UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Marijana R. Miljić

PRONALAŽENJE REGULATORNIH MOTIVA U SKUPU NUKLEOTIDNIH SEKVENCI - ELEKTRONSKA LEKCIJA

master rad

Beograd, 2023.

Mentor:

dr Jovana Kovačević, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Mirjana Maljković, docent
Univerzitet u Beogradu, Matematički fakultet

Aleksandar Veljković, asistent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Mami, tati, sestri i bratu

Naslov master rada: Pronalaženje regulatornih motiva u skupu nukleotidnih sekvenci - elektronska lekcija

Rezime: U ovom radu će biti prikazani algoritmi koji se koriste za pronalaženje regulatornih motiva u skupu nukleotidnih sekvenci. Implementirana je elektronska lekcija koja se sastoji od teorijskog dela i interaktivnog dela. Interaktivni deo omogućuje da se pokrenu algoritmi za pronalaženje regulatornih motiva koji su pisani u *Python*-u, pri čemu je svaki algoritam detaljno opisan. Za lekciju je korišćena interaktivna *Jupyter* sveska. Pored *Jupyter* sveske napravljena je i *HTML* verzija.

Ključne reči: motivi, elektronska lekcija, algoritmi, profilna matrica, bioinformatika

Sadržaj

1 Uvod **1**

2 Deterministički algoritmi **4**

2.1 Prva formulacija 5

2.2 Druga formulacija 7

2.3 Problem niske medijane 12

3 Probabilistički algoritmi **15**

3.1 Pohlepna pretraga motiva 17

3.2 Poboljšana pohlepna pretraga motiva 20

3.3 Probabilistička pretraga motiva 25

3.4 Gibsovo uzorkovanje 28

3.5 EM algoritam 33

4 Uputstvo za korišćenje elektronske lekcije **38**

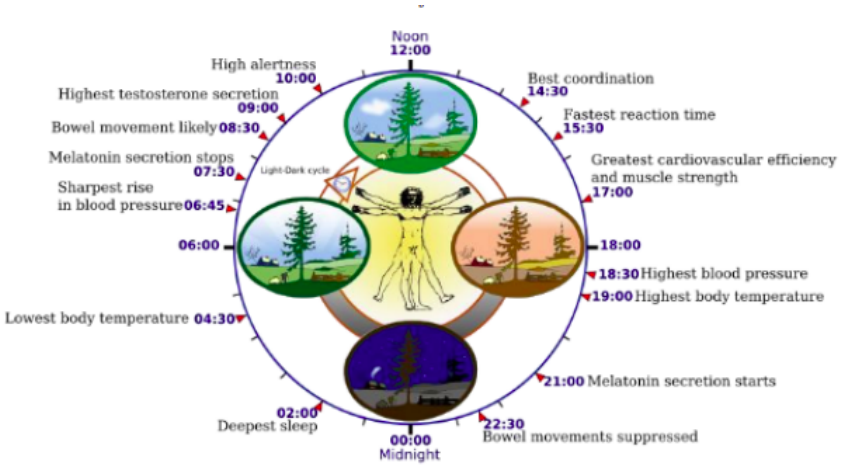
5 Zaključak **40**

Bibliografija **41**

Glava 1

Uvod

Raspored aktivnosti biljaka, životinja i bakterija u toku dana kontroliše „unutrašnji časovnik" koji se naziva cirkadijalni sat. Na Slici 1.1 su predstavljeni različiti procesi koji su kontrolisani ovim satom. Postoji tačno vreme kada telo u toku dana ima najnižu temperaturu, kada kreće i prestaje lučenje hormona melatonina i slično.



Slika 1.1: Cirkadijalni ritam [8]

Cirkadijalnim satom upravljaju molekularni mehanizmi, a jedan od njih je regulacija sinteze proteina. S obzirom na to, možemo postaviti pitanje kako ćelije znaju kada da uspore, a kada da ubrzaju proizvodnju odgovarajućih proteina. Geni koji regulišu cirkadijalni ritam organizama se nazivaju cirkadijalni geni. Ovi geni kontrolišu mnoge procese i funkcije koji se ponavljaju u organizumu na svakih 24 časa, kao što su spavanje, disanje, lučenje hormona i koordiniraju ponašanje stotine drugih gena. Fokusiraćemo se na biljke, jer kod njih cirkadijalni sat igra ključnu ulogu u

njihovom rastu i razvoju. Biolozi procenjuju da su preko hiljadu gena kod biljaka cirkadijalni, uključujući gene koji se odnose na fotosintezu, fotoreceptore i cvetanje. Zaključeno je da svaka ćelija prati dan i noć nezavisno od drugih ćelija i da su tri biljna gena LCY, CCA1 i TOC1, tačnije proteini koje ovi geni kodiraju odgovorna za upravljanje cirkadijalnim genima.

Unutar *DNK* su zapisane informacije za sintezu proteina u ćeliji. Svaki segment *DNK* koji sadrži takvu informaciju predstavlja jedan gen. Pored *DNK*, postoje i različite vrste ribonukleinskih kiselina, odnosno *RNK*. Dok je *DNK* predstavljen dvostrukim lancem koji čine nukleotidi: adenin (A), citozin (C), guanin (G) i timin (T), *RNK* je predstavljen jednostrukim lancem. Umesto timina, kod *RNK* se pojavljuje nukelotid uracil koji se označava sa U. Da bi nastali proteini neophodno je da se na osnovu dva lanca od *DNK* konstruiše *RNK* molekul. Formiranje *RNK* od *DNK* predstavlja jednostavno prepisivanje iz oba lanca *DNK*, uz zamenu nukleotida T sa U. Pomenuto prepisivanje nukeotida predstavlja prvi korak sinteze proteina, tj. transkripciju gena. Početak ovog procesa iniciraju posebna jedinjenja, takozvani transkripcioni faktori (kod biljaka to su uglavnom pomenuti LCY, CCA1 i TOC1), koji se vezuju za kratke *DNK* segmente ispred gena. Ovi kratki segmenti nazivaju se regulatorni motivi.

Drugi korak jeste prevođenje, odnosno translacija, prepisanog *RNK* u proteine. Imamo 4 nukelotida A, C, G, U i njih treba da prevedemo u 20 mogućih aminokiselina. Genetski kod predstavlja preslikavanje skupa kodona (3-grama) u skup aminokiselina (Slika 1.2). Kako preslikavamo redom kodone iz *RNK* u proteine, potrebno je da znamo gde je kraj nekog proteina. Postoje STOP kodoni koji označavaju da iza njih nema više kodona aminokiselina koji kodiraju protein. Ti stop kodoni su UAA, UAG, UGA.

0	AAA	K	16	CAA	Q	32	GAA	E	48	UAA	*
1	AAC	N	17	CAC	H	33	GAC	D	49	UAC	Y
2	AAG	K	18	CAG	Q	34	GAG	E	50	UAG	*
3	AAU	N	19	CAU	H	35	GAU	D	51	UAU	Y
4	ACA	T	20	CCA	P	36	GCA	A	52	UCA	S
5	ACC	T	21	CCC	P	37	GCC	A	53	UCC	S
6	ACG	T	22	CCG	P	38	GCG	A	54	UCG	S
7	ACU	T	23	CCU	P	39	GCU	A	55	UCU	S
8	AGA	R	24	CGA	R	40	GGA	G	56	UGA	*
9	AGC	S	25	CGC	R	41	GGC	G	57	UGC	C
10	AGG	R	26	CGG	R	42	GGG	G	58	UGG	W
11	AGU	S	27	CGU	R	43	GGU	G	59	UGU	C
12	AUA	I	28	CUA	L	44	GUA	V	60	UUA	L
13	AUC	I	29	CUC	L	45	GUC	V	61	UUC	F
14	AUG	M	30	CUG	L	46	GUG	V	62	UUG	L
15	AUU	I	31	CUU	L	47	GUU	V	63	UUU	F

Slika 1.2: Genetski kod [5]

Problem pronalaženja motiva bi bio lakši da su regulatorni motivi potpuno očuvani. Ipak, regulatorni motivi mogu da mutiraju, kao i ostali delovi *DNK*. Pitanje je kako da nađemo regulatorne motive ako ne znamo kako izgledaju. Kroz ovaj rad će biti prikazano nekoliko algoritama za pronalaženje regulatornih motiva.

Fokus ovog rada je na elektronskoj lekciji koja će biti deo šireg elektronskog udžbenika iz bioinformatike. U tekstu su objašnjeni, a u lekciji implementirani svi algoritmi koji su predstavljeni u knjizi *Bioinformatics Algorithms: An Active Learning Approach* autora Filipa Kompoa (*Phillip Compeau*) i Pavela Pevznera (*Pavel Pevzner*). Navedena knjiga je glavni udžbenik kursa Uvod u bioinformatiku. Rezultat rada je *Jupyter* sveska sa implementiranim algoritmima za pronalaženje regulatornih motiva koji su pisani u *Python*-u i koja je javno dostupna na *GitHub*-u [9].

Glava 2

Deterministički algoritmi

Za datu kolekciju nukeotidnih sekvenci, želimo da nađemo podsekvence, po jednu u svakoj sekvenci, koje su međusobno slične. Ovaj biološki problem je poznat kao problem pronalaženja motiva. Ovom problemu se može pristupati na različite načine, pa će kroz rad biti prikazano više njegovih formulacija.

Slika 2.1: Deset nasumično generisanih nukleotidnih sekvenci takvih da je u svaku sekvencu ugrađen isti šablon [5]

Kako bismo simulirali realnu kolekciju sekvenci u kojoj bismo tražili regulatorne motive, generisaćemo na slučajan način 10 nukleotidnih sekvenci dužine 600 i u svaku sekvencu ubaciti šablon AAAAAAAAGGGGGGG dužine 15 na proizvoljnu poziciju. Na Slici 2.1 je prikazan segment ove kolekcije. Umesto da svaka sekvenca sadrži isti ubačeni šablon, on se može mutirati na proizvoljno odabranim pozicijama. Primer mutiranja šablona AAAAAAAAGGGGGGG u svakoj nukelotidnoj sekvenci na 4 proizvoljno odabrane pozicije možemo videti na Slici 2.2. Ova kolekcija je poznata pod nazivom *Subtle Motif Search* kolekcija i u nastavku rada će poslužiti

kao *benchmark* kolekcija za testiranje različitih algoritama za rešavanje problema nalaženja motiva.

Slika 2.2: Deset nasumično generisanih nukelotidnih sekvenci takvih da je u svaku sekvencu ugrađen po jedan mutirani šablon [5]

Na osnovu prethodno opisane kolekcije sekvenci sa ugrađenim šablonom možemo uvesti sledeću formalizaciju. Nek je dat skup sekvenci Dna i ceo broj d. Kažemo da je k-gram s (k, d)- motiv za kolekciju sekvenci Dna ako se s pojavljuje u svakoj sekvenci iz Dna sa najviše d propusta. Kanonski motiv je (k, d)-motiv koji nije mutirao ni na jednoj poziciji. Motivi koji se pojavljuju u niskama sa najviše d propusta u odnosu na kanonski motiv nazivamo instancama kanonskog motiva.

