SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD BR. 1901

PORAVNAVANJE PROTEINSKIH NIZOVA UZ POMOĆ GENETSKIH ALGORITAMA

Bojan Dunaj

Zagreb, lipanj 2011.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD BR. 1901

PORAVNAVANJE PROTEINSKIH NIZOVA UZ POMOĆ GENETSKIH ALGORITAMA

Bojan Dunaj

Zagreb, lipanj 2011.

Sadržaj

1 Uvod

U lipnju 2000., Sanger Center (Cambridge, Engleska), objavio je prvu radnu verziju ljudskog genoma, kôda dugog 3 milijarde znakova koji razlikuje *homo sapiensa* od ostalih vrsta. Ovo monumentalno postignuće zasigurno je od presudnog značaja za znanost, međutim doprinos *znanju* tada nije bio tako očigledan. Ako znanje u danom kontekstu smatramo skupom odgovora na pitanja poput "Koji su geni odgovorni za imunološki sustav?", ili "Jesmo li bliži srodnici čimpanzama ili gorilama?", tada objavljivanje cjelovitog niza DNK nije trenutno donijelo značajne promjene u našim spoznajama. Postoji vidljiv nesrazmjer između dostupnosti novih podataka o biološkim nizovima, s jedne strane, te znanstvenog razumijevanja dostupnih podatka, s druge. Taj nesrazmjer smanjuje *bioinformatika* – interdisciplinarno polje koje ujedinjuje biologiju, računarske znanosti, matematiku, statistiku i teoriju informacija, kako bi se analizirali biološki podaci radi interpretacije i predviđanja.

Bioinformatika nikada nije bila popularna kao što je to danas slučaj. Genomska revolucija usko je vezana uz napretke u računarskoj znanosti, a svaki projekt pronalaska genoma kao posljedicu ima veoma brzo prikupljanje velikih količina podataka, značenje kojih se otkriva kasnijim interpretacijama. Računala, osim u skladištenju, imaju ključnu ulogu u toj interpretaciji. Problemi poput identifikacije i ekspresije gena, savijanja RNK i proteina, poravnavanja bioloških nizova u svrhu otkrivanja evolucijskih promjena, srodnosti i sl.; analize metaboličkih putova itd., nose sa sobom prilično visoke zahtjeve na računalne resurse. Farmaceutske kompanije su posebno zainteresirane za uporabu bioinformatike u svrhu pronalaženja novih lijekova.

Mnogi od najvažnijih problema toliko su veliki, da je iscrpna pretraga prostora rješenja praktično neizvediva, a standardne metode i algoritmi kojima raspolažu biolozi zahtijevaju previše vremena, novca i/ili računalnih resursa. U takvim slučajevima, rješenje je obično u postavljanju jednostavnije hipoteze (koja često dovodi do pravih odgovora na kriva pitanja [1]), ili u uporabi računalnih algoritama koji su u mogućnosti pretraživati velike prostore rješenja u razumnom vremenu. Upravo kod takvih problema na vidjelo izlazi snaga evolucijskog računarstva. Nalazimo se u vremenu kada je mogućnost iskorištavanja znanja o evolucijskim procesima u rješavanju problema možda i jednako uzbudljiva kao i sama evolucija u prirodnim sustavima.

Poravnavanje nizova je konfiguracija dvije ili više nizova rezidua – nukleotida(DNK) ili aminokiselina(proteina) – koja pokušava pronaći maksimalnu sličnost među njima. Svoju primjenu nalazi u rekonstrukciji filogenetskih stabala; kao sredstvo predviđanja funkcije i/ili strukture nepoznatog proteina poravnavajući ga s proteinima kojima su struktura ili funkcija već poznati; kao sredstvo pronalaženja uzoraka za karakterizaciju familija proteina[2], itd. Unatoč važnosti problema, poravnavanje nizova ostaje jedan od najvećih izazova bioinformatike.

U ovom radu korišten je pristup evolucijskog računarstva, konkretno genetskog algoritma, koji u kontekstu poravnavanja nizova spadaju u skupinu iterativnih heurističkih metoda. Genetski algoritam je realiziran kao rješenje minimalnog broja i kompleksnosti operatora (u usporedbi s jednim od postojećih rješenja temeljenih na genetskim algoritmima[3]), te je ovaj rad istraživanje dosega takvog koncepta.

2 Uvod u bioinformatiku

Može se reći da je biologija tradicionalno opservacijska i laboratorijska znanost, međutim, uz nevjerojatan pritok golemih količina podataka u genetici, genomici i proteomici, te uz pomoć bioinformatike i sličnih disciplina, ona zasigurno postaje i dedukcijska i informacijska znanost.

Bioinformatika se bavi analizom, skladištenjem, manipulacijom i interpretacijom makromolekula kao što su DNK, RNK ili proteini. Također, može se definirati kao primjena statistike i računarske znanosti na područje molekularne biologije. Primarni je cilj ove discipline bolje razumijevanje bioloških procesa, a ono što je razlikuje od drugih pristupa je usredotočenost na razvoj računalno intenzivnih tehnika kao što su raspoznavanje uzoraka, rudarenje podataka, strojno učenje, vizualizacije.[4]

Glavni istraživački napori u ovom polju uključuju: identifikaciju gena, sastavljanje genoma, usporedbe genoma, dizajniranje lijekova, otkrivanje lijekova, strukturno poravnanje proteina, predviđanje strukture proteina, predviđanje strukture RNK[5], predviđanje ekspresije gena i proteinsko-proteinskih reakcija, modeliranje evolucije, te poravnavanje nizova (DNK, RNK i proteina) koje je centralna tema ovog rada.

2.1 Centralna dogma molekularne biologije

Izraz iz naslova prvi je upotrijebio Francis Crick[6], jedan od suotkrivača strukture DNK molekule. Dogma govori o prijenosu informacija u stanici, i platforma je za razumijevanje pretvorbe informacija sadržanih u nizovima biopolimera: DNK, RNK i protein. Općenito, transferi koji predstavljaju normalan tok informacija u živim organizmima su sljedeći:

- kopiranje DNK informacije u RNK (transkripcija):

  Proces stvaranja ekvivalentne kopije molekule RNK koristeći niz nukleotida molekule DNK. DNK i RNK su nukleinske kiseline koje koriste parove baza kao komplementarni jezik koji može biti preveden sa DNK u RNK u prisustvu odgovarajućih enzima. Za vrijeme transkripcije, niz DNK je preveden pomoću enzima RNK-polimeraze, koja proizvodi komplementarni, antiparalelni RNK lanac. Za razliku od replikacije DNK,

transkripcija u RNK lanac uključuje bazu uracil (U) na mjestima gdje bi se, u DNK lancu, pojavio timin (T). Transkripcija je prvi korak koji vodi prema ekspresiji gena. Odsječak DNK preveden u molekulu RNK kodira barem jedan gen. Ako transkribirani (prevedeni) gen kodira protein, rezultat transkripcije je molekula (mRNK). Molekula (mRNK) će zatim stvoriti protein kroz proces translacije.[7]

- kopiranje RNK informacije u protein (translacija)

  Za vrijeme translacije, ribosom dekodira molekulu mRNK proizvedenu transkripcijom, kako bi se proizveo lanac aminokiselina koji će se kasnije **saviti** u protein. Jednu aminokiselinu u lancu kodira tri susjedna nukleotida mRNK, koji se nazivaju kodon. Tablica 2.1 prikazuje popis 20 aminokiselina u organizmu, njihove nazive, kratice te okvirnu veličinu. Tablica 2.2 je RNK kodon tablica – prikazuje kodone i pripadajuće aminokiseline u koje se translatiraju.

