FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

BIOINFORMATIKA

Određivanje poravnanja parova sljedova korištenjem HMM

Autori
Tomislav Božurić
Martin Pisačić
Krešimir Topolovec

Mentor
doc. dr. sc. Mirjana Domazet-Lošo

Siječanj, 2019

Sadržaj

1 Opis algoritma i vizualizacija

U ovome radu korišten je modificirani Viterbijev algoritam pomoću kojeg korištenjem dinamičkog programiranja možemo pronaći najvjerojatniju sekvencu skrivenih stanja koja ujedno predstavlja optimalno poravnanje dvaju nizova.

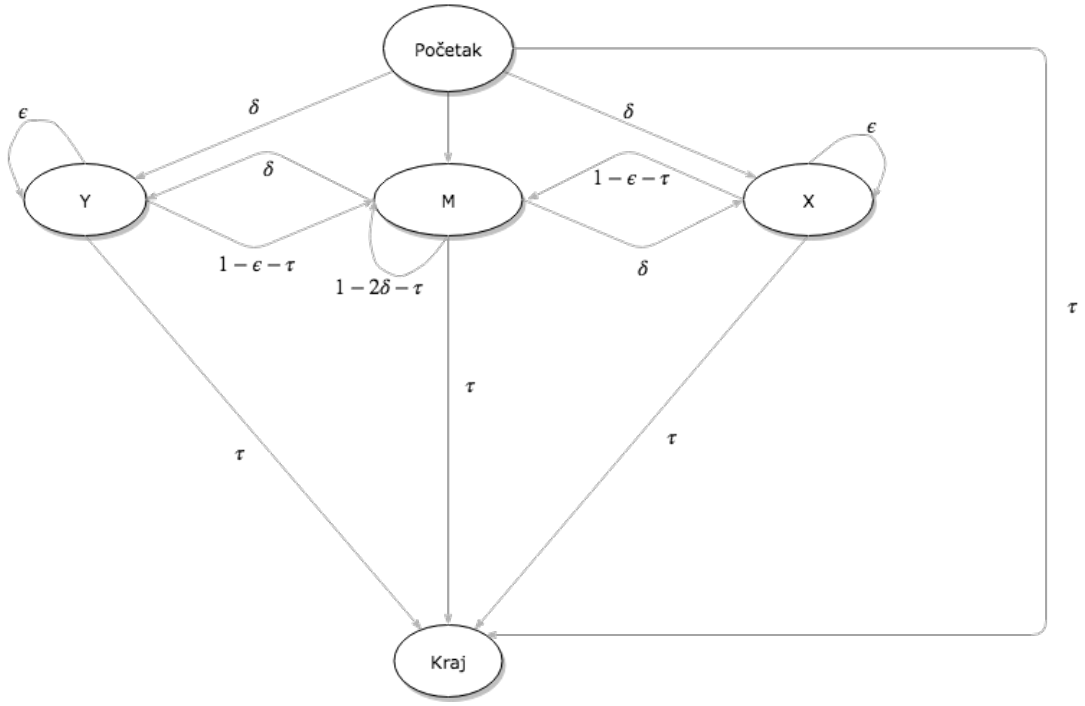
1.1 Primjena Viterbijevog algoritma na Pair-HMM

Da bismo klasičan Viterbijev algoritam tranformirali u Pair-HMM, moramo napraviti nekoliko izmjena. Prvo moramo odrediti vjerojatnosti za emitiranje simbola iz pojedinih stanja prikazanih na dijagramu 1. Npr. stanje $M(match)$ ima vjerojatnosnu distribuciju p_{ab} za emitaranje para simbola ab. Stanja X i Y imaju vjerojatnost emitiranja q_a simbola a i praznine u drugi niz. Dakle, ako iz stanja X emitiramo simbol x_i s vjerojatnošću q_{x_i}, u suprotni niz se ubacuje praznina. Nužno je potrebno definirati vjerojatnosti prijelaza između stanja tako da je suma vjerojatnosti odlaska iz pojedinog stanja jednaka 1. Vjerojatnost prijelaza iz stanja M u stanje X i Y opisujemo oznakom δ, a vjerojatnost u ostanka u stanju X ili Y sa ϵ. Takva definicija ne definira kompletni model koji omogućava vjerojatnosnu distribuciju po svim mogućim sekvecama. Za kompletiranje modela dodajemo početno i krajnje stanje *Pocetak* i *Kraj*. Također, definiramo vjerojatnost τ koja je jednaka za sve prijelaze iz stanja M, X i Y u stanje *End*, te za stanje *Begin* u stanja M. Iz razloga što ovakav tip modela generira dva poravnata niza nazivamo ga engl. pair HMM kako bi ga razlikovali od standardnih HMM koji generiraju jedan niz.