Na primer, na Slici 2.2 za datu kolekciju nukeotidnih sekvenci, AAAAAAA-AGGGGGGG je (k, d)-motiv. Navedeni (k, d)-motiv se pojavljuje kroz instance, na primer AgAAgAAAGGttGGG i cAAtAAAAcGGcGGG, a kanonski motiv se ne pojavljuje ni u jednoj nukleotidnoj sekvenci. Dakle, kanonski motiv se ne mora pojaviti ni u jednoj sekvenci.

2.1 Prva formulacija

Prva formulacija problema pronalaženja motiva je predstavljena Problemom 1.

Problem 1: Problem nalaženja motiva, formulacija 1

Naći sve $(k, d) - motive$ *u skupu niski.*
Ulaz: Skup niski Dna, celi brojevi k (dužina motiva) i d(maksimalan broj razlika).
Izlaz: Svi $(k, d) - motivi$ u Dna.

Najpre ćemo pokušati da dati problem rešimo grubom silom. Pretraga grubom silom je algoritamska tehnika koja ispituje sve moguće kandidate za rešenje nekog problema. Takvi algoritmi garantuju tačno rešenje, ali je za njihovo izvršavanje potrebno dosta vremena, jer broj kandidata može biti veliki.

Rešenje 1

Iako pretraga grubom silom podrazumeva ispitivanje svih mogućih kandidata, očigledno je da nema potrebe ispitivati sve moguće podniske nad azbukom {A,C,G,T}. Svaki (k, d)-motiv se može razlikovati na najviše d pozicija od njegove instance u nekoj od sekvenci skupa *Dna*. Stoga možemo suziti prostor pretrage i generisati sve takve k-grame i proveriti koji od njih su (k, d)-motivi. Ovaj algoritam se naziva algoritam *Motif Enumeration* i predstavljen je Algoritmom 1.

Algoritam 1: MotifEnumeration(Dna,k,d)

```
1 Patterns ← an empty set
2 for each k − mer Pattern in Dna do
3 |   for for each k − mer Pattern′ differing from Pattern by at most d
  |     mismatches do
4 |   |   if Pattern′ appears in each string from Dna with at most d
  |   |     mismatches then
5 |   |   |   add Pattern′ to Patterns
6 |   end
7 end
8 remove duplicates from Patterns
9 return Patterns
```

Ukoliko imamo t sekvenci dužine n, vremenska složenost algoritma je $O(n \cdot 4^k \cdot t)$. Mana ovog algoritma je da je prilično spor za velike k i d, iako smo redukovali broj kandidata i njegovih instanci. Pored toga, neće uvek važiti da svaka sekvenca iz *Dna* sadrži instancu (k, d)-motiva, jer se u skupu sekvenci mogu naći geni koji nisu pod kontrolom odgovarajućih transkripcionih faktora.

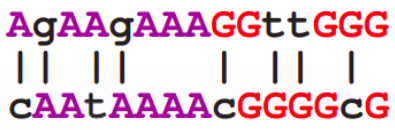
Rešenje 2

Ovaj pristup podrazumeva da unutar kolekcije *Dna* odredimo dva najsličnija k-grama koji pripadaju različitim sekvencama. Postupak je sledeći:

- poredimo parove niski iz kolekcije *Dna*

- uočimo dva najsličnija k-grama u dve niske iz Dna
- napravimo kanonski motiv
- proverimo da li je to (k, d)-motiv.

Dobijene niske se od kanonskog motiva mogu razlikovati na d pozicija, a među sobom na $2 \cdot d$. Na primer, dva k-grama AgAAgAAAGGttGGG i cAAtAAAAcGGGGcG se razlikuju od AAAAAAAAGGGGGGG na 4 pozicije, ali međusobno imaju čak 8 od 15 nepodudaranja (Slika 2.3).



Slika 2.3: Primer dva 15-grama čija je distanca od kanonskog 15-grama jednaka 4, a međusobna distanca jednaka 8[5]

Na *benchmark* kolekciji je ovim pristupom pronađeno nekoliko hiljada parova k-grama koji su se razlikovali na manje od 8 pozicija. Zbog toga ovaj pristup nije dobar izbor za rešavanje problema nalaženja motiva.

2.2 Druga formulacija

U prethodnoj formulaciji ideja je bila da određujemo jednu po jednu instancu motiva u niskama kolekcije Dna. Nova formulacija podrazumeva određivanje cele kolekcije motiva odjednom. Za ovakav pristup je neophodno imati funkciju koja procenjuje koliko su nađeni motivi međusobno slični.

Posmatramo t sekvenci skupa Dna svaki dužine n i slučajnim izborom izaberemo k-gram iz svake sekvence kako bismo formirali kolekciju motiva $Motifs$, koju predstavljamo $t \times k$ matricom motiva. Primer takve matrice se može videti na Slici 2.4. U svakoj koloni velikim slovima ćemo predstavljati najzastupljeniji nukleotid. Ukoliko u koloni postoji više najzastupljenijih nukeotida, onda proizvoljno izaberemo jedan. Variranjem izbora k-grama u svakoj sekvenci, možemo konstruisati veliki broj različitih matrica motiva od sekvenci skupa Dna. Cilj je naći takve k-grame koje će rezultirati matricom sa najviše velikih slova.

Konsenzus niska za kolekciju $Motifs$ je sastavljena od najzastupljenijih nukleotida u svakoj koloni i označava se sa $Consensus(Motifs)$ (Slika 2.4).

Da bismo među više kolekcija motiva izabrali najbolju, neophodno je definisati funkciju pomoću koje bi se mogao oceniti kvalitet kolekcije motiva. Jedan od načina da se ta funkcija zada je kao zbir najmanje zastupljenih nukelotida u svakoj koloni u oznaci $Score(Motifs)$. Na Slici 2.4 vidimo da druga i poslednja kolona u kolekciji motiva $Motifs$ doprinose skoru $Score(Motifs)$ po 4.

Motifs	T	C	G	G	G	G	g	T	T	T	t	t
	c	C	G	G	t	G	A	c	T	T	a	C
	a	C	G	G	G	G	A	T	T	T	t	C
	T	t	G	G	G	G	A	c	T	T	t	t
	a	a	G	G	G	G	A	c	T	T	C	C
	T	t	G	G	G	G	A	c	T	T	C	C
	T	C	G	G	G	G	A	T	T	c	a	t
	T	C	G	G	G	G	A	T	T	c	C	t
	T	a	G	G	G	G	A	a	c	T	a	C
	T	C	G	G	G	t	A	T	a	a	C	C
SCORE(*Motifs*)	3	+ 4	+ 0	+ 0	+ 1	+ 1	+ 1	+ 5	+ 2	+ 3	+ 6	+ 4
CONSENSUS(*Motifs*)	T	C	G	G	G	G	A	T	T	T	C	C

Slika 2.4: Primer kolekcije motiva, konsenzus niske i skora matrice [5]

Nova ideja podrazumeva nalaženje kolekcije $Motifs$ koja minimizuje $Score(Motifs)$, tj. kolekcije u kojoj je broj manje zastupljenih nukleotida što manji i zahteva drugačiju definiciju problema pronalaženja motiva (Problem 2).

Problem 2: Problem pronalaženja motiva, formulacija 2

Za dati skup sekvenci, pronaći skup $k-grama$, po jedan iz svake sekvence sa minimalnim skorom među svim mogućim takvim skupovima

Ulaz: Skup sekvenci Dna, ceo broj k.

Izlaz: Skup $k-grama$ $Motifs$, po jedan iz svake sekvence skupa Dna, tako da je vrednost $Score(Motifs)$ minimalna.

Pokušajmo da rešimo ovaj problem primenom grube sile. Algoritam grube sile za prolanaženja motiva razmatra svaki mogući izbor k-grama $Motifs$ iz Dna (jedan k-gram iz svake sekvence od n nukelotida) i vraća kolekciju $Motifs$ sa minimalnim rezultatom. Možemo izabrati k-gram u svakoj od t sekvenci na $n-k+1$ načina, pa postoji $(n-k+1)^t$ načina da se formira skup motiva $Motifs$. Za svaki izbor $Motifs$,

računa se $Score(Motfis)$ u $k \cdot t$ koraka. Ako pretpostavimo da je k mnogo manje od n, vremenska složenost algoritma je O(n^t · k · t). U *benchmark* kolekciji n = 600, t=10 i k =15, ali u realnim primerima ovi parametri mogu biti i veći. Algoritam je dosta spor, pa je potrebno da reformulišemo problem pronalaženja motiva.

Treća formulacija

U prethodnom rešenju smo ispitivali sve moguće kolekcije motiva i za svaku određivali konsenzus nisku. U novom rešenju ideja je obrnuta, tj. podrazumeva da istražimo sve potencijalne k-grame za konsenzus nisku i onda u datoj kolekciji sekvenci da pronađemo kolekciju motiva koja joj najviše odgovara. Da bismo izračunali koliko jedna niska odgovara kolekciji motiva, potrebna nam je funkcija koja to ocenjuje.

Na Slici 2.5 vidimo da se predloženi $Score(Motifs)$ može računati i red po red. Svaki sabirak u ovom zbiru predstavlja propuste ($mismatches$) između $Consensus(Motifs)$ i k-grama u odgovarajućem redu matrice motiva, tj. Hamingovo rastojanje između ovih niski. Svaka vrednost na kraju reda odgovara Hamingovom rastojanju između tog reda i konsenzus niske. Hamingovo rastojanje između dva k-grama je broj pozicija na kojima se razlikuju.

Motifs	T	C	G	G	G	G	g	T	T	T	t	t	3
	c	C	G	G	t	G	A	c	T	T	a	C	4
	a	C	G	G	G	G	A	T	T	T	t	C	2
	T	t	G	G	G	G	A	c	T	T	t	t	4
	a	a	G	G	G	G	A	c	T	T	C	C	3
	T	t	G	G	G	G	A	c	T	T	C	C	2
	T	C	G	G	G	G	A	T	T	c	a	t	3
	T	C	G	G	G	G	A	T	T	c	C	t	2
	T	a	G	G	G	G	A	a	c	T	a	C	4
	T	C	G	G	G	t	A	T	a	a	C	C	+ 3
SCORE(*Motifs*)	3 +	4 +	0 +	0 +	1 +	1 +	1 +	5 +	2 +	3 +	6 +	4 =	**30**
CONSENSUS(*Motifs*)	T	C	G	G	G	G	A	T	T	T	C	C	

Slika 2.5: Računanje skora matrice red-po-red [5]

Ukoliko imamo niz k-grama $Motifs = \{Motif_1, ..., Motif_t\}$ i k-gram $Pattern$ definišemo $d(Pattern, Motifs)$ kao sumu Hamingovih rastojanja između $Pattern$ i

svakog $Motif_i$,

$$d(Pattern, Motifs) = \sum_{i=1}^{t} HammingDistance(Pattern, Motif_i)$$

Možemo zaključiti da skor matrice motiva ($Score(Motifs)$) odgovara rastojanju konsenzus niske od matrice motiva ($d(Consensus(Motifs), Motifs)$).

U novom pristupu, umesto da tražimo kolekciju k-grama $Motifs$ koja minimizuje $Score(Motifs)$, tražićemo potencijalnu konsenzus nisku $Paterrn$ koja minimizuje $d(Pattern, Motifs)$ među svim mogućim izborima $Pattern$ i svim mogućim izborima skupa k-grama $Motifs$ iz Dna. Problemom 3 je predstavljena nova formulacija problema pronalaženja motiva.

Problem 3: Problem pronalaženja motiva, formulacija 3.