- kopiranje DNK u DNK (replikacija DNK)

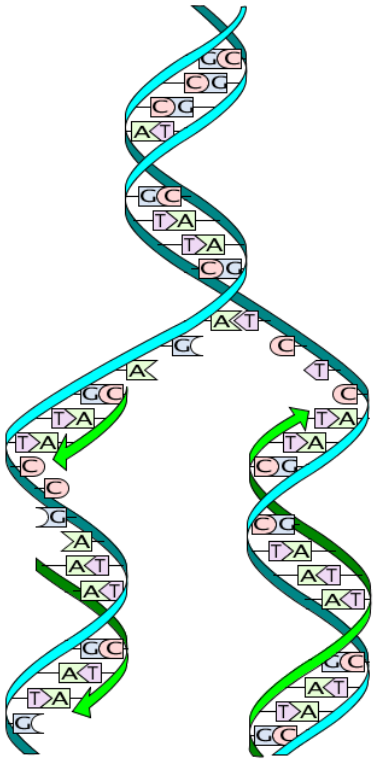
  Replikacija počinje na određenim mjestima u genomu, takozvanim *origin*-ima. Prvotno se dvostruka spirala odmotava i razdvaja na dva komplementarna niza, a zatim se pomoću slobodnih nukleotida i enzima DNK-polimeraze na svakom od dva niza stvaraju komplementarni nizovi, što će rezultirati s dvije nove spirale, identične originalnoj. Svaki se DNK niz sastoji od četiri moguća nukleotida: A (adenin), C (citozin), T (timin), G(guanin). Nukleotidi se vežu vodikovim vezama te formiraju bazne parove. Adenin se veže s timinom, a citozin s guaninom. Slika 2.1 na pojednostavljen način prikazuje proces.

Tablica 2.1 Aminokiseline – nazivi i kratice

Aminokiselina	**Kratica(1)**	**Kratica(3)**	**Veličina**
Alanin	ala	A	Mali
Arginin	arg	R	Veliki
Asparagin	asn	N	Srednji
Asparaginska kiselina	asp	D	Srednji
Cistein	cys	C	Mali
Glutaminska kiselina	glu	E	Veliki
Glutamin	gln	Q	Veliki
Glicin	gly	G	Mali
Histidin	his	H	Veliki
Izoleucin	ile	I	Srednji
Leucin	leu	L	Srednji
Lizin	lys	K	Veliki
Metionin	met	M	Veliki
Fenilalanin	phe	F	Srednji
Prolin	pro	P	Srednji
Serin	ser	S	Mali
Treonin	thr	T	Mali
Triptofan	trp	W	Veliki
Tirozin	tyr	Y	Srednji
Valin	val	V	Mali

Tablica 2.2 - RNA kodoni i pripadajuće aminokiseline

		2nd base							
		U		C		A		G	
1st base	U	UUU	(Phe/F) Phenylalanine	UCU	(Ser/S) Serine	UAU	(Tyr/Y) Tyrosine	UGU	(Cys/C) Cysteine
		UUC	(Phe/F) Phenylalanine	UCC	(Ser/S) Serine	UAC	(Tyr/Y) Tyrosine	UGC	(Cys/C) Cysteine
		UUA	(Leu/L) Leucine	UCA	(Ser/S) Serine	UAA	Stop (*Ochre*)	UGA	Stop (*Opal*)
		UUG	(Leu/L) Leucine	UCG	(Ser/S) Serine	UAG	Stop (*Amber*)	UGG	(Trp/W) Tryptophan
	C	CUU	(Leu/L) Leucine	CCU	(Pro/P) Proline	CAU	(His/H) Histidine	CGU	(Arg/R) Arginine
		CUC	(Leu/L) Leucine	CCC	(Pro/P) Proline	CAC	(His/H) Histidine	CGC	(Arg/R) Arginine
		CUA	(Leu/L) Leucine	CCA	(Pro/P) Proline	CAA	(Gln/Q) Glutamine	CGA	(Arg/R) Arginine
		CUG	(Leu/L) Leucine	CCG	(Pro/P) Proline	CAG	(Gln/Q) Glutamine	CGG	(Arg/R) Arginine
	A	AUU	(Ile/I) Isoleucine	ACU	(Thr/T) Threonine	AAU	(Asn/N) Asparagine	AGU	(Ser/S) Serine
		AUC	(Ile/I) Isoleucine	ACC	(Thr/T) Threonine	AAC	(Asn/N) Asparagine	AGC	(Ser/S) Serine
		AUA	(Ile/I) Isoleucine	ACA	(Thr/T) Threonine	AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine
		AUG	(Met/M) Methionine	ACG	(Thr/T) Threonine	AAG	(Lys/K) Lysine	AGG	(Arg/R) Arginine
	G	GUU	(Val/V) Valine	GCU	(Ala/A) Alanine	GAU	(Asp/D) Aspartic acid	GGU	(Gly/G) Glycine
		GUC	(Val/V) Valine	GCC	(Ala/A) Alanine	GAC	(Asp/D) Aspartic acid	GGC	(Gly/G) Glycine
		GUA	(Val/V) Valine	GCA	(Ala/A) Alanine	GAA	(Glu/E) Glutamic acid	GGA	(Gly/G) Glycine
		GUG	(Val/V) Valine	GCG	(Ala/A) Alanine	GAG	(Glu/E) Glutamic acid	GGG	(Gly/G) Glycine

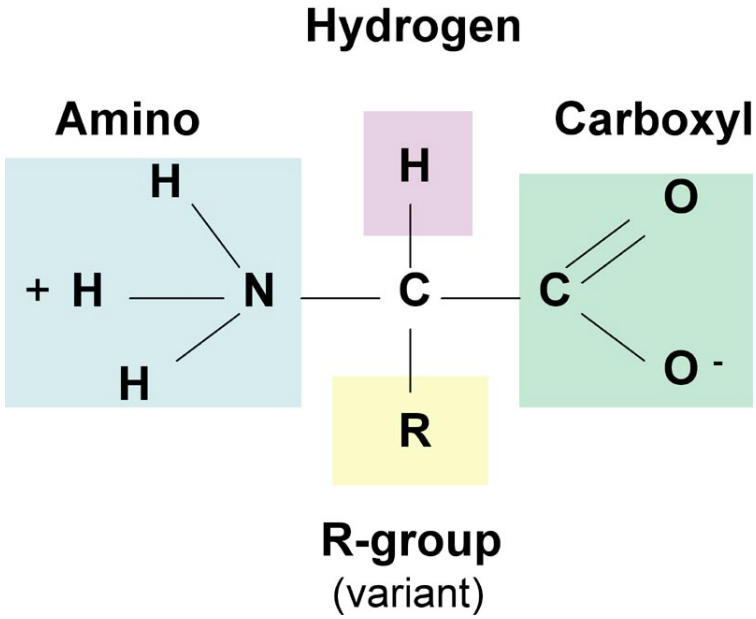


Slika 2.1 - apstraktna ilustracija replikacije DNK niza

2.2 O proteinima

U ovom odjeljku daju se osnovne informacije o građi i funkciji proteina (bjelančevina), kako bi se olakšalo razumijevanje problema i terminologija opisanih kasnije u radu.