Generiranje nizova počinje iz stanja *Begin* i cirkulira između dva koraka i stanja M, X, Y sve dok proces ne dostigne stanje *End*. Navedena dva koraka su:

- odaberi novo stanje prema distribuciji vjerojatnosti prijelaza iz trenutnog stanja
- odaberi par simbola koji se dodaje u poravnanje prema definiranoj distribuciji vjerojatnosti emitiranja simobla

Bitno je uočiti da različita poravnanja mogu imati različite duljine zbog ovisnosti o distribuciji vjerojatnosti. Kako je vjerojatnost prijelaza iz stanja M, X ili Y u stanje *End* jednaka τ, očekivana duljina nizova jednaka je $1/\tau$.



Slika. 1: Model HMM-a korištenog za poravnanje parova sekvenci

1.2 Algoritam: Viterbijev algoritam za HMM [3]

Viterbijev algoritam dati će najvjerojatniji put kroz skrivena stanja u pair HMM.

Inicijalizacija:

$$v^M(0,0) = 1$$
$$v^\bullet(i,0) = v^\bullet(0,j) = 0$$

Korak:

za svaki $i = 1, ..., n$, $j = 1, ..., m$

$$v^M(i,j) = p_{x_i y_i} \max \begin{cases} (1-2\delta-\tau)v^M(i-1,j-1) \\ (1-\epsilon-\tau)v^X(i-1,j-1) \\ (1-\epsilon-\tau)v^Y(i-1,j-1) \end{cases} \tag{1}$$

$$v^X(i,j) = q_{x_i} \max \begin{cases} \delta v^M(i-1,j) \\ \epsilon v^X(i-1,j) \end{cases} \tag{2}$$

$$v^Y(i,j) = q_{y_j} \max \begin{cases} \delta v^M(i,j-1) \\ \epsilon v^Y(i,j-1) \end{cases} \tag{3}$$

Uvjet zaustavljanja: $v^E = \max(v^M(n,m), v^X(n,m), v^Y(n,m))$

Iz navedenih koraka jasno je da se radi o formi algoritma sličnoj kao kod poravnanja slijedova dinamičkim programiranjem. Možemo reči da Viterbijev algoritam spada u skupinu algoritama koji koriste dinamičko programiranje.
Zbog činjenice da su vjerojatnosti brojevi u intervalu $[0, 1]$, gore navedeni algoritam nije upotrebiv za implementaciju na računalu zbog velikog broja množenja brojeva bliskih nuli, pa se u praksi koristi logaritamska inačica tog algoritma koja je navedena u nastavku.

1.3 Algoritam: optimalno poravnanje logaritamskih kvota [3]

Inicijalizacija: $V^M(0,0) = -2log(\eta), V^X(0,0) = V^Y(0,0) = -\infty$
$V^\bullet(i,-1) = V^\bullet(-1,j) = -\infty$
Korak:
za svaki $i = 0, ..., n$, $j = 0, ..., m$ osim (0,0):

$$V^M(i,j) = s(x_i, y_j) + \max \begin{cases} V^M(i-1, j-1) \\ V^X(i-1, j-1) \\ V^Y(i-1, j-1) \end{cases} \tag{4}$$

$$V^X(i,j) = \max \begin{cases} V^M(i-1, j) - d \\ V^X(i-1, j) - e \end{cases} \tag{5}$$

$$V^Y(i,j) = \max \begin{cases} V^M(i, j-1) - d \\ V^Y(i, j-1) - e \end{cases} \tag{6}$$

Uvjet zaustavljanja: $V = \max(V^M(n,m), V^X(n,m) + c, V^Y(n,m) + c)$
Pri čemu su:
$s(a,b) = \log \frac{p_{ab}}{q_a q_b} + \log \frac{1-2\delta-\tau}{(1-\eta)^2}$
$d = -\log \frac{\delta(1-\epsilon-\tau)}{(1-\eta)(1-2\delta-\tau)}$
$e = -\log \frac{\epsilon}{1-\eta}$
$c = \log(1-2\delta-\tau) - \log(1-\epsilon-\tau)$

Tablica 1: Vizualizacija početnog stanja matrice V_M

indeks	-1	0	1	2	...	m
-1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	$-\infty$	$-2log\eta$				
1	$-\infty$					
2	$-\infty$					
...	$-\infty$					
n	$-\infty$					