Na osnovu skupa sekvenci Dna, naći Pattern i kolekciju $k-grama$ $Motifs$ po jedan iz svake sekvence takav da je $d(Pattern, Motifs)$ minimalan.

Ulaz: Skup sekvenci Dna, ceo broj k.

Izlaz: $k-gram$ $Pattern$ i skup $k-grama$ $Motifs$ koji minimizuje $d(Pattern, Motifs)$ među svim mogućim izborima $Pattern$ i $Motifs$.

Analizirajmo formulaciju 3 u odnosu na formulaciju 2. U formulaciji 2 minimizujemo $Score(Motifs)$, a u formulaciji 3 $d(Pattern, Motifs)$. Umesto minimizacije funkcije po jednoj promenljivoj($Score(Motifs)$), minimizujemo funkciju po dve promenljive ($d(Pattern, Motifs)$). Ima smisla postaviti pitanje da li je formulacija 3 zapravo teži problem od formulacije 2. U formulaciji 3, promenljive $Pattern$ i $Motifs$ nisu nezavisne i stoga nema potrebe da uzimamo u obzir sve moguće niske $Pattern$ i sve moguće kolekcije $Motifs$ kako bismo minimizovali $d(Pattern, Motifs)$. Umesto toga, za fiksiranu nisku $Pattern$ možemo birati elemente kolekcije $Motifs$ jedan po jedan, iz svake niske, koji će minimizovati $d(Pattern, Motifs)$ za fiksiranu nisku $Pattern$. Da bismo ilustrovali ovu ideju, definišemo $Motifs(Pattern, Dna)$ kao kolekciju k-grama koja minimizuje $d(Pattern, Motifs)$ za dati $Pattern$ i sve moguće kolekcije $Motifs$ iz Dna. Na Slici 2.6 je prikazano 5 obojenih 3-grama koji predstavljaju $Motifs(AAA, Dna)$.

Dna

ttaccttAAC
gATAtctgtc
ACGgcgttcg
ccctAAAgag
cgtcAGAggt

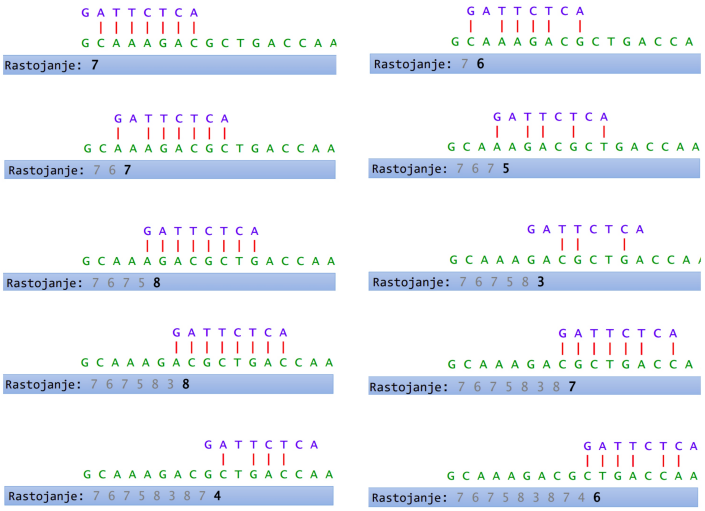
Slika 2.6: Kolekcija sekvenci *Dna* sa označenim 3-gramima koji predstavljaju $Motifs(AAA, Dna)$ [8]

Umesto da za fiksirano *Pattern* ispitujemo sve moguće kolekcije $Motifs$, možemo generisati k-grame po jedan iz svake sekvence koji će minimizovati rastojanje , tj. možemo izabrati k-gram iz svakog reda nezavisno od ostalih redova. Za izbor ovakvog k-grama potrebno je da definišemo Hamingovo rastojanje dve niske različitih dužina.

Hamingovo rastojanje između dve niske različitih dužina u oznaci $d(Pattern, Text)$, predstavlja minimum Hamingovih rastojanja između k-grama $Pattern$ i svih k-grama u dužoj niski $Text$, tj.

$$d(Pattern, Text) = \min_{\forall k-gram Pattern' iz Text} HammingDistance(Pattern, Pattern')$$

Na Slici 2.7 je dat postupak računanja $d(GATTCTCA, GCAAAGACGCTGACCAA)$. Zaključujemo da je $d(GATTCTCA, GCAAAGACGCTGACCAA) = 3$.



Slika 2.7: Postupak računanja $d(GATTCTCA, GCAAAGACGCTGACCAA)$ [8]

Definišemo rastojanje između k-grama $Pattern$ i skupa niski $Dna = \{Dna_1, ..., Dna_t\}$ u oznaci $d(Pattern, Dna)$ kao zbir rastojanja između $Pattern$ i svake niske skupa Dna, tj.

$$d(Pattern, Dna) = \sum_{i=1}^{t} d(Pattern, Dna_i)$$

Za sekvence Dna prikazane na Slici 2.8 važi $d(AAA, Dna) = 1+1+2+0+1 = 5$.

```
ttaccttAAC  1
gATAtctgtc  1
ACGgcgttcg  2
ccctAAAgag  0
cgtcAGAggt  1
```

Slika 2.8: Kolekcija sekvenci Dna sa vrednostima $d(AAA, Dna_i)$ za $i = 1, ..., 5$ [5]

2.3 Problem niske medijane

Kao što smo videli, da bismo minimizovali funkciju $d(Pattern, Motifs)$, dovoljno je da nađemo k-gram $Pattern$ koji minimizuje $d(Pattern, Dna)$ među svim k-gramima $Pattern$. Takav k-gram zovemo niska medijana (Problem 4).

Problem 4: Problem niske medijane

Naći niske medijane
Ulaz: Skup sekvenci Dna i ceo broj k.
Izlaz: k-gram k-mer koji minimuzuje rastojanje $d(k-\text{mer}, Dna)$.

Algoritam *Median String* za rešavanje problema niske medijane je predstavljen Algoritmom 2. Ovaj algoritam izračunava $d(k\text{-mer}, Dna)$ za svaki od 4^k k-grama i kao rezultat će vratiti k-gram koji ima najmanje rastojanje od skupa Dna.

Za izačunavanje rastojanja između k-grama i jedne niske skupa Dna potrebno je $k \cdot (n-k+1)$ koraka. Kako imamo t niski, za izračunavanje rastojanja k-grama od skupa potrebno nam je $t \cdot k \cdot (n-k+1)$, tj. približno $t \cdot k \cdot n$ koraka. Zaključujemo da algoritam *Median String* ima vremensku složenost $O(4^k \cdot k \cdot n \cdot t)$.

Algoritam 2: MedianString(Dna,k)

```
1 distance ← ∞
2 for each k − mer Pattern from AA...AA to TT...TT do
3 |   if distance > d( Pattern,Dna) then
4 |   |   distance ← d(Pattern, Dna )
5 |   |   Median ← Pattern
6 end
7 return Median
```

Možemo uporediti vremensku složenost algoritma grube sile i algoritma *Median String*. Algoritam *Median String* se u praksi pokazao bolje od algoritma grube sile (vremenska složenost $O(n^t \cdot k \cdot t)$), jer dužina motiva k ne prelazi 20, a t se meri u hiljadama. U Tabeli 2.1 prikazano je da je algoritam MedianString testiran na *benchmark* skupu prespor za k=15. Kada je testiran na istom skupu za k=13, kao rešenje je vratio kolekciju motiva $Motifs$ sa konsenzus niskom AAAAAtAGaGGGG, pri čemu je $Score(Motifs)$ koji je izračunat kao suma Hamingovih rastojanja između svakog motiva iz $Motifs$ i konsenzus niske jednak 29.

Tabela 2.1: Rezultati testiranja algoritama na *benchmark* skupu

Algoritam	**k**	**Rešenje**	**Skor**
Median String	13	`AAAAAtAGaGGGG`	29
Median String	15	previše spor	

Traženi (k, d) motiv: `AAAAAAAAGGGGGGG`

Problem koji se javlja pri implementaciji algoritma *Median String* jeste pisanje funkcije $d(Pattern, Dna)$ kao zbir rastojanja između $Pattern$ i svake sekvence u Dna. Ispod je Algoritmom 3 dat pseudokod za računanje $d(Pattern, Dna)$.

Algoritam 3: distanceBetweenPatternAndStrings(*Pattern*,*Dna*)

```
1  k ← |Pattern|
2  distance ← 0
3  for each string Text in Dna do
4  |  HammingDistance ← ∞
5  |  for each k − mer Pattern′ in Text do
6  |  |  if HammingDistance > hammingDistance(Pattern,
   |  |    Pattern′) then
7  |  |  |  HammingDistance ← hammingDistance(Pattern,
   |  |  |    Pattern′)
8  |  end
9  |  distance ← distance + HammingDistance
10 end
11 return distance
```

Glava 3

Probabilistički algoritmi

Deterministički algoritmi daju uvek isto rešenje za isti ulaz, dok kod probabilističkih to ne mora biti slučaj zato što oni imaju bar jedan korak koji predstavlja neki slučajan izbor (*randomness*). Stoga zbog postizanja najbolje tačnosti, ove algoritme treba pokrenuti više puta i uzeti najbolji rezulat.

Da bismo analizirali neke od probabilističkih rešenja problema nalaženja motiva, neophodno nam je da definišemo matrice $Count(Motifs)$ i $Profile(Motifs)$ za skup motiva $Motifs$. $Count(Motifs)$ je matrica dimenzije $4 \times k$ koja predstavlja broj pojavljivanja svakog nukleotida u svakoj koloni matrice motiva. Dakle, element matrice $Count(Motifs)$ na poziciji (i, j) predstavlja koliko se puta nukleotid i pojavljuje u koloni j matrice $Motifs$. Slika 3.1 predstavlja $Count(Motifs)$ za kolekciju motiva $Motifs$ sa Slike 2.4. Ovo rezultira profilnom matricom u oznaci $Profile(Motifs)$, gde element ove matrice na poziciji (i, j) predstavlja frekvenciju i-og nukleotida u koloni j matrice $Motifs$. Zbir vrednosti bilo koje kolone ove matrice mora biti 1.

A:	2	2	0	0	0	0	9	1	1	1	3	0
C:	1	6	0	0	0	0	0	4	1	2	4	6
G:	0	0	10	10	9	9	1	0	0	0	0	0
T:	7	2	0	0	1	1	0	5	8	7	3	4

Slika 3.1: Primer $Count(Motifs)$ [5]

Svaka kolona $Profile(Motifs)$ odgovara distribuciji verovatnoće tj. koloni nenegativnih brojeva čiji je zbir 1. Na Slici 3.2 je dat primer profilne matrice gde vidimo da druga kolona odgovara vrednostima verovatnoća 0.2, 0.6, 0.0 i 0.2, za A, C, G i T.

A:	.2	.2	0	0	0	0	.9	.1	.1	.1	.3	0
C:	.1	.6	0	0	0	0	0	.4	.1	.2	.4	.6
G:	0	0	1	1	.9	.9	.1	0	0	0	0	0
T:	.7	.2	0	0	.1	.1	0	.5	.8	.7	.3	.4

Slika 3.2: Primer $Profile(Motifs)$ [5]

Neka je data kolekcija k-grama $Motifs$ izabranih iz t sekvenci Dna. Profilnu matricu sa k kolona možemo posmatrati kao kolekciju od k kockica, koju ćemo baciti nasumično da bismo generisali k-gram. Na primer, ako je prva kolona profilne matrice (0.2, 0.1, 0.0, 0.7) onda generišemo A kao prvi nukleotid sa verovatnoćom 0.2, C sa verovatnoćom 0.1, G sa verovatnoćom 0.0 i T sa verovatnoćom 0.7.