Proteini su dugi lanci aminokiselina. U kemijskom smislu, aminokiseline su molekule koje sadrže centralni ugljikov atom (C_α), amino grupu (NH2), karboksilnu grupu (COOH) i bočni ogranak(R). Slijed amino grupa, ugljikovih atoma C_α i i karboksilnih grupa naziva se **okosnica** proteina (eng. *backbone*). Razlika između dvije aminokiseline je u sastavu i strukturi bočnog ogranka. Od dvadeset aminokiselina, devetnaest ih ima strukturu prikazanu na slici 2.2 (aminokiselina prolin ima vezu između amino grupe i bočnog ogranka).



Slika 2.2-općenita struktura aminokiselina

Proteini se međusobno razlikuju jedino po broju aminokiselina koje ih tvore, te po redoslijedu u kojem se pojavljuju. Vezane aminokiseline nazivaju se *rezidue* ili *peptidi*. Dvije vezane aminokiseline nazivaju se *dipeptid*, a u slučaju da se radi o više o dvije aminokiselina, govorimo o *polipeptidu.* Duži lanci nazivaju se *polipeptidni lanci.*

Govoreći o strukturi proteina, biokemija razlikuje četiri različite razine strukture proteina, ilustrirane na slici 2.3:

- *Primarna struktura*

  Odnosi se na niz aminokiselina u polipeptidnom lancu (lancu, slijedu aminokiselina).

- *Sekundarna struktura*

  Odnosi se na ponavljajuće lokalne strukture u proteinu, učvršćene vodikovim vezama. Najčešće su: α-uzvojnica (eng. *α-helix),* β-nabrana ploča (eng. *β-sheet),* okret (eng. *turn).*

- *Tercijarna struktura*

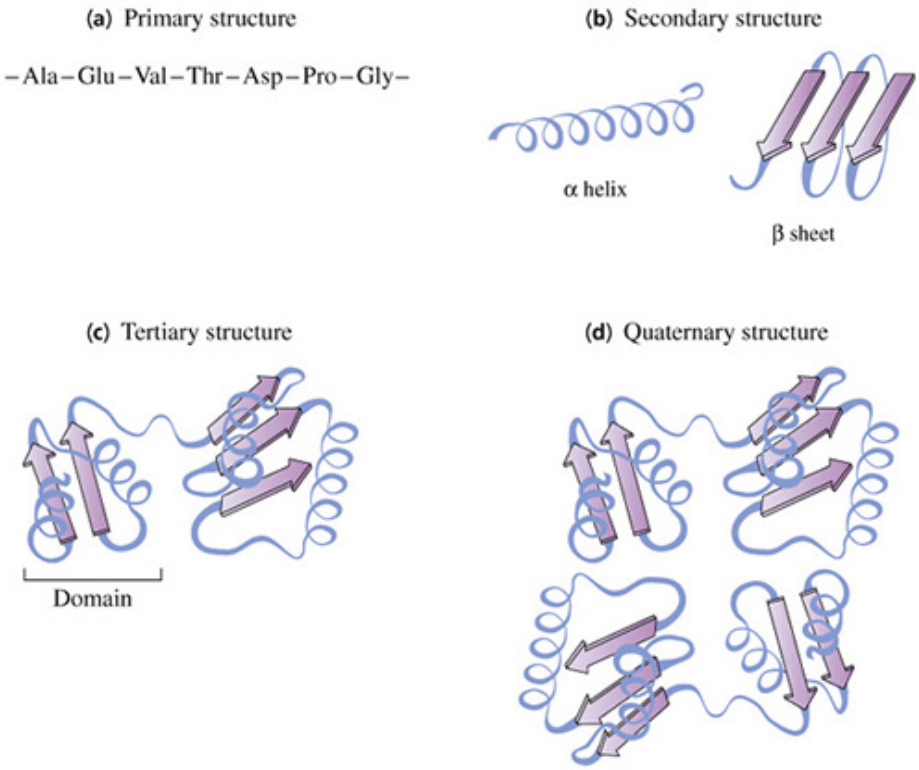
  Trodimenzionalna struktura čitavog polipeptidnog lanca.[8]; sveukupni oblik proteina; međusobni prostorni odnos sekundarnih struktura u proteinu.

- *Kvartarna struktura*

  Odnosi se na strukture formirane od više polipeptidnih lanaca, koje u ovom kontekstu nazivamo proteinske podjedinice, koje funkcioniraju kao jedan protein (proteinski kompleks).

Protein Data Bank (PDB) je baza podataka u kojoj se nalaze 3D strukturni podaci za biološke makromolekule, uglavnom dobiveni rentgenskom kristalografijom (eng. *X-ray crystallography)* i nuklearnom magnetskom rezonancom. 28. lipnja 2011. u bazi se nalazilo 68660 proteina[9], od toga 17036 ljudskih proteina.

Na web adresi američkog *National Centre for Biotechnology Information* [10] nalazi se baza podataka proteina (i dr.) iz koje se mogu saznati razne informacije o željenim proteinima, npr. slijed aminokiselina.



Slika 2.3 - razine strukture proteina

2.3 Glavni problemi bioinformatike

U sljedećih nekoliko odlomaka navode se i ukratko opisuju najčešći i najvažniji problemi, na rješavanje kojih se primjenjuju metode bioinformatike. Problem poravnavanja proteina, kao centralna tema rada, detaljnije je analiziran u poglavlju 4.

2.3.1 Sastavljanje genoma

Ljudski genom sastoji se od 3.3 milijarde baznih parova nukleotida, organiziranih u 23 para kromosoma veličine do nekoliko stotina milijuna baznih parova. Radi ilustracije, naivno pretpostavimo da bi smo mogli čitati baze ljudskog genoma brzinom od jedne baze u sekundi. Radeći osam sati na dan, dvjesto dana u godini, trebalo bi nam 572 godine kako bismo pročitali cijeli genom. Očigledno, postoje načini učinkovitiji od sekvencijalnog čitanja baza. Danas, brzo se sekvenciranje zasniva na masovno paralelnoj obradi fragmenata molekula DNK, uz neizbježnu

uporabu računalne snage modernih računala. Nadalje, razvoj "pametnih" algoritama, koji su u stanju sekvencirane fragmente sastaviti u njihov točan redoslijed, ključan je moment ovih napora. [5]

2.3.2 Pronalaženje gena

Nakon sekvenciranja, pronalaženje funkcionalnih dijelova DNK niza sljedeći je važan zadatak za bioinformatiku; primarno gena koji kodiraju proteine, a osim toga i RNK gene(nekodirajući RNK) te regulacijske regije genoma. Dijelovi gena koji kodiraju informaciju zovu se eksoni, a nekodirajući dijelovi nazivaju se introni. Kod DNK->mRNK transkripcije, prepisuju se i eksoni i introni, međutim introni se uklanjaju iz mRNK prije translacije u protein.[11] U DNK nizu također ističemo regije koje nazivamo promotori. To su regulatorni elementi koji se nalaze u blizini gena koje reguliraju, a funkcija im je označavanje početnog nukleotida za transkripciju u mRNK.