U tablici iznad možemo vidjet početno stanje matrice u kojoj pamtimo logaritamske kvote za stanje podudaranja. Iz modificiranog Viterbijevog algoritma koji koristiti logaritamske kvote iterativno korištenjem dinamičkog programiranja punimo tablicu i upisujemo novo izračunate vrijednosti. Analogno radimo za tablice ispod, te kada dođemo do kraja iste te tablice koristimo za

rekonstrukciju optimalnog poravnanja. Želimo li izračunati primjerice za vrijednosti $i = 1, j = 0$ izraz za V_M bi izgledao ovako:

$$V^M(1,0) = s(x_1, y_0) + \max \begin{cases} V^M(0,-1) = -\infty \\ V^X(0,-1) = -\infty \\ V^Y(0,-1) = -\infty \end{cases} \tag{7}$$

Analogno popunjavamo preostali dio tablice. Ovdje možemo primjetiti kako za računanje vrijednosti u koraku i, j nam trebaju podatci iz koraka $i-1, j-1$, tako da je potrebno pararelno popunjavati sve tri prikazane matrice.

Tablica 2: Vizualizacija početnog stanja matrice V_X

indeks	-1	0	1	2	...	m
-1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	$-\infty$	$-\infty$				
1	$-\infty$					
2	$-\infty$					
...	$-\infty$					
n	$-\infty$					

Tablica 3: Vizualizacija početnog stanja matrice V_Y

indeks	-1	0	1	2	...	m
-1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
0	$-\infty$	$-\infty$				
1	$-\infty$					
2	$-\infty$					
...	$-\infty$					
n	$-\infty$					

Kada smo završili s forward dijelom algoritma, odnosno kada su nam sve strukture podatka popunjene spremni za pronalazak optimalnog poravnanja dvije sekvence. Kako nam je uvjet "zaustavljanja" $V = \max(V^M(n,m), V^X(n,m) + c, V^Y(n,m) + c)$, tražimo maksimum između navednih stanja. Zatim generiramo poravnanje te u povratku prema početku ponavljamo postupak tražeći maksimum između V_M, V_X, V_Y.Na idućem primjeru je ilustriran primjer pronalaska optimalnog poravnanja gdje se na poziciji (n,m) vrijednost računa kako je prethodno spomenuto.

Tablica 4: Vizualizacija pronalaska optimalnog poravnanja

0	1	2	3	...	m-1	m	
0	$V^\bullet(0,0)$						
1		$V^\bullet(1,1)$					
2							
3							
...							
n-1				...		$V^\bullet(n-1,m-1)$	
n							$V^\bullet(n,m)$

1.4 Memorijski optimalniji algoritam optimalnih poravnanja logaritamskih kvota

U prošlome poglavlju prikazali smo algoritam poravnanja logaritamskih kvota koji se često koristi u praksi jer vrijednosti iz iteracije u iteraciju ne iščezavaju kao što je to slučaj kod klasične vjerojatnosne izvedbe. Pogledamo li ponovno jednadžbe algoritma:

Korak:
za svaki $i = 0, ..., n$, $j = 0, ..., m$ osim (0,0):

$$V^M(i,j) = s(x_i, y_j) + \max \begin{cases} V^M(i-1, j-1) \\ V^X(i-1, j-1) \\ V^Y(i-1, j-1) \end{cases}$$

$$V^X(i,j) = \max \begin{cases} V^M(i-1, j) - d \\ V^X(i-1, j) - e \end{cases}$$

$$V^Y(i,j) = \max \begin{cases} V^M(i, j-1) - d \\ V^Y(i, j-1) - e \end{cases}$$

intuitivno je kako se za samo izvršavanje algoritma svi međurezultati mogu pohranjivati u 2D polja, iz kojih se u narednim koracima uzimaju prethodno izračunate vrijednosti. Prednost ovog pristupa svakako je njegova jednostavnost, no mana je što za veće instance sekvenci koje se poravnavaju memorija sve više raste, pri čemu računalo više nije niti u stanju pohraniti sve strukture u svoju radnu memoriju. No ovaj problem može se elegantno riješiti ako pogledamo jednadžbe koje ponavljamo iz koraka u korak. Pogledamo li malo bolje jednadžbe, vidimo da u nekome koraku i i j vrijednosti za taj korak računamo samo u ovisnost o vrijednostima iz prethodnoga koraka ili o vrijednostima iz trenutnoga koraka ali iz ranijih iteracija. Pošto prilikom programske implementacije vrijednosti i i j vrtimo kao dvije ugniježđene petlje, tako ćemo za svaku vrijednost i proći kroz sve vrijednosti j što se u našim indeksima uvijek odnosi na dvije vrijednosti iz ranije iteracije (označenih sa $i-1$) te iz vrijednost trenutne iteracije (označenih sa $j-1$).
Modificirana verzija algoritma radi na sljedeći način:

1. Kreiraj 3 polja M', X', Y' ukupne veličine M
2. Inicijaliziraj vrijednosti polja na početne vrijednosti algoritma
3. za svaki $i = 0, ..., N$
   kreiraj pomoćna polja M'_p, X'_p, Y'_p
   za svaki $j = 0, ..., M$
   ažuriraj:
   $M'_p[j] = s(x_i, y_j) + max(M'[j-1], X'[j-1], Y'[j-1])$
   $X'_p[j] = max(M'[j-1] - d, X'[j-1] - e)$
   $Y'_p[j] = max(M'_p[j-1] - d, Y'_p[j-1] - e)$
   kraj petlje
   zamijeni:
   $M' \leftarrow M'_p$
   $X' \leftarrow X'_p$
   $Y' \leftarrow Y'_p$

kraj petlje

1.5 Procjena parametara

Parametre skrivenog Markovljevog modela procjenili smo statističkom metodom maksimalne izglednosti parametara (*Maximum Likelihood Estimation*) s zaglađivanjem nad bazom prikupljenih parova sekvenci ebole, hepatitisa, HIV-a, virusa bjesnoće i tripomizina. Nakon procjena vjerojatnosti prijelaza iz stanja u stanje, kako naš skriveni Markovljev model koristi tri različite vjerojatnosti (δ, τ, ϵ), prijelaze iz jednog stanja u neko drugo stanje smo odredili uprosječivanjem odgovarajućih prijelaza.

$$e(\alpha, \beta) = \frac{E(\alpha,\beta)+smoothing}{\sum_{\alpha,\beta} E(\alpha,\beta)+2 \cdot smoothing}$$

Gornju formulu koristimo za procjenu vjerojatnosti generiranja poravnatog para, odnosno za generiranje matrice vjerojatnosti emitiranja parova simbola. Primjerice za procjenu vjerojatnosti emitiranja para simbola (A,T) ili (C,G) analogno za sve ostale parove. Funkcija $E(\alpha, \beta)$ predstavlja broj pojavljivanja para (α, β) u poravnatim sekevencama iz baze podataka za učenje. Sumu u nazivniku možemo lakše iterpretirati i kao duljinu poravnatih sekvenci. Parametar *smoothing* je uveden kako bi i parovima koji se nisu pojavili u bazi podataka za učenje dodjelili određenu vjerojatnost, odnosno kako bi naš model postao robustniji, kako bi mu povećali efikasnu moć generalizacije i smanjili pretreniranost na podatke koje je već "vidio".

$$t(X, Y) = \frac{T(X,Y)+smoothing}{\sum_{X,Y} T(X,Y)+2 \cdot smoothing}$$

Gornju formulu koristimo za procjenu vjerojatnosti priejlaza između stanja skrivenog Markovljevog modela. Kako smo već ustanovili da naš model koristi tri vjerojatnosti (δ, τ, ϵ) nakon dobivene tranzicijske matrice korištenjem gornje formule za sve prijelaze između stanja koji imaju definiranu istu vjerojatnost računamo srednju vrijednost. U navednoj formuli $T(X, Y)$ označava broj prijelaza između stanja X u Y. Primjerice T(M,X) bi predstavljao koliko puta smo imali prijelaz između stanja M u X u bazi poravnatih sekvenci. Analogno suma u nazivniku predstavlja ukupan broj svih prijelaza u bazi poravnatih sekvenci. Analogno smo koristili *smoothing* parametar kako bi model bolje generalizirao.

2 Testiranje, analiza točnosti, vremena izvođenja i utroška memorije

Vremenska i memorijska složenost modificiranog Viterbijevog algoritma jest $\mathcal{O}(NM)$. U svrhu analize točnosti algoritma korištena je usporedba poravnanja s alatom MAFFT (`https://mafft.cbrc.jp/alignment/software/`).

2.1 Testiranje performansi algoritma

Za primjer testiranja performansi programa izvučeno je nekoliko parova sekvenci nad kojima smo izvršili dani algoritam te mjerili vrijeme izvođenja i memorijsku potrošnju. Primjeri su odabrani tako da predstavljaju sekvence različitih duljina kako bi vidjeli kako duljina sekvenci koje se poravnavaju utječe na same performanse algoritma. Kako je i predviđeno, što su sekvence koje se poravnavaju duže, tako je i trajanje izvođenja algoritma dulje, a memorijski utrošak veći.