$Pr(a|P)$ definišemo kao uslovnu verovatnoću generisanja k-grama a za datu profilnu matricu P. Ako je k-gram a sličan konsenzus niski onda je $Pr(a|P)$ visok. Generisanje pojedinačnih nukleotida je međusobno nezavisno. Ukoliko sa $P_{a_i,i}$ označimo verovatnoću slova a_i na poziciji i, onda se verovatnoća $Pr(a|P)$ dobija jednostavnim množenjem pojedinačnih verovatnoća:

$$Pr(a|P) = \prod_{i=1}^{k} P_{a_i,i}.$$

Neka je Tabelom 3.1 predstavljena profilna matrica Profile.

Tabela 3.1: Profile

A	1/2	7/8	3/8	0	1/8	0
C	1/8	0	1/2	5/8	3/8	0
T	1/8	1/8	0	0	1/4	7/8
G	1/4	0	1/8	3/8	1/4	1/8

Koristeći profilnu matricu dobijamo da je $Pr(ATACAG|Profile) = \frac{1}{2} \cdot \frac{1}{8} \cdot \frac{3}{8} \cdot \frac{5}{8} \cdot \frac{1}{8} \cdot \frac{1}{8} = 0.001602$.

Kada je data profilna matrica, možemo proceniti verovatnoću svakog k-grama u sekvenci i naći najverovatniji k-gram u sekvenci. Ispod je Problemom 5 predstavljen problem nalaženja najverovatnijeg k-grama u niski $Text$.

Problem 5: Problem najverovatnijeg k-grama

Naći najverovatniji k-gram u niski.
Ulaz: Data je niska $Text$, ceo broj k i profilna matrica $4 \times k$.
Izlaz: Najverovatniji k-gram u $Text$.

Na osnovu profilne matrice koja je predstavljena Tabelom 3.1, računamo najverovatniji 6-gram u niski *ctataaaccttacat* (Slika 3.3). Postupak je sledeći:

- uzimamo podniske dužine 6
- za svaku podnisku a i datu profilnu matricu P računamo $Pr(a|P)$
- na osnovu izračunatih verovatnoća pronalazimo najverovatniji 6-gram.

U navedenom primeru, najverovatniji 6-gram je *aaacct*.

Window, Highlighted Red	Calculations	Pr(a|P)
ctataaaccttacat	1/8 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 7/8 x 0 x 0 x 1/8 x 0	0
ctataaaccttacat	1/2 x 1/8 x 3/8 x 0 x 1/8 x 0	0
ctataaaccttacat	1/8 x 7/8 x 3/8 x 0 x 3/8 x 0	0
ctataaaccttacat	1/2 x 7/8 x 3/8 x 5/8 x 3/8 x 7/8	.0336
ctataaaccttacat	1/2 x 7/8 x 1/2 x 5/8 x 1/4 x 7/8	.0299
ctataaaccttacat	1/2 x 0 x 1/2 x 0 1/4 x 0	0
ctataaaccttacat	1/8 x 0 x 0 x 0 x 0 x 1/8 x 0	0
ctataaaccttacat	1/8 x 1/8 x 0 x 0 x 3/8 x 0	0
ctataaaccttacat	1/8 x 1/8 x 3/8 x 5/8 x 1/8 x 7/8	.0004

Slika 3.3: Izračunavanje najverovatnijeg 6-grama [4]

3.1 Pohlepna pretraga motiva

Pohlepni algoritmi predstavljaju pristup rešavanju optimizacionih problema gde se u svakom koraku bira najbolja opcija koja je dostupna u tom trenutku. Pohlepni algoritmi tako biraju lokalno optimalno rešenje u svakom koraku u nadi da će ih ti izbori dovesti do globalnog optimuma. Ova strategija ne mora uvek dovoditi do optimalnog rešenja. Na primer, ukoliko prilikom igranja šaha gledamo samo jedan potez unapred, moguće je da će to dovesti do lošeg rezultata. Ipak, ovi algoritmi su jednostavni i često imaju malu vremensku složenost, što ih čini efikasnim za

određene vrste problema. Međutim, nekad se može desiti da ne pronađu globalno optimalno rešenje.

U nastavku ćemo analizirati jedan od probabilističkih algoritama za rešavanje problema nalaženja motiva, *Greedy Motif Search*, koji je predstavljen Algoritmom 4. Ideja algoritma je da postavlja svaki od k-grama iz Dna_1 (prva sekvenca iz skupa sekvenci $Dna = \{Dna_1, ..., Dna_t\}$) kao prvi motiv $Motif_1$ potencijalno traženog skupa motiva $Motifs$. Za dati izbor k-grama iz Dna_1 pravi profilnu matricu i postavlja za $Motif_2$ najverovatniji k-gram iz Dna_2. Zatim se ažurira profilna matrica na osnovu $Motif_1$ i $Motif_2$ i postavlja se za $Motif_3$ najverovatniji k-gram iz Dna_3. Nakon pronalaženja $i-1$ k-grama $Motifs$ u prvim $i-1$ sekvencama iz Dna, ovaj algoritam konstruiše profilnu matricu i bira najverovatniji k-gram iz Dna_i na osnovu profilne matrice. Kada dobijemo skup k-grama $Motifs$ proverava se da li taj skup nadmašuje trenutnu kolekciju motiva sa najboljim rezultatom, a zatim se bira naredni $Motif_1$ i ponavlja se proces generisanja $Motifs$.

Algoritam 4: GreedyMotifSearch(Dna, k, t)

```
1  BestMotifs ← motif matrix formed by first k − mers in each string
     from Dna
2  for each k − mer Motif in the first string from Dna do
3  |   Motif_1 ← Motif
4  |   for i=2 to t do
5  |   |   form Profile from motifs Motif_1, ..., Motif_{i−1}
6  |   |   Motif_i ← Profile-most probable k − mer in the i − th string in
   |   |     Dna
7  |   end
8  |   Motifs ← {Motif_1, ..., Motif_t}
9  |   if Score(Motifs) < Score(BestMotifs) then
10 |   |   BestMotifs ← Motifs
11 end
12 return BestMotifs
```

ttACCTtaac
gATGTctgtc
acgGCGTtag
ccctaACGAg
cgtcagAGGT

Slika 3.4: Kolekcija sekvenci Dna sa označenim instacama $(4, 1)$-motiva $ACGT$ [5]

Želimo da pomoću ovog algoritma nađemo $(4, 1)$-motiv $ACGT$ u kolekciji sekvenci Dna sa Slike 3.4, pri čemu su velikim slovima u svakoj sekvenci predstavljene instance $(4, 1)$-motiva $ACGT$, plavim slovima su predstavljeni nukleotidi koji se poklapaju sa motivom, a crvenim oni koji se ne poklapaju sa motivom $ACGT$. Pretpostavimo da je algoritam izabrao iz prve sekvence $ACCT$ i konstruišemo profilnu matricu (Slika 3.5).

A:	**1**	0	0	0
C:	0	**1**	**1**	0
G:	0	0	0	0
T:	0	0	0	**1**

Slika 3.5: Profilna matrica [5]

Sada algoritam treba da nađe najverovatniji k-gram u drugoj sekvenci. Problem predstavlja to što u profilnoj matrici postoje nule, tako da je verovatnoća svakog 4-grama 0, osim $ACCT$. Dakle, ako $ACCT$ nije prisutan u svakoj sekvenci, mala je šansa da će ovaj algoritam vratiti rešenje. U nastavku ćemo pokazati kako se Laplasovo pravilo sukcesije može iskoristiti u prevazilaženju ovog problema.

Razmotrimo performanse algoritma $Greedy\ Motif$. U odnosu na algoritam $Median\ String$, ovaj algoritam je brži. Za razliku od algoritma $Median\ String$, algoritam $GreedyMotif$ uspeva da pronađe rešenje za $k = 15$. U Tabeli 3.2 je prikazan rezultat algoritma $Greedy\ Motif$ testiran na *benchmark* skupu. Kao rezultat vraća kolekciju motiva $Motifs$ sa konsenzus niskom gtAAAtAgaGatGtG, pri čemu je $Score(Motifs) = 29$, što znači da daje lošije rešenje od algoritma $Median\ String$.

Tabela 3.2: Rezultati testiranja algoritama na *benchmark* skupu

Algoritam	**k**	**Rešenje**	**Skor**
$Median\ String$	13	AAAAAtAGaGGGG	29
$Median\ String$	15	previše spor	
$Greedy\ Motif$	15	gtAAAtAgaGatGtG	58
Traženi (k, d) motiv:		AAAAAAAAGGGGGGG	

3.2 Poboljšana pohlepna pretraga motiva

Na Slici 3.6 je prikazana uslovna verovatnoća generisanja $TCGTGGATTTCC$ za datu profilnu matricu $Profile$. Možemo videti da četvrti simbol u $TCGTGGATTTCC$ dovodi da $Pr(TCGTGGATTTCC|Profile)$ bude jednak 0. Niska $TCGTGGATTTCC$ se razlikuje od konsenzus niske $TCGGGGATTTCC$ na samo jednoj poziciji. $TCGTGGATTCC$ ima istu malu verovatnoću kao $AAATCTTGGAA$, koja se veoma razlikuje od konsenzus niske.

Profile	A:	.2	.2	.0	.0	.0	.0	.9	.1	.1	.1	.3	.0
	C:	.1	.6	.0	.0	.0	.0	.0	.4	.1	.2	.4	.6
	G:	.0	.0	1	1	.9	.9	.1	.0	.0	.0	.0	.0
	T:	.7	.2	.0	.0	.1	.1	.0	.5	.8	.7	.3	.4

$$\Pr(\texttt{TCGTGGATTTCC}|Profile) = .7 \cdot .6 \cdot 1 \cdot .0 \cdot .9 \cdot .9 \cdot .9 \cdot .5 \cdot .8 \cdot .7 \cdot .4 \cdot .6 = 0$$

Slika 3.6: Pr(TCGTGGATTTCC | Profile) [5]

Jedan od načina da se ovaj fenomen bolje modeluje jeste da se nule zamene malim vrednostima koje se nazivaju pseudovrednosti. Postupak uvođenja pseudovrednosti naziva se Laplasovo pravilo sukcesije. Naime, ukoliko ponovimo neki eksperiment n puta i dobijemo s puta uspeh, potrebno je izračunati verovatnoću da sledeći eksperiment ima uspešan ishod. Uvodimo oznake X_1, ..., X_{n+1} (uslovno nezavisne logičke promenjive) pri čemu je X_i jednak 1 ako je i-ti eksperiment imao uspešan ishod ili 0 ako je neuspešan. Tada važi:

$$Pr(X_{n+1} = 1|X_1 + \ldots + X_n = s) = \frac{s}{n}.$$

Kako znamo da su u sledećem eksperimentu dva moguća ishoda (tačno/netačno), onda smo obavili $n + 2$ eksperimenta sa $s + 1$ uspešnim ishodom. Dodate vrednost predstavljaju pseudovrednosti i važi:

$$Pr(X_{n+1} = 1|X_1 + \ldots + X_n = s) = \frac{s+1}{n+2}.$$

Da bismo primenili Laplasovo pravilo sukcesije na problem nalaženja motiva, posmatrajmo kolekciju motiva $Motifs$ i nad njom generisane matrice $Count(Motifs)$ i $Profile(Motifs)$ (Slika 3.7).