Zbog nedostatka znanja o tome kako precizno prepoznati eksone u genu, ili promotore gena, ne postoje deterministički sustavi koji vrše željenu funkciju, već se koriste statističke metode ili adaptivni sustavi poput neuronskih mreža. Takvi sustavi mogu (više ili manje) naučiti dobra pravila za karakterizaciju eksona iz dovoljnog broja poznatih eksona.[5]

2.3.3 Dizajn lijekova

Dizajn lijekova je proces pronalaženja novih medikamenata temeljen na znanju o biološkoj meti.[12] Lijek je većinom mala molekula(nije polimer)[13] koja aktivira ili inhibira funkcije ciljane biomolekule, npr. proteina, s rezultatom koji je terapeutski koristan za pacijenta. Ukratko, takve molekule se dizajniraju na način da budu komplementarne obliku i naboju svoje biološke mete, kako bi se na istu vezale, tj. kako bi imale afinitet vezanja s metom. Tehnike predviđanja afiniteta su prilično uspješne, međutim postoji niz drugih faktora koji se moraju uzeti u obzir pri dizajniranju. Neki od njih su biodostupnost (eng. *bioavailability)*[14], poluvrijeme izlučivanja (vrijeme potrebno da supstanca izgubi pola svoje farmakološke, fiziološko ili radiološke aktivnosti) [15], te nedostatak nuspojava.

Računala se u ovom području koriste u više faza:

- identifikacija ili pronalazak potencijalnih malih molekula s odgovarajućim afinitetom[16]

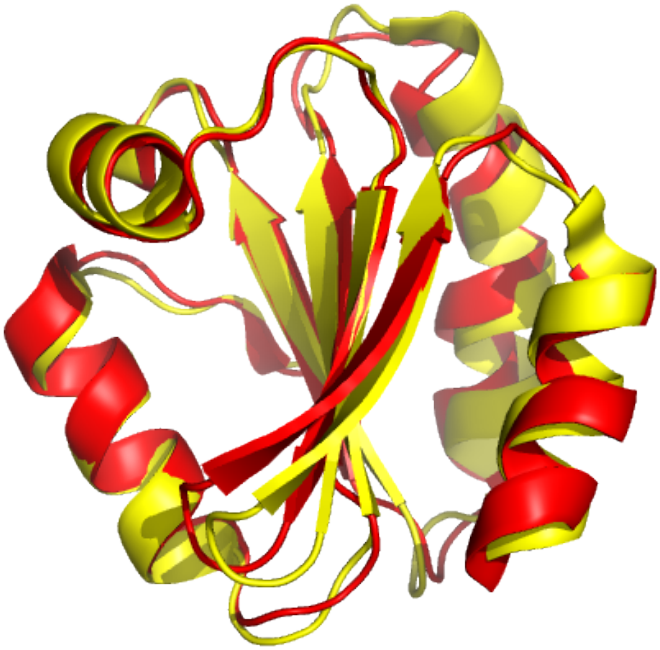
- optimizacija afiniteta kandidata
- optimizacija drugih farmakoloških svojstava uz zadržavanje afiniteta

2.3.4 Strukturno poravnavanje

Strukturno poravnavanje pokušava utvrditi homologiju između dva ili više polimera, na temelju njihovih trodimenzionalnih oblika. Nakon niza odgovarajućih rotacija i/ili translacija, moguće je superponirati jedan polimer na drugi.

Ovo je posebno važna tehnika za proteine i razumijevanje njihovih nizova, strukture, funkcije i evolucije, posebice stoga što je, u procesu evolucije, trodimenzionalna struktura proteina manje podložna promjenama od samog niza aminokiselina u molekuli proteina.[1] Iz tog je razloga strukturna usporedba u prednosti nad usporedbama nizova kod identifikacije evolucijski udaljenijih proteina, međutim kod interpretacije rezultata za takve proteine valja biti oprezan, budući da je moguće da se radi o konvergentnoj evoluciji[17], pojavi pri kojoj više nepovezanih proteina (npr. kod nepovezanih organizama) teži k istoj ili sličnoj tercijarnoj strukturi.

Također, biološka funkcija proteina izravno je ovisna o njegovoj trodimenzionalnoj strukturi, što znači da poravnavajući protein nepoznate funkcije s proteinom kojem je funkcija poznata, možemo doći do zaključaka o funkciji nepoznatog proteina.[1] Budući da je struktura proteina direktno ovisna o pripadajućem slijedu aminokiselina, zaključujemo kako je funkcija proteina također izravno ovisna o slijedu proteina.[18] Zašto je to važno? Precizna struktura proteina određuje se laboratorijski putem rentgenske kristalografije ili putem nuklearne magnetske rezonance, no ti su procesi vremenski i financijski zahtjevni. Algoritamsko predviđanje strukture (opisano u sljedećem odjeljku) je računalno iznimno težak zadatak, stoga je strukturno poravnavanje korisna metoda kod određivanja funkcije proteina. Primjer vizualiziranog poravnavanja vidljiv je na slici 2.4:



Slika 2.4-Strukturno poravnanje tierodoksina u ljudima i vinskoj mušici (Drosophila melanogaster)

Jedan od programa za strukturno poravnanje, imena STRAP, dostupan je na web adresi [19]. Popis programa može se pronaći na web adresi [20].

2.3.5 Predviđanje strukture proteina

Predviđanje strukture proteina je predviđanje trodimenzionalne sekundarne, tercijarne i kvartarne strukture iz primarne strukture proteina.

Kako zbog svoje važnosti, tako i zbog težine problema, kvalitetno predviđanje tercijarne strukture proteina spada u skup *Svetih Gralova* bioinformatike.[5] Zbog već spomenute cijene i trajanja laboratorijskih procesa određivanja strukture, očekivano je da postoji potražnja za dobrim računalnim metodama. Modeliranjem svih sila koje utječu na savijanje proteina, javlja se potreba za računalnim simulacijama kompletnog procesa savijanja, počevši od slijeda aminokiselina koje protein čine. Međutim, zbog goleme kompleksnosti takvih simulacija, do danas su takve simulacije uspjele simulirati savijanje samo za iznimno kratke vremenske intervale.