Specifikacije računala na kojemu se algoritam izvršava su:

- OS: Ubuntu 18.04.1 LTS 64 bit
- CPU: Intel® Core™ i7-7500U CPU @ 2.70GHz
- RAM: 12 GB DDR4, 2133 MHz
- Disk: Samsung 850 EVO SSD, brzina pisanja do 520 MB/s, brzina slijednog čitanja do 540 MB/s

Rezultati izvođenja algoritma dani su u tablici ispod:

#	*Sekvenca 1*	*Duljina*	*Sekvenca 2*	*Duljina*	*Vrijeme izvodenja*	*Max utrošak memorije*
1	AF086833.2 Ebola virus - Mayinga, Zaire, 1976, complete genome	18959	JF828358.1 Lloviu virus strain MS-Liver-86/2003, complete genome	18927	4,9 min / 297 s	5 GiB
2	HIV: Ref.A1 .RW .92 .92RW008 .AB253421	9643	HIV:Ref.A1 .UG .92 .92UG037 .AB253429	9690	79 s / 76594 ms	1,3 GiB
3	AF384372.1 Hepatitis B virus isolate G376-A6, complete genome	3125	NC_004102.1 Hepatitis C virus genotype 1, complete genome	9646	25016 ms	431,4 MiB
4	FJ424484.1 Rabies virus red fox/08RS-1981/Udine/2008 nucleoprotein (N) gene, partial cds	1350	MG996466.1 Rabies lyssavirus isolate VIEV_RV_Dog_69_461_2016 nucleoprotein gene, complete cds	1353	1585 ms	26,2 MiB
5	embl:BF056441 BF056441; 7k05a04.x1 NCI_CGAP_GC6 Homo sapiens cDNA clone IMAGE:3443238 3' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.	675	embl:BE848719 BE848719; uw40c07.y1 Soares_thymus_2NbMT Mus musculus cDNA clone IMAGE:3419148 5' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.	699	474 ms	6.8 MiB

Tablica 5: Trajanje izvođenja i memorijski utrošak izvođenja algoritma

Koristeći memorijski optimiziranu verziju algoritma koja umjesto 2D polja čuva vrijednosti samo za prethodni i trenutni korak u vidu 1D polja, postigli smo puno manju iskoristivost memorije, a koja se najbolje vidim na većim instancama (pr. virus ebole). Ovim pristupom vrijeme izvođenja se nije promijenilo i vremenska složenost je ostala ista, no memorijska složenost se smanjila. Rezultati su prikazani u tablici ispod:

#	*Sekvenca 1*	*Sekvenca 2*	*Vrijeme izvodenja*	*Max utrošak memorije*
1	AF086833.2 Ebola virus - Mayinga, Zaire, 1976, complete genome	JF828358.1 Lloviu virus strain MS-Liver-86/2003, complete genome	4,95 min / 297 s	1,0 GiB
2	HIV: Ref.A1 .RW .92 .92RW008 .AB253421	HIV:Ref.A1 .UG .92 .92UG037 .AB253429	79 s / 76594 ms	267,8 MiB
3	AF384372.1 Hepatitis B virus isolate G376-A6, complete genome	NC_004102.1 Hepatitis C virus genotype 1, complete genome	25196 ms	86,6 MiB
4	FJ424484.1 Rabies virus red fox/08RS-1981/Udine/2008 nucleoprotein (N) gene, partial cds	MG996466.1 Rabies lyssavirus isolate VIEV_RV_Dog_69_461_2016 nucleoprotein gene, complete cds	1579 ms	5,4 MiB
5	embl:BF056441 BF056441; 7k05a04.x1 NCI_CGAP_GC6 Homo sapiens cDNA clone IMAGE:3443238 3' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.	embl:BE848719 BE848719; uw40c07.y1 Soares_thymus_2NbMT Mus musculus cDNA clone IMAGE:3419148 5' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.	414 ms	1,5 MiB