Motifs

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

COUNT(*Motifs*)

```
A: 2 1 1 1
C: 0 1 1 1
G: 1 1 1 0
T: 1 1 1 2
```

PROFILE(*Motifs*)

```
2/4 1/4 1/4 1/4
 0  1/4 1/4 1/4
1/4 1/4 1/4  0
1/4 1/4 1/4 2/4
```

Slika 3.7: Matrice bez primene Laplasovog pravila [5]

Kako Laplasovo pravilo sukcesije dodaje svakom elementu matrice $Count(Motifs)$ 1, na Slici 3.8 se mogu videti ažurirane $Count(Motifs)$ i $Profile(Motifs)$.

COUNT(*Motifs*)

```
A: 2+1 1+1 1+1 1+1
C: 0+1 1+1 1+1 1+1
G: 1+1 1+1 1+1 0+1
T: 1+1 1+1 1+1 2+1
```

PROFILE(*Motifs*)

```
3/8 2/8 2/8 2/8
1/8 2/8 2/8 2/8
2/8 2/8 2/8 1/8
2/8 2/8 2/8 3/8
```

Slika 3.8: Matrice nakon primene Laplasovog pravila [5]

Jedina promena koju treba da uvedemo algoritmu *Greedy Motif Search* (Algoritam 4) je da zamenimo liniju 5. Modifikacija je predstavljena Algoritmom 5.

Algoritam 5: GreedyMotifSearchWithPseudocounts(Dna, k, t)

```
1  BestMotifs ← motif matrix formed by first k − mers in each string
     from Dna
2  for each k − mer Motif in the first string from Dna do
3  |  Motif_1 ← Motif
4  |  for i=2 to t do
5  |  |  apply Laplace's Rule of Succession to form Profile from motifs
   |  |    Motif_1, ..., Motif_{i−1}
6  |  |  Motif_i ← Profile-most probable k − mer in the i − th string in
   |  |    Dna
7  |  end
8  |  Motifs ← {Motif_1, ..., Motif_t}
9  |  if Score(Motifs) < Score(BestMotifs) then
10 |  |  BestMotifs ← Motifs
11 end
12 return BestMotifs
```

Ilustrovaćemo primenu poboljšanog pohlepnog algoritma na primeru pronalaženja motiva unutar kolekcije sekvenci *Dna* (Slika 3.4).

Skup motiva gradimo dodavajući jedan po jedan motiv i pretpostavimo da je algoritam izabrao iz prve niske ACCT. Sada konstruišemo $Count(Motifs)$ i $Profile(Motifs)$ koristeći Laplasovo pravilo sukcesije (Slika 3.9).

Motifs **ACCT**

COUNT(*Motifs*)				
A:	1+1	0+1	0+1	0+1
C:	0+1	1+1	1+1	0+1
G:	0+1	0+1	0+1	0+1
T:	0+1	0+1	0+1	1+1

PROFILE(*Motifs*)			
2/5	1/5	1/5	1/5
1/5	2/5	2/5	1/5
1/5	1/5	1/5	1/5
1/5	1/5	1/5	2/5

Slika 3.9: Prvi korak algoritma [5]

Koristimo profilnu matricu da izračunamo verovatnoće svih 4-grama iz druge niske.

gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
$1/5^4$	$4/5^4$	$1/5^4$	$4/5^4$	$2/5^4$	$2/5^4$	$1/5^4$

Slika 3.10: Verovatnoće svih 4-grama iz druge niske [5]

Sa Slike 3.10 vidimo da su dva najverovatnija 4-grama iz druge niske ATGT i GTct. Izaberimo ATGT. Na Slici 3.11 su data dva motiva $Motifs$, $Count(Motifs)$ i $Profile(Motifs)$.

Motifs **ACCT**
ATGT

COUNT(*Motifs*)				
A:	2+1	0+1	0+1	0+1
C:	0+1	1+1	1+1	0+1
G:	0+1	0+1	1+1	0+1
T:	0+1	1+1	0+1	2+1

PROFILE(*Motifs*)			
3/6	1/6	1/6	1/6
1/6	2/6	2/6	1/6
1/6	1/6	2/6	1/6
1/6	2/6	1/6	3/6

Slika 3.11: Drugi korak algoritma [5]

Koristimo dobijenu profilnu matricu da izračunamo verovatnoće svih 4-grama iz treće niske (Slika 3.12).

acgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$12/6^4$	$2/6^4$	$2/6^4$	$12/6^4$	$3/6^4$	$2/6^4$	$2/6^4$

Slika 3.12: Verovatnoće svih 4-grama iz treće niske [5]

Vidimo na Slici 3.12 da su dva najverovatnija 4-grama iz treće niske acgG i GCGT. Biramo `acgG`. Skupu motiva $Motifs$ dodamo `acgG` i ažuriramo $Count(Motifs)$ i $Profile(Motifs)$ (Slika 3.13).

Motifs

ACCT
ATGT
acgG

COUNT(*Motifs*)

A:	3+1	0+1	0+1	1+1
C:	0+1	2+1	1+1	0+1
G:	0+1	0+1	2+1	1+1
T:	0+1	1+1	0+1	2+1

PROFILE(*Motifs*)

4/7	1/7	1/7	1/7
1/7	3/7	2/7	1/7
1/7	1/7	3/7	2/7
1/7	2/7	1/7	3/7

Slika 3.13: Treći korak algoritma [5]

Koristimo profilnu matricu da izračunamo verovatnoće svih 4-grama iz četvrte niske (Slika 3.14).

ccct	ccta	ctaA	taAC	aACG	ACGA	CGAg
$18/7^4$	$3/7^4$	$2/7^4$	$1/7^4$	$16/7^4$	$36/7^4$	$2/7^4$

Slika 3.14: Verovatnoće svih 4-grama iz četvrte niske [5]

Najverovatniji 4-gram u četvrtoj niski je $ACGA$. Dodajemo u $Motifs$ motiv `ACGA` i ažuriramo $Count(Motifs)$ i $Profile(Motifs)$ (Slika 3.15).

Motifs

ACCT
ATGT
acgG
ACGA

COUNT(*Motifs*)

A:	4+1	0+1	0+1	0+1
C:	0+1	3+1	1+1	0+1
G:	0+1	0+1	3+1	1+1
T:	0+1	1+1	0+1	2+1

PROFILE(*Motifs*)

5/8	1/8	1/8	2/8
1/8	4/8	2/8	1/8
1/8	1/8	4/8	2/8
1/8	2/8	1/8	3/8

Slika 3.15: Četvrti korak algoritma [5]

Najzad, računamo verovatnoće svih 4-grama iz pete niske (Slika 3.16).

cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
$1/8^4$	$8/8^4$	$8/8^4$	$8/8^4$	$10/8^4$	$8/8^4$	$60/8^4$

Slika 3.16: Verovatnoće svih 4-grama pete niske Dna [5]

Najverovatniji 4-gram u petoj niski je AGGT. Algoritam *Greedy Motif Search* je proizveo matricu motiva $Motifs$, koja podrazumeva ispravan $Consensus(Motifs)$ ACGT (Slika 3.17).

Motifs

ACCT
ATGT
acgG
ACGA
AGGT

CONSENSUS(*Motifs*) ACGT

Slika 3.17: Skup motiva $Motifs$ koji je vratio algoritam i $Consensus(Motifs)$ [5]

Algoritam *Median String* je prespor za $k > 13$, a algoritam *Greedy Motif* je brži i može da nađe motive dužine $k = 15$. Veća brzina donosi manju tačnost. U Tabeli 3.3 je prikazano da je algoritam *Greedy Motif* primenom Laplasovog pravila testiran na *benchmark* kolekciji vratio kolekciju motiva $Motifs$ sa konsenzus niskom AAAAAtAgaGGGGtt, pri čemu je $Score(Motifs) = 41$.

Tabela 3.3: Rezultati testiranja algoritama na *benchmark* skupu

Algoritam	**k**	**Rešenje**	**Skor**
Median String	13	AAAAAtAGaGGGG	29
Median String	15	previše spor	
Greedy Motif	15	gtAAAtAgaGatGtG	58
Greedy Motif Laplace	15	AAAAAtAgaGGGGtt	41
Traženi (k, d) motiv:		AAAAAAAAGGGGGGG	

3.3 Probabilistička pretraga motiva

Prethodno opisani algoritam *Greedy Motif Search* uvek kreće od istog skupa motiva. Ideja je da umesto kretanja od početnih pozicija, krećemo od nekih nasumično izabranih. Kako ne bismo iterirali kroz niske kao u prethodnom algoritmu, možemo pokušati da pravimo profilnu matricu od skupa motiva i obrnuto i da te korake ponavljamo sve dok se skor motiva smanjuje. U nastavku će biti prikazan algoritam *Randomized Motif Search* čija je ideja da se ažuriranjem profilne matrice i matrice motiva dobija što verovatniji skup motiva.

Pre nego što analiziramo pomenuti algoritam, potrebno je dati sledeću formalizaciju. Neka je dat skup sekvenci Dna i profilna matrica $Profile$ dimenzije $4 \times k$. Definišemo $Motifs(Profile, Dna)$ kao kolekciju k-grama formiranu od najverovatnijih k-grama iz svake sekvence skupa Dna.

Možemo početi sa nasumično izabranim k-gramima motiva iz kolekcije sekvenci Dna, konstruisati matricu $Profile(Motifs)$, a zatim iskoristiti tu matricu da bi se konsturisala nova kolekcija k-grama $Motifs$:

$$Motifs(Profile(Motifs), Dna).$$

Dobijamo $Motifs$, a zatim konstruišemo profilnu matricu:

$$Profile(Motif(Profile(Motifs), Dna))$$

koju koristimo da bismo dobili najverovatnije k-grame:

$$Motifs(Profile(Motifs(Profile(Motifs), Dna)), Dna).$$

Nastavljamo dalju iteraciju:

$$...Profile(Motifs(Profile(Motifs(Profile(Motifs), Dna)), Dna))...$$

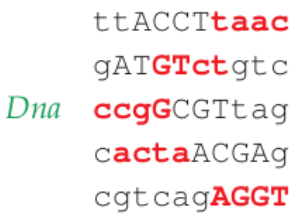
sve dok se skor poboljšava.

Ovaj algoritam počinjemo sa nasumično izabranim k-gramima pa je potreban generator slučajnih brojeva ($Random(N)$) koji jednako verovatno može vratiti bilo koji ceo broj između 1 i N. Ispod je Algoritmom 6 dat pseudokod algoritma *Randomized Motif Search*.

Algoritam 6: Randomized Motif Search(Dna, k, t))

```
randomly select k − mers Motifs = (Motif_1,...,Motif_t) in each string
  from Dna
BestMotifs ← Motifs
while forever do
    Profile ← Profile(Motifs)
    Motifs ← Motifs(Profile, Dna)
    if Score(Motifs) < Score(BestMotifs) then
        BestMotifs ← Motifs
    else
        return BestMotifs
end
```

Ilustrovaćemo primenu *Randomized Motif Search* algoritma na primeru pronalaženja motiva unutar kolekcije sekvenci Dna. Na Slici 3.18 su crvenom bojom označeni nasumično izabrani k-grami iz svake sekvence, a velikim slovima su predstavljene instance $(4, 1)$-motiva $ACGT$.



