa* (`https://github.com/rvaser/spoa`). Uspoređena je brzina rada programa i točnost konsenzusa koji se generira. Koristilo se globalno poravnanje sa linearnim procijepima, vrijednost $d = -8$, a funkcija $T(a, b)$ vraća 5 ako su znakovi a i b jednaki, odnoso -4 ako su različiti. Rezultati za par primjera dani su tablicom 5.2. Izlazi oba programa su konsenzusi konačnog POA grafa. Pogreška je definirana kao razlika između generiranog konsenzusa i stvarnog konsenzusa. Za usporedbu konsenzusa koristio se program *edlib* (`https://github.com/Martinsos/edlib`). Kao što je očekivano *Spoa* program je znatno brži i točniji od programa predstavljenog u ovom radu.

Tablica 5.2: Tablica usporedbe programa predstavljenog u ovom radu i Spoa na nekoliko primjera. Različiti primjeri imaju različiti broj slijedova koji se poravnavaju.

	Program izložen u ovom radu		Spoa	
	Vrijeme (s)	Pogreška ($\%$)	Vrijeme (s)	Pogreška ($\%$)
1.	4.03	5.4	0.059	3.8
2.	9.32	4.8	0.193	2.8
3.	9.77	9.7	0.373	1.2
4.	8.2	9.2	0.188	0.8
5.	9.8	10	0.407	1
6.	13.08	15.2	0.73	1.4

6. Zaključak

Poravnanje bioloških slijedova jedan je od glavnih problema područja bioinformatike. Algoritmi za poravnanje parova slijedova, poput Needleman-Wunsch i Smith-Waterman algoritma, temelj su mnogim drugim algoritmima. Oni su vrlo efikasni za poravnanje parova slijedova, ali njihovo korištenje za višestruka poravnanja je daleko preskupo. Za potrebe višestrukih poravnanja razvili su se nove progresivne metode od kojih je jedna i algoritam za poravnanje grafa parcijalnog uređaja i slijeda. Graf parcijalnog uređaja ili POA graf je način prikaza poravnanja više slijedova bez gubitka informacija. Problem kod algoritma poravnanja grafa parcijalnog uređaja i slijeda je nemogućnost njegove paralelizacije za ostvarenje višestrukog poravnanja.

U ovom radu izložen je algoritam za poravnanje više grafova parcijalnog uređaja. Algoritam se temelji na proširenju postojećeg algoritma poravnanja grafa parcijalnog uređaja i slijeda. Ovaj algoritam pokušava riješiti problem nemogućnosti paralelizacije postupka poravnanja POA grafa i slijeda. Prikazana je implementacija algoritma u programu pisanom programskim jezikom C++. Implementacijom tog algoritma uspješno je pokazano da njegovom paralelizacijom možemo ubrzati postupak višestrukog poravnanja.

LITERATURA

[1] P. Bawono, M. Dijkstra, W. Pirovano, A. Feenstra, S. Abeln, i J. Heringa. *Multiple Sequence Alignment*. Springer New York, New York, 2017.

[2] J. Dursi. Understanding partial order alignment for multiple sequence alignment, 2015. URL `http://simpsonlab.github.io/2015/05/01/understanding-poa/`. Pristupljeno: 20.04.2021.

[3] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[4] C. Lee. Generating consensus sequences from partial order multiple sequence alignment graphs. *Bioinformatics*, 19(8):999–1008, 2003.

[5] C. Lee, C. Grasso, i M. F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[6] S. B. Needleman i C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[7] T.F. Smith i M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

Poopćenje algoritma za poravnanje parcijalnog uređaja

Sažetak

Poravnanje bioloških slijedova jedna je od temeljnih zadaća područja bioinformatike. Među algoritmima poravnanja razlikujemo algoritme poravnanja parova slijedova i algoritme poravnanja više slijedova. Jedan od najpopularnijih algoritama višestrukog poravnanja je poravnanje grafa parcijalnog uređaja i slijeda. Problema ovog algoritma je nemogućnost njegove paralelizacije. U ovom radu se predstavlja proširenje tog algoritam na poravnanja dva grafa parcijalnog uređaja što omogućuje njegovu paralelizaciju.

Ključne riječi: bioinformatika, poravnanje parova slijedova, višestruko poravnanje, graf parcijalnog uređaja

Generalization of partial order alignment

Abstract

Alignment of biological sequences is one of the fundamentals tasks of bioinformatics. Alignment algorithms can be divided into two groups: pairwise alignment algorithms and multiple sequence alignment algorithms. One of the most popular multiple sequence alignment algorithms is partial order graph alignment. The problem this algorithm has is its inability to be parallelized. In this paper an algorithm based on partial order graph alignment is presented which aligns two partial order graphs. This added functionality allows it to be parallelized.

Keywords: bioinformatics, pairwise alignment, multiple sequence alignment, partial order graph