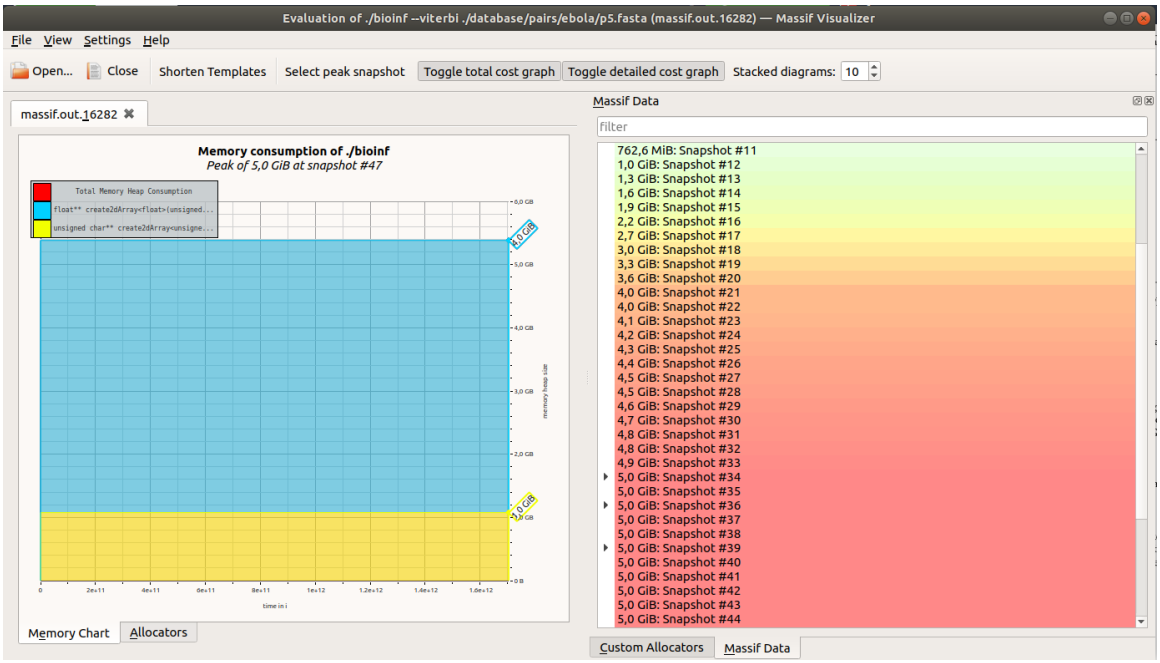
Tablica 6: Trajanje izvođenja i memorijski utrošak izvođenja memorijski optimizirane verzije algoritma

Ono što možemo primjetiti je kako se memorija u svakoj instanci umanjila za otprilike 5 puta što je znatno osjetnije na većim instancama.

2.2 Grafovi utroška memorije za navedene slučajeve

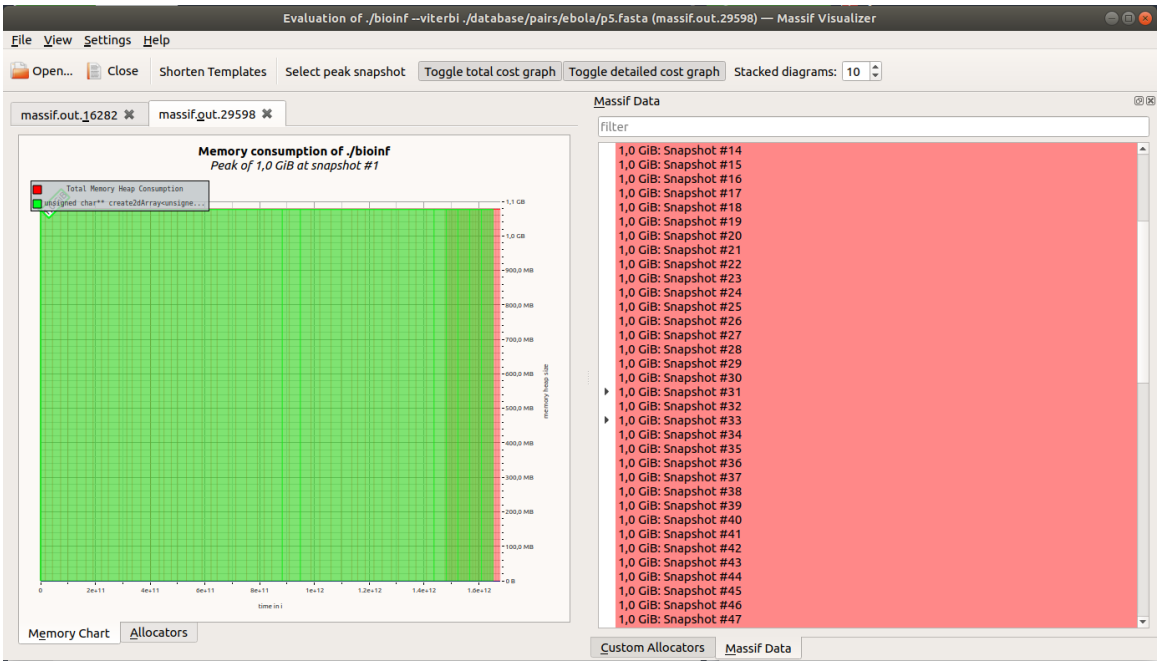
U ovome podpoglavlju dajemo uvid u potrošnju memorije za svaki od gore navedenih slučajeva testiranja programa na parovima sekvenci. Ovdje možemo vidjeti koji djelovi algoritma troše najviše memorije, te što to znači za samu efikasnost i upotrebu algoritma. Grafove utroška prikazati ćemo na primjeru instance *1* iz navedenih tablica.