Slika 3.18: Nasumično izabrani k-grami iz svake sekvence skupa Dna [5]

Na osnovu k-grama $Motifs$ pravimo $Profile(Motifs)$ (Slika 3.19).

Motifs			
t	a	a	c
G	T	c	t
c	c	g	G
a	c	t	a
A	G	G	T

PROFILE(*Motifs*)				
A:	0.4	0.2	0.2	0.2
C:	0.2	0.4	0.2	0.2
G:	0.2	0.2	0.4	0.2
T:	0.2	0.2	0.2	0.4

Slika 3.19: Matrica $Profile(Motifs)$ dobijena na osnovu nasumično izabranih k-grama [5]

Sa Slike 3.20 vidimo da na osnovu profilne matrice možemo izračunati verovatnoću svakog k-grama u kolekciji sekvenci Dna. Iz svakog reda biramo najverovatniji k-gram i formiramo skup k-grama $Motifs$ i nastavljamo iteraciju.

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
.0016	.0016	**.0128**	.0064	.0016	.0016	.0016
gATG	ATGT	TGTc	GTct	Tctg	ctgt	tgtc
.0016	**.0128**	.0016	.0032	.0032	.0032	.0016
ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
.0064	.0036	.0016	**.0128**	.0032	.0016	.0016
cact	acta	ctaA	taAC	aACG	ACGA	CGAg
.0032	.0064	.0016	.0016	.0032	**.0128**	.0016
cgtc	gtca	tcag	cagA	agAG	gAGG	AGGT
.0016	.0016	.0016	.0032	.0032	.0032	**.0128**

Slika 3.20: Verovatnoće svih k-grama iz Dna [5]

Kako jedno pokretanje algoritma nije dovoljno za dobijanje najboljeg rešenja, u praksi se ovaj algoritam pokreće hiljadama puta i kao rešenje bira se skup motiva sa najnižim skorom. U Tabeli 3.4 je prikazan rezultat ovog algoritma testiran na *benchmark* kolekciji nakon 100 000 pokretanja (svaki put sa novim izabranim k-gramima). Rezultat je kolekcija motiva $Motifs$ sa konsenzus niskom AAAAAA-AAacaGGGG, pri čemu je $Score(Motifs) = 43$, što znači da daje lošije rešenje od algoritma *Greedy Motif Laplace* i algoritma *Median String*, ali bolje rešenje od algoritma *Greedy Motif*.

Tabela 3.4: Rezultati testiranja algoritama na *benchmark* skupu

Algoritam	**k**	**Rešenje**	**Skor**
Median String	13	AAAAAtAGaGGGG	29
Median String	15	previše spor	
Greedy Motif	15	gtAAAtAgaGatGtG	58
Greedy Motif Laplace	15	AAAAAtAgaGGGGtt	41
Randomized Motif	15	AAAAAAAAacaGGGG	43
Traženi (k, d) motiv:		AAAAAAAAGGGGGGG	

Ima smisla postaviti pitanje kako slučajno odabrani motivi mogu da nas dovedu do ovakvog rešenja. Verovatnoća da u kolekciji *Subtle Motif Search* slučajnim izborom u jednoj sekvenci nađemo ubačeni motiv je $\frac{1}{(600-15+1)} = \frac{1}{586}$, a verovatnoća da ćemo promašiti je $1 - \frac{1}{586}$. Dakle, verovatnoća u celoj kolekciji je $(1 - \frac{1}{586})^{10} \sim 0.983$, što znači da ukoliko algoritam pokrenemo 1000 puta, 983 puta ćemo promašiti i 17 puta pogoditi bar neki motiv. Ako ne pogodimo motiv, pogodićemo deo niske generisan na slučajan način gde nam je svaki nukleotid generisan sa jednakom verovatnoćom na svakoj poziciji (0.25) i možemo očekivati uniformnu profilnu matricu (sve vrednosti 0.25) koja algoritam ne usmerava prema rešenju. Ukoliko pogodimo bar jedan motiv, vrednosti u matrici neće biti sasvim slučajne i matrica će usmeravati pretragu.

3.4 Gibsovo uzorkovanje

Prethodno opisani algoritam *Randomized Motif Search* podrazumeva promenu svih motiva iz $Motifs$ u jednoj iteraciji i tad ispravni motivi mogu biti odbačeni. Iz tog razloga modifikujemo algoritam tako da u svakoj iteraciji bira jedan k-gram iz trenutnog skupa motiva $Motifs$, izbacuje ga, pravi profilnu matricu od ostatka motiva, a zatim ažurira matricu motiva samo nad izbačenim redom u matrici. Ova modifikacija algoritma *Randomized Motif Search* se naziva Gibsovo uzorkovanje (*Gibbs Motif Search*).

U ovom algoritmu rezultat svakog koraka zavisi samo od rezultata prethodnog i način odabira sledećeg koraka nije deterministički, već zasnovan na slučajnom uzorku, tj. koristi nasumično izabrane k-grame $Motifs$=($Motif_1$,...,$Motif_t$) da izabere bolji skup motiva, dok *Randomized Motif Search* deterministički bira nove motive

koristeći

$$Motifs(Profile(Motifs), Dna).$$

Gibsovo uzorkovanje bira nasumično jedan broj i između 1 i t, pri čemu je t broj sekvenci iz skupa Dna i menja k-gram iz $Motif_i$. Za opis rada ovog algoritma potreban je napredniji generator slučajnih brojeva. $Random(p_1, ..., p_n)$ je defnisan za bilo koji skup nenegativnih brojeva koji nužno zadovoljava uslov $\sum_{i=1}^{n} p_i = 1$. Ukoliko je $\sum_{i=1}^{n} p_i = C > 0$, onda $Random(p_1, ..., p_n) = Random(p_1/C, ..., p_n/C)$ pri čemu je $(p_1/C, ..., p_n/C)$ raspodela verovatnoće.

Potrebno je definisati šta predstavlja nasumično generisan k-gram u niski $Text$. Za svaki k-gram $Pattern$ u niski $Text$ računamo $Pr(Pattern|Text)$, što daje $n = |Text| - k + 1$ verovatnoća $(p_1, ...p_n)$. Možemo formirati generator slučajnih brojeva $Random(p_1, ..., p_n)$ zasnovan na njima. Gibsovo uzorkovanje koristi ovaj generator slučajnih brojeva da izabere k-gram koji je nasumično generisan. Ova procedura se ponavlja N puta, ali u praksi zavisi od raznih pravila zaustavljanja.

Ispod je Algoritmom 7 dat pseudokod algoritma *Gibbs Motif Search*.

Algoritam 7: *Gibbs Motif Search*(Dna,k, t, N)

```
1  randomly select k − mers Motifs = (Motif_1, ..., Motif_t) in each string
     from Dna
2  BestMotifs ← Motifs
3  for i ← to N do
4  |  i ← Random(t)
5  |  Profile ← profile matrix formed from all strings in Motifs except
   |    for Motif_i
6  |  Motif_i ← Profile-randomly generated k − mer in the i − th
   |    sequence
7  |  if Score(Motifs) < Score(BestMotifs) then
8  |  |  BestMotifs ← Motifs
9  end
10 return BestMotifs
```

Ilustrovaćemo primenu *Gibbs Motifs Search* algoritma na primeru pronalaženja motiva unutar kolekcije sekvenci Dna (Slika 3.21). U početnom koraku iz svake sekvence su nasumično izabrani k-grami i nasumično je izabrana treća sekvenca za uklanjanje.

Slika 3.21: Izbacivanje treće sekvence iz skupa sekvenci *Dna* [5]

Uklanjanjem treće sekvence, dobijamo novi skup motiva $Motifs$, a zatim računamo $Count(Motifs)$ i $Profile(Motifs)$ bez primene Laplasovog pravila (Slika 3.22).

Motifs

```
t a a c
G T c t
a c t a
A G G T
```

COUNT(*Motifs*)

```
A: 2 1 1 1
C: 0 1 1 1
G: 1 1 1 0
T: 1 1 1 2
```

PROFILE(*Motifs*)

```
A: 2/4 1/4 1/4 1/4
C:  0  1/4 1/4 1/4
G: 1/4 1/4 1/4  0
T: 1/4 1/4 1/4 2/4
```

Slika 3.22: Matrice bez primene Laplasovog pravila [5]

Sada koristeći $Profile(Motifs)$ računamo verovatnoće svih 4-grama iz niske *ccgGCGTtag*. Na Slici 3.23 se uočava da su samo dve verovatnoće različite od nule. Susrećemo se sa problemom koji se javio i kod *Greedy Motif Search*. Problem ćemo rešiti primenom Laplasovog pravila.

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
0	0	0	1/128	0	1/256	0

Slika 3.23: Verovatnoće svih k-grama iz treće sekvence[5]

Kako Laplasovo pravilo sukcesije dodaje svakom elementu matrice Count(Motifs) 1, na Slici 3.24 se mogu videti ažurirane Count(Motifs) i Profile(Motifs).

COUNT(*Motifs*)

A:	3	2	2	2
C:	1	2	2	2
G:	2	2	2	1
T:	2	2	2	3

PROFILE(*Motifs*)

A:	3/8	2/8	2/8	2/8
C:	1/8	2/8	2/8	2/8
G:	2/8	2/8	2/8	1/8
T:	2/8	2/8	2/8	3/8

Slika 3.24: Matrice nakon primene Laplasovog pravila[5]

Nakon ažuriranja ove dve matrice, ponovo računamo verovatnoće svih 4-grama iz niske *ccgGCGTtag* (Slika 3.25).

ccgG	cgGC	gGCG	GCGT	CGTt	GTta	Ttag
$4/8^4$	$8/8^4$	$8/8^4$	$24/8^4$	$12/8^4$	$16/8^4$	$8/8^4$

Slika 3.25: Verovatnoće svih k-grama iz treće sekvence[5]

Sekvencu *ccgGCGTtag* vraćamo u kolekciju sekvenci *Dna*, ali ugrađeni motiv *ccgG* iz treće sekvence skupa *Dna* zamenjujemo sa *GCGT*. Sada biramo prvu sekvencu iz skupa *Dna* za uklanjanje i ponavljamo postupak (Slika 3.26).

	ttACCT**taac**		----------
	gAT**GTct**gtc		gAT**GTct**gtc
Dna	ccg**GCGT**tag	⟶	ccg**GCGT**tag
	c**acta**ACGAg		c**acta**ACGAg
	cgtcag**AGGT**		cgtcag**AGGT**

Slika 3.26: Uklanjanje prve sekvence iz skupa *Dna* [5]

Na Slici 3.27 su dat skup motiva $Motifs$ i $Profile(Motifs)$ bez primene Laplasovog pravila.

Motifs

G	T	c	t
G	C	G	T
a	c	t	a
A	G	G	T

PROFILE(*Motifs*)

A:	2/4	0	0	1/4
C:	0	2/4	1/4	0
G:	2/4	1/4	2/4	0
T:	0	1/4	1/4	3/4

Slika 3.27: Matrice bez primene Laplasovog pravila [5]

Potrebno je da ažuriramo $Count(Motifs)$ i $Profile(Motifs)$ koristeći Laplasovo pravilo (Slika 3.28).