Metoda je od velike važnosti za medicinu (dizajn lijekova) i biotehnologiju (npr. dizajn novih enzima). Svake dvije godine, dosezi trenutnih metoda se ocjenjuju u CASP (Critical Assessment of Techniques for Protein Structure Prediction) eksperimentu[21].

Predviđanje sekundarne strukture proteina zasada ima točnost od oko 80%.[22] Predviđanje tercijarne strukture još se uvijek smatra neriješenim problemom, između ostalog zato što je prostor mogućih struktura za svaki protein astronomske veličine (npr. reda veličine 10^{143} mogućih prostornih konformacija[23]).

Prema CASP eksperimentu, u periodu 2006.-2010. godine, najbolji je program I_TASSER, a popis programa s istom svrhom može se pronaći na web adresi [22].

3 O genetskim algoritmima

U ovom se poglavlju daje uvod u genetske algoritme, opisuje se njihova najčešća struktura i izvedba, genetski operatori, načini selekcije, parametri, relevantna terminologija te mogućnosti primjene.

3.1 Uvod

Genetski algoritmi (eng. Genetic Algorithms; GA) pripadaju skupini algoritama **evolucijskog računarstva**, koje je potpodručje umjetne inteligencije. Radi se o (meta)heurističkim algoritmima, koji se služe nekima od evolucijskih principa koji su prisutni u prirodi. Spomenuti evolucijski principi su: nasljeđivanje, reprodukcija, križanje, mutacija, selekcija (preživljavanje najboljih). Ovi se algoritmi mogu koristiti za rješavanje širokog spektra problema, jer su prilično općeniti. U načelu, to su optimizacijski algoritmi. "Klasični" adaptivni sustavi su djelotvorni ako optimiziraju prostor stanja s jednim optimumom (tj. jednim ekstremom — maksimumom ili minimumom — parametra kojeg želimo optimizirati). Problemi se javljaju kod prostora s više lokalnih optimuma, jer se može dogoditi da algoritam "zapne" u nekom od njih, koji može biti prilično udaljen od globalnog. Genetski algoritam pretražuje stohastički, ali usmjereno, razne dijelove prostora pretrage i stoga postoji veća vjerojatnost da će doći do globalnog optimuma. Prostor pretrage može biti n-dimenzionalan. Pojam genetski algoritam se odnosi na model koji je uveo John Holland. Veliki dio teorije genetskih algoritama se odnosi upravo na taj model, kojeg bismo mogli nazvati kanonskim genetskim algoritmom. Metaforička usporedba genetskih algoritama (i evolucijskih algoritama općenito) s prirodnim sustavima dana je u tablici 3.1:

Tablica 3.1

GENETSKI (EVOLUCIJSKI) ALGORITAM	PRIRODA
Problem za optimizaciju	Okoliš
Potencijalna rješenja	Živući organizmi u okolišu
Kvaliteta rješenja (funkcija dobrote)	Stupanj prilagođenosti jedinke okolišu

Potencijalni problem genetskih algoritama je nedostatak matematički rigoroznih dokaza koji bi formalno dokazali njihovu ispravnost. Međutim, potreba za formalizacijom je diskutabilna - obično se kvaliteta neke varijacije genetskog algoritma ocjenjuje empirijski, tako što se eksperimentalno testira na nekom problemu optimizacije.

Ideja evolucijskog računarstva pojavila se u 1960-ima, zahvaljujući I. Rechenbergu i njegovom radu *"Evolutionsstrategie",* a drugi istraživači su nastavili rad na području. Genetske algoritme izumili su i razvili John Holland, njegove kolege i studenti, što je rezultiralo Hollandovom knjigom *"Adaptation in Natural and Artificial Systems",* 1975. godine. 1992. godine John R. Koza iskoristio je ideju GA za evoluciju izvršivih računalnih programa, metodu je nazvao *genetsko programiranje,* a svoje je ideje izrazio u knjizi *"Genetic Programming: On the Programming of Computers by Means of Natural Selection".[24]*

GA svoju primjenu nalaze u bioinformatici, filogenetici, ekonomiji, kemiji, matematici, fizici itd.

Kod dizajniranja genetskog algoritma potrebno je algoritam i pripadajuće parametre prilagoditi problemu. Najvažniji zadaci u tom koraku su dobar odabir strukture jedinke – ta struktura mora što vjernije predstavljati prostor rješenja; također genetski operatori moraju biti prilagođeni toj strukturi. Također je krucijalno odabrati dobru funkciju dobrote, mjeru kvalitete rješenja, jer je to kriterij selekcije koji kontrolira, usmjerava proces.

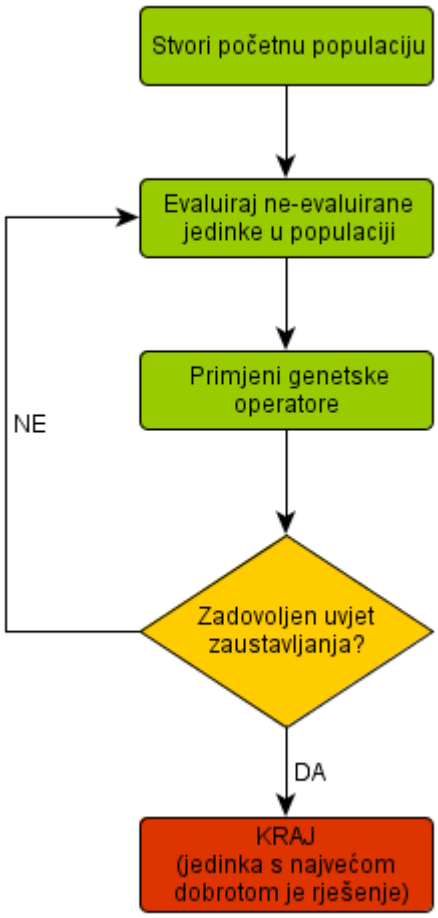
Nakon postavljanja početnih uvjeta, (nažalost) veliki utjecaj na ishod algoritma imaju i parametri kao što su vjerojatnost križanja, vjerojatnost mutacije, veličina populacije, broj generacija. Dobri intervali za navedene vrijednosti mogu se utvrđivati eksperimentalno, međutim valja uzeti u obzir relativnu sporost izvođenja GA.

3.2 Izvođenje

Izvođenje genetskog algoritma može se prikazati pseudokodom 3.1 [25].