U klasičnoj verziji algoritma za sva tri moguća stanja: M, X, Y pratimo sve viđene vjerojatnosti u matricama dimenzija N x M, a memorijsku potrošnju možemo vidjeti na slici ispod:



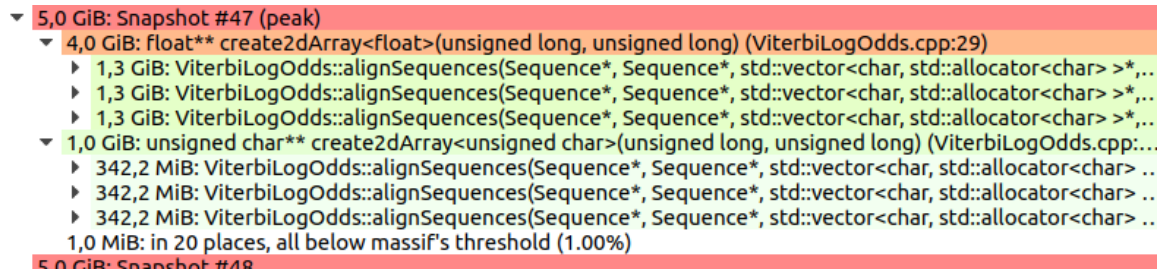
Slika. 2: Memorijska potrošnja klasičnog algoritma

Graf utroška memorije za optimizirani algoritam dan je na slici ispod:



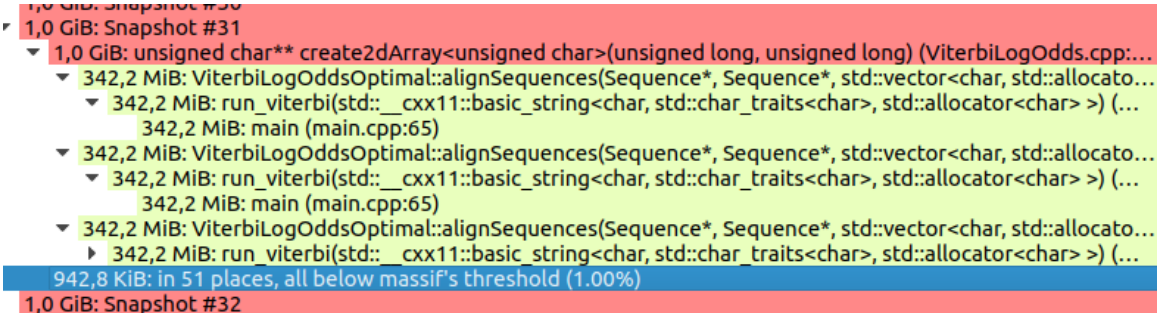
Slika. 3: Memorijska potrošnja optimiziranog algoritma

Ono što također možemo primjetiti je i koje strukture zauzimaju najviše memorije, što je prikazano na slikama ispod:



Slika. 4: Strukture podataka za klasični algoritam

Graf utroška memorije za optimizirani algoritam dan je na slici ispod:



Slika. 5: Strukture podataka za optimizirani algoritam

Kao što je i očekivano, 2D polja (matrice) zauzimaju više memorije no što to čine klasična polja.

2.3 Analiza točnosti

Za provjeru kvalitete rješenja računali smo *score* na temelju supstitucijske tablice **Blosum62** i pri tome smo definirali kaznu za otvaranje praznine -5, i kaznu za produživanje praznine kao -1. Rezultate smo usporedili s alatom *MAFFT* koji je očekivano dao bolje rezultate jer je naš model vjerojatnosne distribucije učio na ograničenoj bazi podataka pa sam procjena parametara ovisi o toj bazi poravnatih sekvenci. U nastavku su dane usporedbe za nekoliko sekvenci.