COUNT(*Motifs*)

A:	3	1	1	2
C:	1	3	2	1
G:	3	2	3	1
T:	1	2	2	4

PROFILE(*Motifs*)

A:	3/8	1/8	1/8	2/8
C:	1/8	3/8	2/8	1/8
G:	3/8	2/8	3/8	1/8
T:	1/8	2/8	2/8	4/8

Slika 3.28: Matrice nakon primene Laplasovog pravila [5]

Sada računamo verovatnoće svih 4-grama iz uklonjene prve sekvence skupa *Dna* (Slika 3.29).

ttAC	tACC	ACCT	CCTt	CTta	Ttaa	taac
$2/8^4$	$2/8^4$	$72/8^4$	$24/8^4$	$8/8^4$	$4/8^4$	$1/8^4$

Slika 3.29: Verovatnoće svih 4-grama iz prve sekvence [5]

Vraćamo u skup sekvenci uklonjenu prvu sekvencu, ali umesto *taac* ubacujemo motiv *ACCT*. Zatim uklanjamo četvrtu sekvencu iz skupa *Dna* i ponavljamo postupak (Slika 3.30).

	ttACCTtaac		ttACCTtaac
	gATGTctgtc		gATGTctgtc
Dna	ccgGCGTtag	⟶	ccgGCGTtag
	cactaACGAg		----------
	cgtcagAGGT		cgtcagAGGT

Slika 3.30: Uklanjanje četvrte sekvence iz skupa *Dna* [5]

Kao i u prethodnim postupcima, ažuriramo matrice $Count(Motifs)$ i $Profile(Motifs)$ koristeći Laplasovo pravilo (Slika 3.31).

Motifs

```
A C C T
G T c t
G C G T
A G G T
```

COUNT(*Motifs*)

```
A: 3 1 1 1
C: 1 3 3 1
G: 3 2 3 1
T: 1 2 1 5
```

PROFILE(*Motifs*)

```
A: 3/8 1/8 1/8 1/8
C: 1/8 3/8 3/8 1/8
G: 3/8 2/8 3/8 1/8
T: 1/8 2/8 1/8 5/8
```

Slika 3.31: Matrice nakon primene Laplasovog pravila [5]

Verovatnoće svih 4-grama iz uklonjenje četvrte sekvence su prikazane na Slici 3.32.

cact	acta	ctaA	taAC	aACG	ACGA	CGAg
$15/8^4$	$9/8^4$	$2/8^4$	$1/8^4$	$9/8^4$	$27/8^4$	$2/8^4$

Slika 3.32: Verovatnoće svih 4-grama iz četvrte sekvence [5]

Sada vraćamo uklonjenu četvrtu sekvencu, ali motiv *acta* menjamo sa *ACGA*. Algoritam počinje da konvergira.

Opisani algoritam ne mora uvek da vrati globalni maksimum. On mora biti pokrenut više puta kako bismo bili sigurni da je pronašao globalni maksimum, a ne lokalni maksimum. U opštem slučaju ovaj algoritam brzo konvergira i mala je verovatnoća da se će se zaglaviti u lokalnom maksimumu. U Tabeli 3.5 je prikazan rezultat algoritma testiran na *benchmark* kolekciji. *Gibbs Motif Search* pronalazi kolekciju motiva $Motifs$ sa konsenzus niskom AAAAAAgAGGGGGGt, pri čemu je $Score(Motifs) = 38$.

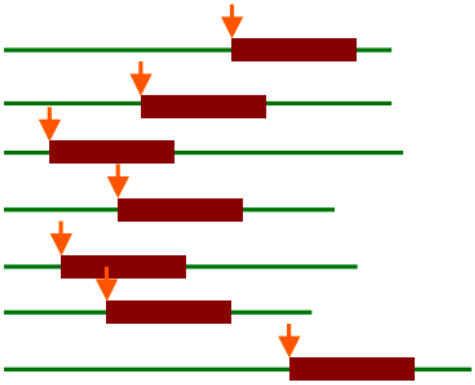
3.5 EM algoritam

U potrazi za regulatornim motivima u skupu sekvenci *Dna*, tj. pozicijama gde tražene instance motiva počinju u svakoj sekvenci analiziraćemo još jedan probabilistički algoritam. Na Slici 3.33 je dat primer početnih pozicija traženih instanci motiva. Za datu kolekciju sekvenci, neka je Z matrica u kojoj Z_{ij} odgovara verovatnoći da instanca motiva počinje na poziciji j u sekvenci i, a PWM (*probability weight matrix*) matrica koja sadrži frekvenciju svakog nukleotida na svakoj poziciji

Tabela 3.5: Rezultati testiranja algoritama na *benchmark* skupu

Algoritam	**k**	**Rešenje**	**Skor**
Median String	13	AAAAAtAGaGGGG	29
Median String	15	previše spor	
Greedy Motif	15	gtAAAtAgaGatGtG	58
Greedy Motif Laplace	15	AAAAAtAgaGGGGtt	41
Randomized Motif	15	AAAAAAAAacaGGGG	43
Gibbs Motif	15	AAAAAAgAGGGGGGt	38
Traženi (k, d) motiv:		AAAAAAAAGGGGGGG	

u motivu i u ostatku sekvence. Tada govorimo o raspodeli nukleotida unutar motiva (motivska raspodela (eng. *motif distribution*)) i o raspodeli nukleotida izvan motiva (pozadinska raspodela (eng. *background distribution*)). Možemo koristiti matricu PWM za dobijanje matrice Z, a zatim koristeći matricu Z da ažuriramo matricu PWM. Ova dva koraka se ponavljaju određeni broj puta ili do konvergencije [7]. Algoritam koji se zasniva na ova dva koraka naziva se EM algoritam (*Expectation-Maximization*) i u nastavku će biti detaljno opisan njegov rad.



Slika 3.33: Primer početnih pozicija traženih motiva [6]

E korak

Prvi korak u EM algoritmu je generisanje matrice PWM. Na početku PWM možemo inicijalizovati nasumično birajući početne pozicije motiva. Ukoliko tu matricu označimo sa p, $p_{c,k}$ je verovatnoća da se nukleotid c pojavi na poziciji k u motivu. Takođe računamo $p_{c,k}$ za $k = 0$ što predstavlja pozadinsku raspodelu motiva, tj.

verovatnoću nukleotida c na pozicijama u sekvenci izvan (pre i posle) kandidata motiva.

U E koraku generišemo matricu Z, koja nam predstavlja verovatnoće početnih pozicija instanci motiva u svakoj sekvenci. Neka je dat skup sekvenci Dna, pri čemu je X_i i-ta sekvenca iz Dna i neka je L dužina sekvence, a W dužina motiva. Tada verovatnoću sekvence X_i pri uslovu da motiv u toj sekvenci počinje na poziciji j dobijamo formulom:

$$Pr^t(X_i|Z_{ij}=1,p) = Pr(X_i|Z_{ij}=1) = \prod_{k=1}^{j-1} p_{c_k,0} \prod_{k=j}^{j+W-1} p_{c_k,k-j+1} \prod_{k=j+W}^{L} p_{c_k,0}.$$

Prvi i poslednji proizvod u formuli iznad odgovaraju verovatnoćama da deo sekvence koji nije kandidat da bude motiv potiče iz neke pozadinske raspodele, dok srednji proizvod predstavlja verovatnoću da razmatrani kandidat dolazi iz motivske raspodele motiva.

Z_{ij} u iteraciji t računamo po sledećoj formuli:
$Z_{ij}^t = \frac{Pr^t(X_i|Z_{ij}=1)Pr^t(Z_{ij}=1)}{\sum_{k=1}^{L-W+1} Pr^t(X_i|Z_{ik}=1)Pr^t(Z_{ik}=1)}$.

U nastavku ćemo kroz primer prikazati E korak. Neka je data i-ta sekvenca $GCTGTAG$, matrica p, gde su nasumično izabrane vrednosti, pri čemu je $W = 3$ (Slika 3.34).

$p =$

	0	1	2	3
A	0.25	0.1	0.5	0.2
C	0.25	0.4	0.2	0.1
G	0.25	0.3	0.1	0.6
T	0.25	0.2	0.2	0.1

Slika 3.34: Primer PWM matrice

Na osnovu matrice p, dobijamo da je:

1. $Z_{i,1} = 0.3 \cdot 0.2 \cdot 0.1 \cdot 0.25 \cdot 0.25 \cdot 0.25 \cdot 0.25$
2. $Z_{i,2} = 0.25 \cdot 0.4 \cdot 0.2 \cdot 0.6 \cdot 0.25 \cdot 0.25 \cdot 0.25$
3. $Z_{i,3} = 0.25 \cdot 0.25 \cdot 0.2 \cdot 0.1 \cdot 0.1 \cdot 0.25 \cdot 0.25$
4. $Z_{i,4} = 0.25 \cdot 0.25 \cdot 0.25 \cdot 0.3 \cdot 0.2 \cdot 0.2 \cdot 0.25$
5. $Z_{i,5} = 0.25 \cdot 0.25 \cdot 0.25 \cdot 0.25 \cdot 0.2 \cdot 0.5 \cdot 0.6.$

M korak

Nakon što smo izračunali Z, možemo koristiti rezultat kako bismo ažurirali PWM i verovatnoće nukleotida na pozicijama u sekvenci izvan (pre i posle) kandidata motiva. Ukoliko sa n_c označimo ukupan broj pojavljivanja karaktera c u skupu sekvenci, tada $n_{c,k}$ definišemo kao:

$$n_{c,k} = \begin{cases} \sum_i \sum_{\{j|X_{i,j+k-1}=c\}} Z_{ij}, & k > 0 \\ n_c - \sum_{j=1}^{W} n_{c,j}, & k = 0, \end{cases}$$

pri čemu unutrašnja suma za $k > 0$ ide po j-tim pozicijama koja može biti kanditat za početnu poziciju motiva dužine W i na kojima se nalazi nukleotid c.

PWM možemo ažurirati u iteraciji $t+1$ koristeći sledeću formulu:

$$p_{c,k}^{t+1} = \frac{n_{c,k} + d_{c,k}}{\sum_b (n_{b,k} + d_{b,k})},$$

pri čemu su $d_{c,k}$ i $d_{b,k}$ pseudovrednosti (najčešće 1), a $b \in \{A, C, G, T\}$.

U nastavku ćemo kroz primer prikazati M korak. Neka je na Slici 3.35 dat skup sekvenci i matrica Z.

A C A G C A
$Z_{1,1} = 0.1,\ Z_{1,2} = 0.7,\ Z_{1,3} = 0.1,\ Z_{1,4} = 0.1$

A G G C A G
$Z_{2,1} = 0.4,\ Z_{2,2} = 0.1,\ Z_{2,3} = 0.1,\ Z_{2,4} = 0.4$

T C A G T C
$Z_{3,1} = 0.2,\ Z_{3,2} = 0.6,\ Z_{3,3} = 0.1,\ Z_{3,4} = 0.1$

Slika 3.35: Kolekcija sekvenci *Dna* i primer matrice Z

$p_{A,1}$ računamo po formuli:

$$p_{A,1} = \frac{Z_{1,1} + Z_{1,3} + Z_{2,1} + Z_{3,3} + 1}{Z_{1,1} + Z_{1,2} ... + Z_{3,3} + Z_{3,4} + 4}.$$

Ispod je Algoritmom 8 predstavljen pseudokod EM algoritma.