Pseudokod 3.1

```
pocetak
      stvori populaciju P(0)
      ponavljaj
            odaberi P(t) iz P(t-1)
            primjeni genetske operatore na P(t)
      dok nije zadovoljen uvjet zaustavljanja
kraj
```



Slika 3.1 - dijagram toka izvođenja genetskog algoritma

3.2.1 Parametri genetskog algoritma

Važni parametri, s utjecajem na kvalitetu rješenja, vrijeme izvođenja, općenito učinkovitost su:

- *struktura i veličina jedinke*
- *veličina populacije*
- *broj generacija*
- *postotak eliminacije (ovisno o operatoru selekcije)*
- *veličina turnira (kod turnirske selekcije)*
- *vjerojatnost križanja*
- *vjerojatnost mutacije*

3.3 Jedinka i populacija

Početak svakog genetskog algoritma znači stvaranje početne populacije jedinki – rješenja kandidata. Kako bi taj zadatak bio moguć, prvo valja definirati strukturu jedinke. Najčešća struktura GA jedinke (jedinka se također ponegdje naziva **kromosom, genotip, genom**) jest binarni niz od *n* znamenaka. Slijedi primjer dvije 8-bitne jedinke:

00101001

10011011

Kromosom se može sastojati od više **gena**, kao što je prikazano primjerom sljedeće jedinke:

01001101 101110101

gen1 *gen2*

Prethodno se može interpretirati na sljedeći način: kromosom od jednog gena, iz prvog primjera, može predstavljati jednodimenzionalni prostor, npr, interval [0,255]. U drugom primjeru, kromosom s dva gena, nazovimo ih *x* i *y*, može predstavljati dvodimenzionalni prostor, npr. x ϵ [0,255], *y* ϵ [0,511] (gen y ima 9 bitova). Naravno, n-bitni prostor može se preslikati u bilo koji drugi prostor, a razlučivost prikaza tog prostora ovisi o veličini tog prostora, i broju bitova gena.

Decimalnu vrijednost binarnog niza nazovimo s *b***,** donju granicu željenog prostora s *dg*, gornju granicu *gg*; tada svaki binarni niz možemo preslikati u taj prostor jednadžbom 3.1:

$$x = \mathrm{dg} + \frac{b}{2^n + 1} * (\mathrm{gg} - \mathrm{dg}) \tag{3.1}$$

Obrnuto, preslikavanje iz prostora rješenja u decimalnu vrijednost binarnog niza, izvodi se jednadžbom 3.2:

$$\mathrm{b} = \frac{\mathrm{x} - \mathrm{dg}}{\mathrm{gg} - \mathrm{dg}} * (2^n - 1) \tag{3.2}$$

Generiranje početne populacije uglavnom se odvija slučajnim procesom, koji bi *trebao* stvoriti dovoljno raznoliku populaciju za daljnje primjene evolucijskih operatora.

3.4 Funkcija dobrote

U biološkim okruženjima, između organizama postoji čitav niz varijacija koje ih čine više ili manje prilagođenima preživljavanju i reproduciranju u tim staništima. Na sličan način modeliramo i genetske algoritme, kako bi bolja rješenja imale veće šanse za "preživljavanje", te na taj način usmjeravala pretragu koja se odvija. Šansu jedinke za prijenos u sljedeću generaciju, ili za prijenos dijela svog genetskog kôda u sljedeću generaciju (križanjem), nazivamo dobrotom (eng. *fitness*), a modeliramo je funkcijom dobrote (eng. *fitness function*).

Funkcija dobrote je mjera kvalitete pojedine jedinke (rješenja). Tom funkcijom GA vrši evaluaciju svake jedinke u populaciji. Ispravno definirana, od presudne je važnosti u genetskom algoritmu jer vrši *selekcijski pritisak* na populaciju, tj. usmjerava je k boljim rješenjima. Funkcija mora u potpunosti ili blisko korelirati s ciljem algoritma, a što brža izračunljivost je zahtjev koji je nužan zbog prirode algoritma – GA se tipično izvodi u velikom broju iteracija nad mnogobrojnom populacijom. Zbog toga se, za kompleksnije ciljne funkcije, kao funkcija dobrote često koristi aproksimacija funkcije cilja, koja ima smanjenu kompleksnost, tj. na neki je način optimizirana u svrhu bržeg izvođenja. Potrebno je funkciju dobrote uravnotežiti kako bi u što većoj mjeri zadovoljavala oba, uglavnom oprečna zahtjeva.

3.5 Genetski operatori

Genetski operatori simuliraju dio prirodnog evolucijskog procesa, u smislu da u populaciju unose varijacije, dobre ili loše, što se očituje pri selekciji, ili služe očuvanju ili kombiniranju kvalitetnog genetskog materijala.

U genetskim algoritmima najčešće su prisutni operatori reprodukcije (eng. *reproduction*), križanja (eng. *crossover, recombination*) te mutacije (eng. *mutation*).

U nekim varijantama GA postoje i drugi operatori, npr. permutacija (eng. *permutation*), inverzija (eng. *inversion*), regrupiranje (eng. *regrouping*), migracije (eng. *migrations*) itd. [26]

3.5.1 Reprodukcija

Također nazivan *aseksualnom reprodukcijom*, ovaj operator izvršava jednostavno kopiranje odabrane jedinke u sljedeću generaciju. Koja će jedinka "imati tu sreću", ovisi o odabranoj metodi selekcije, te, naravno, o funkciji dobrote. Općenito, očito je da jedinke s većom dobrotom imaju više šanse da prežive do sljedeće generacije. Očekivana posljedica ovog operatora je povećanje prosječne dobrote populacije u sljedećoj generaciji.

3.5.2 Križanje

Temeljeno na ideji iz biologije, ovim binarnim operatorom dvije jedinke izmjenjuju, točnije kombiniraju dijelove svog genetskog koda kako bi se proizveli potomci. Pretpostavljajući niz znakova kao reprezentaciju jedinke, u genetskim algoritmima križanje se obično odvija na jedan od tri načina (prikazanih na slici):

- *križanje s jednom točkom prekida (eng. one-point crossover)*

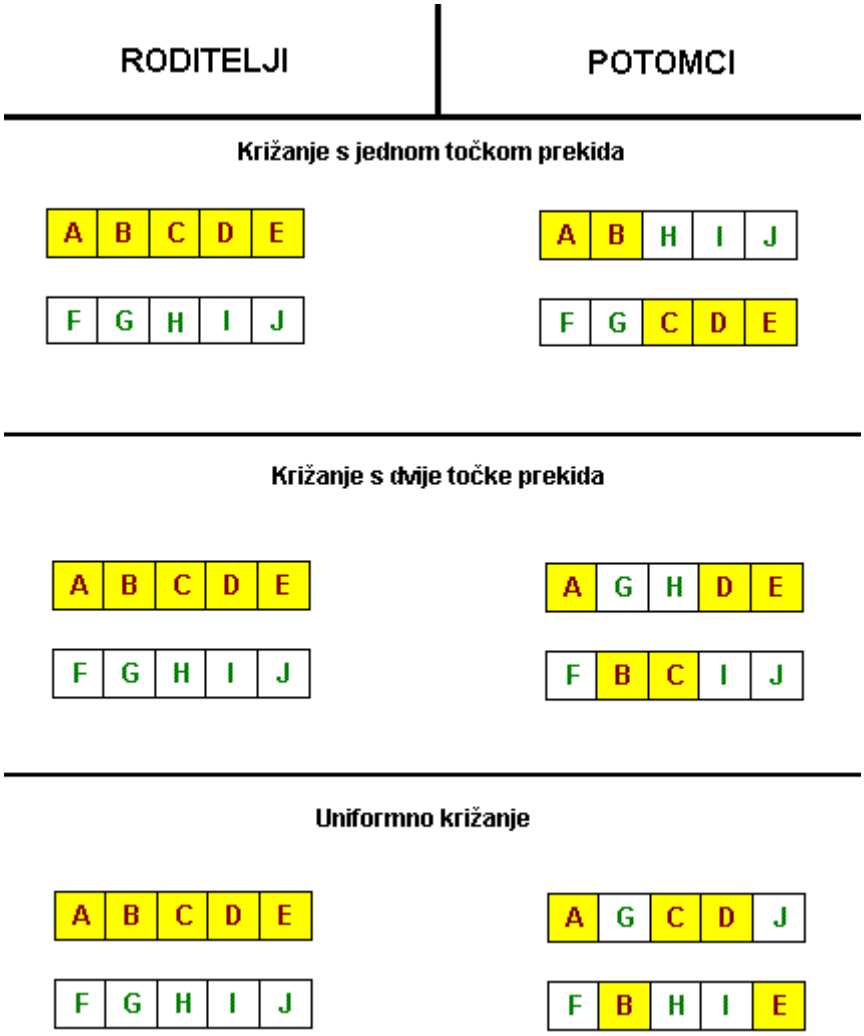
  Definira se jedna točka prekida, na istom mjestu u oba roditelja. Svaki roditelj je podijeljen na lijevi i desni dio, te se potomci stvaraju kombinacijom tih dijelova. Moguće je stvoriti dva različita potomka.