Sekvenca	*Pairwise HMM score*	*MAFFT score*
HIV:Ref.A1.RW.92.92RW008.AB253421, Ref.A1.UG.92.92UG037.AB253429	10296	49486
AF086833.2 Ebola virus - Mayinga, Zaire, 1976, complete genome, JF828358.1 Lloviu virus strain MS-Liver-86/2003, complete genome	13473	46349
FJ424484.1 Rabies virus red fox, MG996466.1 Rabies lyssavirus	1268	7439
Tropomyiosin : Homo sapiens cDNA, Soares Thymus Mus musculus cDNA	545	1761
Hepatitis B virus isolate G376-A6, complete genome, Hepatitis C virus genotype 1, complete genome	-6504	2691

Tablica 7: Usporedba rezultata poravnanja modela učenog na bazi više virusa poravnatoj s alatom *MAFFT*

Nakon procjenjenih parametara na temelju baze poravnatih sekvenci ebole, hepatitisa, HIV-a, virusa bjesnoće i tropomizina pokušali smo bazu podataka izgraditi koristeći alat *ClustalW* nad parovima sekvenci HIV-a. Zatim smo iste te parove poravnali alatom *MAFFT* i dobili smo približno jednake rezultate za poravnate sekvence našeg modela i izlaza *MAFFT*-a za različite sekvence.

Sekvenca	*Pairwise HMM score*	*MAFFT score*
HIV:Ref.A1.RW.92.92RW008.AB253421, Ref.A1.UG.92.92UG037.AB253429	46609	49486
HIV:Ref.A1.RW.92.92RW008.AB253421, Ref.A2.CD.97.97CDKTB48.AF286238	36264	42534
Tropomyiosin : Homo sapiens cDNA, Soares Thymus Mus musculus cDNA	855	1761
Hepatitis B virus isolate G376-A6, complete genome, Hepatitis C virus genotype 1, complete genome	-3273	2691
Nasumične sekvenca duljine 101 i 105 znakova	102	139
Nasumične sekvenca duljine 5097 i 5053 znakova	4300	8751

Tablica 8: Usporedba rezultata poravnanja modela učenog na bazi HIV-a poravnatoj s *ClustalW*-om

Isproban je i model skrivenog Markovljevog modela učen na temelju baze poravatih sekvenci s alatom *ClustalW* na virusima ebole i HIV-a, on je dao bolje rezultate kod poravnanja od prvog modela za viruse na kojima smo testirali (HIV, hepatitis i tropomizin), no bio je lošiji od modela učenog samo na bazi HIV-a s identičnim alatom.

3 Zaključak

Završetkom projekta možemo reći da je bio vrlo izazovan i zanimljiv za rješavanje. Susreli smo se s proučavanjem znanstvenih članaka i njihovom implementacijom, iz prve ruke vidjeli kako memorijska i vremenska složenost utječu na izvršavanje programa te optimizacijom samih algoritama. Projekt je naposlijetku davao zadovoljavajuće rezultate poravnanja parova, iako mu je potrebno nešto više vremena nego klasičnim alatima (*ClustalW, T-Coffe, ...*). Daljna poboljšanja bi mogla ići u smjeru pokušaja daljnje optimizacije utroška memorije i vremena, odnosno na dodatnu analizu samog koda i traženja točaka koje bi mogle pridonjeti ubrzanju rada algoritma. Dodatno, mogli bismo model učiti nad većom bazom parova poravnatih sekvenci kako bi bolje procijenili parametre i napravili model robustnijim. Kvaliteta samog poravnanja ponajviše ovisi o procjenjenim parametrima, odnosno alatima koje koristimo za poravnanje parova i izgradnju baze poravnatih sekvenci.

4 Literatura

[1] Mile Šikić, Mirjana Domazet-Lošo. *Bioinformatika*. Fakultet elektrotehnike i računarstva, 2013.

[2] Byung-Jun Yoon. *Hidden Markov Models and their Applications in Biological Sequence Analysis*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766791, US National Library of Medicine, National Institutes of Health, 2009.

[3] Jun Xie. *Pairwise alignment using HMM*. http://www.stat.purdue.edu/~junxie/topic4.pdf, Purdue University.

[4] Luay Nakhleh,. *Pairwise HMMs and Sequence Alignment*. https://www.cs.rice.edu/~nakhleh/COMP571/Slides-Spring2015/SequenceAlignment-PairwiseHMM.pdf, Rice University, 2015.