E i M korake ponavljamo unapred određeni broj puta ili sve do konvergencije. Jedan od načina da znamo da li je profilna matrica konvergirala jeste da pratimo

Algoritam 8: EM algoritam (Dna, $motifLength$, $numIterations$)

```
1 set initial values for p
2 while i < numIterations do
3 |   re-estimate Z from p (E –step)
4 |   re-estimate p from Z (M-step)
5 end
6 return p, Z
```

koliko se svaki element u PWM promeni nakon maksimiziranja. Algoritam se zaustavlja ukoliko je promena ispod odabranog praga. Preporučljivo je ponovno pokrenuti algoritam sa različitim početnim pozicijama kako bi se pokušala smanjiti mogućnost konvergiranja na lokalnom maksimumu.

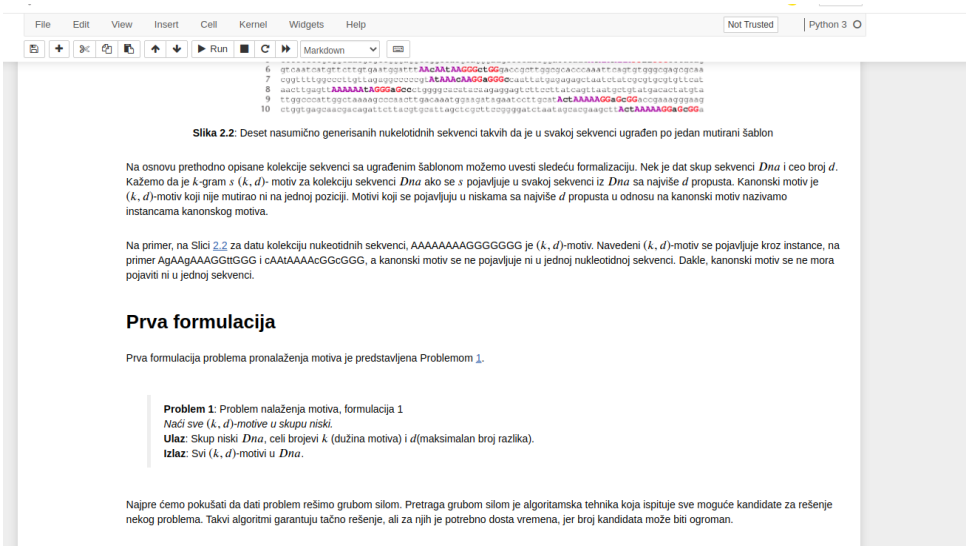
Tabela 3.5 neće biti proširena rezultatima testiranja EM algoritma na *benchmark* kolekciji, budući da su podaci iz tabele preuzeti iz udžbenika u kome ovaj algoritam nije obrađen, a u kome je prikazan samo segment kolekcije sekvenci.

Glava 4

Uputstvo za korišćenje elektronske lekcije

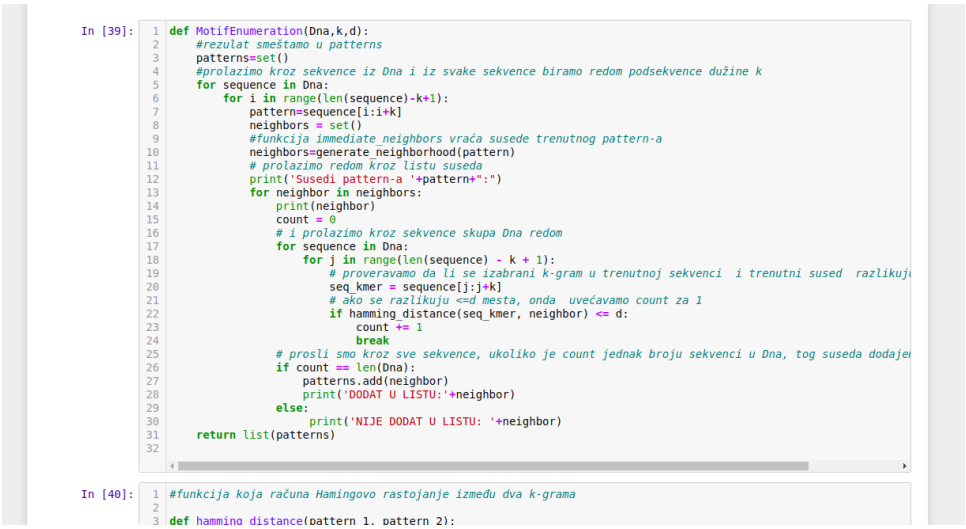
Elektronska lekcija se sastoji od teorijskog i interaktivnog dela. Za lekciju je korišćena interaktivna *Jupyter* sveska [2]. Da biste pokrenuli interaktivnu svesku, potrebno je instalirati *Python*, a zatim *Jupyter Notebook* pomoću *pip* komande (`pip install jupyter`). Da bi se sveska otvorila u pretraživaču, potrebno je u komandnoj liniji uneti `jupyter notebook`. Korišćen je operativni sistem *Linux*.

Teorijski deo služi korisniku da se upozna sa osnovnim pojmovima i problemima pronalaženja regulatornih motiva. Na Slici 4.1 je prikazan segment teorijskog dela elektronske lekcije.



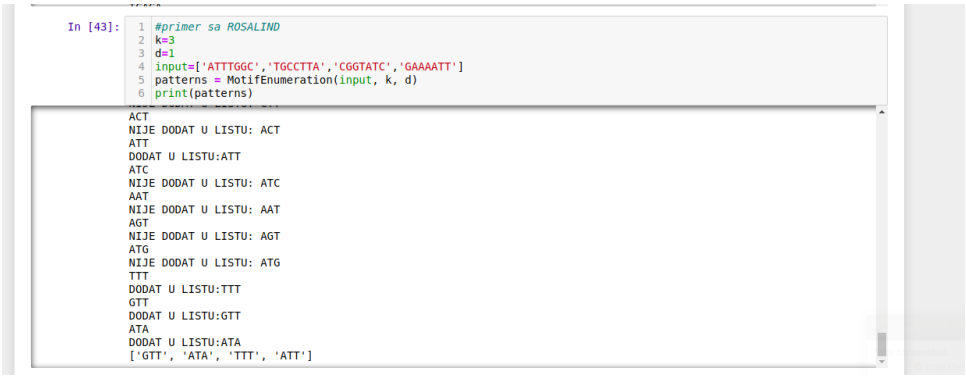
Slika 4.1: Primer teorijskog dela elektronske lekcije

Interaktivni deo služi korisniku da pokrene algoritme za pronalaženje regulatornih motiva koji su pisani u *Python*-u [3], pri čemu su algoritmi detaljno objašnjeni. Na Slici 4.2 je dat primer implementacije algoritma iz elektronske lekcije.



Slika 4.2: Primer implementacije algoritma iz elektronske lekcije

Nakon implementacije, algoritmi su testirani na primerima sa sajta [1]. Jedan test primer je prikazan na Slici 4.3.



Slika 4.3: Test primer

Glava 5

Zaključak

Ovaj rad pruža pregled različitih algoritama koji se koriste za pronalaženje regulatornih motiva u nukleotidnim sekvencama. Kroz detaljno istraživanje i analizu, prikazane su prednosti i nedostaci svakog algoritma.

U prvom delu su uvedeni osnovni pojmovi i date formulacije problema pronalaženja regulatornih motiva, a zatim su pobrojani algoritmi koji su predstavljeni u knjizi *Bioinformatics Algorithms: An Active Learning Approach* autora *Filipa Kompoa (Phillip Compeau)* i *Pavela Pevznera (Pavel Pevzner)*. Prvo su prikazani deterministički algoritmi koji imaju veliku složenost: algoritam grube sile i algoritam zasnovan na niski medijani, a potom probabiblistički algoritmi koji uglavnom nalaze približna rešenja: algoritam pohlepne pretrage, probabilistička pretraga motiva, Gibsovo uzorkovanje i EM algoritam.

Glavni cilj ovog rada bila je implementacija elektronske lekcije. Za lekciju je korišćena *Jupyter* sveska. Lekcija se sastoji od teorijskog i interaktivnog dela. Teorijski deo sadrži tekst koji je sličan kao u pisanom radu, dok interaktivni deo omogućuje da se pokrenu algoritmi za pronalaženje regulatornih motiva koji su pisani u *Python*-u, pri čemu je svaki algoritam detaljno opisan. Elektronska lekcija može poslužiti kao nastavno sredstvo za predavače, a studentima za samostalno istraživanje i učenje u interaktivnom okruženju.

Postoji dosta prostora za poboljšanja trenutno implementiranih algoritama. Fokus je bio na algoritmima koji se koriste na predavanju i vežbama kursa Uvod u Bioinformatiku. Postoje brojne modifikacije pomenutih algoritama koje nisu obrađene u radu, naročito za Gibsovo uzorkovanje.

Bibliografija

[1] Rosalind. 2012. zvanični sajt: https://rosalind.info/problems/list-view/?location=bioinformatics-textbook-track.

[2] Jupyter dokumentacija. 2022. zvanični sajt: https://jupyter-notebook.readthedocs.io/en/stable/.

[3] Python dokumentacija. 2023. zvanični sajt: https://docs.python.org/3/.

[4] Lubica Benuskova. *Sequence Motif Discovery.* University of Otago, New Zeland. zvanični sajt: http://www.cs.otago.ac.nz/cosc348/alignments/Lecture07_MotifSearch.pdf.

[5] Phillip Compeau and Pavel Pevzner. *Bioinformatics Algorithms: An Active Learning Approach, 2nd Edition, Vol. II.* Active Learning Publishers, LLC, 2015. zvanični sajt: https://www.bioinformaticsalgorithms.org/.

[6] Mark Craven. *Learning Sequence Motif Models Using Expectation Maximization (EM) and Gibbs Sampling.* University of Wisconsin–Madison, 2009. zvanični sajt: https://www.biostat.wisc.edu/bmi776/spring-09/lectures/lecture2.pdf.

[7] Manolis Kellis et al. *Computational Biology - Genomes, Networks, and Evolution.* Massachusetts Institute of Technology, 2022. zvanični sajt: https://bio.libretexts.org/Bookshelves/Computational_Biology/Book%3A_Computational_Biology_-_Genomes_Networks_and_Evolution_(Kellis_et_al.).

[8] Jovana Kovačević. *Prezentacija sa predavanja.* Uvod u Bioinformatiku, Matematčki fakultet, 2022. zvanični sajt: http://www.bioinformatika.matf.bg.ac.rs/predavanja/Chapter_2.pdf.

[9] Marijana Miljić. Master rad. 2023. repozitorijum dostupan na: `https://github.com/marijanar95/Master-rad`.

Biografija autora

Marijana Miljić rođena je 31. avgusta 1998. u Čačku. Gimnaziju u Guči je završila kao nosilac Vukove diplome. Smer matematika (računarstvo i informatika) na Matematičkom fakultetu Univerziteta u Beogradu upisala je 2017. godine, a završila 2021. godine. Nakon toga je upisala master studije na istom smeru. Od decembra 2021. godine do danas je zaposlena na poziciji Software developer u firmi ASW inženjering. Projekti na kojima je radila su uglavnom bili zasnovani na veb tehnologijama, a osnovni programski jezik u kojem je rađeno je Java.