- *križanje s dvije točke prekida (eng. two-point crossover)*

  Definiraju se dvije točke prekida, i sav se genetski materijal između točaka zamijeni između roditelja; rezultat su dva potomka.

- *uniformno križanje (eng. uniform crossover)*

  Na razini svakog bita roditelja, slučajnim odabirom se određuje bit kojeg roditelja će činiti potomka na odgovarajućem mjestu. Drugi potomak je komplement prvog (na svakom mjestu sadrži bit od roditelja različitog nego u prvom potomku).



Slika 3.2-varijante operatora križanja

Parametar genetskog algoritma kojim kontroliramo vjerojatnost križanja najčešće označavamo s p_c.

3.5.3 Mutacija

Mutacija je slučajna promjena na dijelu genetskog koda. U prirodi, takve promjene mogu dovesti do bolesti, deformacija, smrti, međutim ponekad rezultiraju pozitivnim rezultatima, u smislu nove funkcionalnosti, ili bolje funkcionalnosti postojećih dijelova organizma. Mutacija je poremećaj genetskog materijala, i kao takva zapravo unosi novi genetski materijal, ne samo u jedinku, već u populaciju.

U kontekstu genetskih algoritama, mutacija je unarni operator koji vrši promjenu na slučajno odabranom dijelu genetskog koda. Najjednostavniji slučaj je promjena na jednom bitu. Tada definiramo vjerojatnost mutacije jednog bita, p_m (obično 0.001-0.01). Uz to možemo definirati i vjerojatnost mutacije kromosoma p_M, u ovisnosti o p_m. Ta je ovisnost definirana jednadžbom 3.3.

$$p_M = 1 - (1 - p_m)^n \tag{3.3}$$

Važnost mutacije u GA očituje se u:

- *ako algoritam zapne u okruženju lokalnog optimuma, mutacija može proizvesti rješenje izvan tog okruženja*
- *ako s vremenom dođe do gubitka kvalitetnog genetskog materijala, postoji šansa da će on biti vraćen mutacijom*
- *mutacija može stvoriti nova, bolja rješenja i tako ubrzati pretragu*

Potrebno je napomenuti da veća vjerojatnost mutacije usporava algoritam, a kako se vjerojatnost mutacije bliži k 1, tako algoritam konvergira k običnoj slučajnoj pretrazi.

3.6 Selekcija

Operator selekcije i njegova primjena služi kao mehanizam poboljšavanja kvalitete rješenja u populaciji u toku vremena, usmjeravajući na taj način pretragu prostora stanja bliže idealnom rješenju. Kažemo da se vrši selekcijski pritisak. Naravno, bilo koja varijanta selekcijskog postupka kao svrhu ima odumiranje nekvalitetnih, a preživljavanje kvalitetnih rješenja, jedinki. Operatore selekcije možemo podijeliti u dvije glavne kategorije, ovisno o načinu stvaranja sljedeće generacije:

- *generacijski*

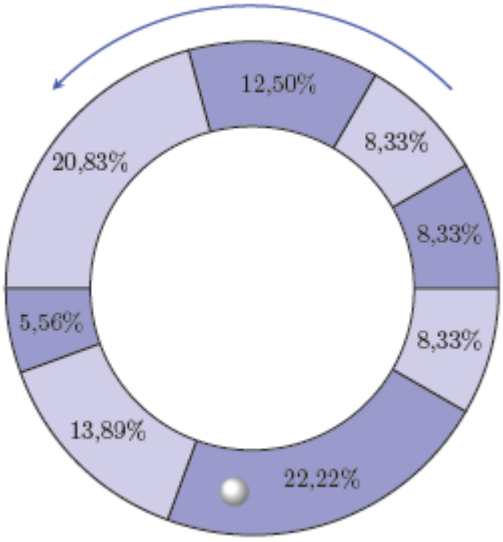
  Na temelju jedinki u trenutnoj populaciji stvara se sljedeća generacija, pomoću genetskih operatora.

- *eliminacijski*

  Određeni dio populacije se eliminira, a nadomješta se iz preostalih jedinki uporabom operatora.

3.6.1 Generacijski jednostavni odabir

Proporcionalni odabir (eng. *roulette wheel selection*) je svojevrsna simulacija kotača ruleta, na kojem svaka jedinka zauzima dio kotača proporcionalno svojoj dobroti. Jedinka se bira (tj. "kotač se vrti") onoliko puta kolika je veličina populacije, kako bi se generirala nova generacija. Slika 3.3 služi za ilustraciju.[27]



Slika 3.3 - "roulette wheel selection"

Bolje jedinke imaju vjerojatnost da budu izabrane više puta, što je nedostatak u slučaju da postoji jedinka sa značajno većom dobrotom od ostalih, jer se populacija lako može zasititi kopijama jedne jedinke. Također, za ovaj odabir dobrota mora biti pozitivna.

3.6.2 Eliminacijski jednostavni odabir

Kao i kod generacijskog jednostavnog odabira, ovaj operator bira jedinke na jednak način, međutim, kako se radi o eliminacijskom odabiru, izgubljene jedinke nadomještamo genetskim operatorima na preostaloj populaciji. Također, želimo da lošije jedinke imaju veću vjerojatnost odabira. Stoga, definiramo *mjeru nekvalitete*, obrnuto proporcionalnu mjeri dobrote.

Parametar kod ovog operatora je postotak eliminacije (broj jedinki koji se eliminira).

3.6.3 Turnirski odabir

Turnirski operator može biti i generacijski i eliminacijski, no za uporabu se preporuča eliminacijski.

Generacijski turnirski operator slučajno bira *k* jedinki, te najbolju prenosi u sljedeću generaciju.

Eliminacijski turnirski operator slučajno bira *k* jedinki, eliminira najlošiju te je zamjenjuje novom uporabom genetskih operatora. Turnirski odabir s *k* jedinki zovemo *k-turnirski odabir.* Pseudokod 3.2 prikazuje općeniti GA s k-turnirskim odabirom. Turnirski odabir pokazao se jednostavnim za implementaciju i prigodnim za paralelizaciju.[25]

Pseudokod 3.2

```
pocetak (GA s k-turnirskim odabirom)
      stvori populaciju P(0)
      ponavljaj
            odaberi k jedinki iz populacije
            pronađi i obrisi najlosiju od k odabranih
            generiraj novu jedinku pomoću genetskih
operatora
      dok nije zadovoljen uvjet zaustavljanja
```

3.6.4 Elitizam

Elitizam u genetskim algoritmima je svojstvo očuvanja, tj. "imuniteta" od eliminacije najbolje jedinke u svakoj generaciji. Svojstvo se pokazalo korisnim u sprječavanju opadanja kvalitete najbolje jedinke.[25]

Elitizam je implicitno ugrađen u eliminacijske odabire.

3.7 Uvjeti zaustavljanja algoritma

Neki od mogućih kriterija zaustavljanja algoritma:

- *broj generacija*
- *broj evaluacija funkcije dobrote*
- *dosegnuta vrijednost funkcije dobrote*
- *stagnacija (određen broj generacija bez poboljšanja vrijednosti dobrote)*
- *vremensko ograničenje*

4 Problem višestrukog poravnavanja proteinskih nizova

U ovom poglavlju predstavlja se poravnavanje bioloških nizova na općenit način, s nešto detaljnijim osvrtom na višestruko poravnavanje nizova (eng. *Multiple Sequence Alignment*), a u žarištu pozornosti se nalazi višestruko poravnavanje proteina. Nizovi ciljani ovim metodama su DNK, RNK i proteini. Opisat će se i neka postojeća rješenja koja su poslužila kao inspiracija za praktični dio ovog rada, ali i kao konkurencija za usporedbu.

4.1 Poravnavanje nizova

Postoje tri razloga zašto je egzaktno uspoređivanje nizova koji predstavljaju biološke molekule od sekundarne važnosti za bioinformatiku. Prvo, moramo očekivati da mjerenja obavljena u laboratorijima ponekad nisu u potpunosti pouzdana. Drugo, poznato je da je proces prenošenja podataka u računalo također podložan greškama uzrokovanim ljudskim faktorom. Treće, i najbitnije, sama priroda je izvor neidentičnih, ali ipak homolognih nizova, u smislu da je evolucija mutirala baze, unosila nove baze, ili brisala postojeće. Zbog toga, preporučljivo je da se pri uspoređivanju bioloških nizova orijentiramo na "mekše" kriterije. [5] Uvodimo pojam **poravnavanja nizova**. Promotrimo dva DNK niza:

```
TTATGCATACCTCATGGGTACT

TTACGCGTACTCATGGTACTT
```

Nakon poravnavanja, oni bi mogli izgledati ovako:

```
TTATGCATAC-C-TCATGGGTACT

TTACGCGTACTCATGGTAC-T--T
```

Ovdje smo uveli najmanji mogući broj praznina, kako bi se najveći mogući broj istovjetnih znakova u nizovima našao na istim pozicijama. Valja uočiti kako se uvođenje praznine u neki od nizova može protumačiti kao brisanje prethodno postojećeg znaka u nizu, ili kao uvođenje novog

znaka u drugi niz. Pitanje koje se postavlja jest kako precizno mjeriti kvalitetu poravnavanja. Ovdje se moramo oslanjati na dogovore o bodovanju identiteta, mutacija, umetanja i brisanja.

Takvi dogovori formalizirani su u obliku supstitucijskih matrica (eng. *substitution matrix*) za bodovanje poravnavanja. Matrice su veličine 20x20 i za svaku kombinaciju među 20 aminokiselina izražavaju brojčanu ocjenu. Ocjenjivanje dvostrukog, parnog poravnavanja (dva niza), primjer kojeg je prikazan na slici, izvodi se tako da se za svaku poziciju iz odgovarajuće matrice pročita ocjena, te se zbroje ocjene za svaku poziciju.

Postoje glavne tri skupine matrica za ocjenjivanje poravnavanja proteina:

- *PAM matrice*

  PAM matrice bazirane su na 1572 promotrene mutacije na 71 obitelji blisko povezanih proteina, brojčane vrijednosti u matrici predstavljaju vjerojatnost zamjene jedne aminokiseline drugom. PAM matrice su normalizirane – PAM1 matrica daje vjerojatnosti zamjene za nizove koji su doživjeli jednu mutaciju na svakih 100 aminokiselina. Mutacije se mogu preklapati, tako da PAM250 matrica opisuje vjerojatnosti zamjene za nizove koji su doživjeli 250 mutacija na 100 aminokiselina, ali kao posljedica preklapanja, samo 80 od 100 aminokiselina je mutirano.

- *BLOSUM matrice*

  Baziraju se na promatranju gotovih poravnanja iz BLOCKS baze podataka. Postoji nekoliko skupina BLOSUM matrica, ovisno o poravnavanjima na kojima se temelje. BLOSUM matrice s niskim brojem (npr. BLOSUM45) mogu se koristiti za ocjenjivanje poravnavanja evolucijski udaljenih nizova, a one s visokim brojem (npr. BLOSUM80) se koriste za evolucijski bliske nizove. Slika 4.2 prikazuje najkorišteniju, BLOSUM62 matricu[28].

- *Gonnet matrice*

  Gonnet je matrica zasnovana na cijeloj SwissProt[29] bazi podataka iz 1991. Matrica ima široku uporabu, ali unatoč tome je u određenoj mjeri selektivna, jer se u SwissProt bazi nalaze određene obitelji proteina.

```
AAB24882        TYHMCQFHCRYVNNHSGEKLYECNERSKAFSCPSHLQCHKRRQIGEKTHEHNQCGKAFPT 60
AAB24881        --------------------YECNQCGKAFAQHSSLKCHYRTHIGEKPYECNQCGKAFSK 40
                                    ****: .***:  * *:** * :****.:* *******..

AAB24882        PSHLQYHERTHTGEKPYECHQCGQAFKKCSLLQRHKRTHTGEKPYE-CNQCGKAFAQ- 116
AAB24881        HSHLQCHKRTHTGEKPYECNQCGKAFSQHGLLQRHKRTHTGEKPYMNVINMVKPLHNS 98
                 **** *:***********:***:**.: .***************     :  *.: :
```

Slika 4.1-primjer dvostrukog poravnavanja dva proteina (razdvojeno u 2 reda), dobivenog programom ClustalX

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Slika 4.2-BLOSUM62 matrica

Čest pristup u poravnavanju je tehnika dinamičkog programiranja. Uobičajeni algoritmi koji se koriste su *Needleman-Wunsch* te *Waterman-Smith* algoritmi. Uz supstitucijske matrice, obično se koristi i *kažnjavanje praznina*, kada se dogodi da se na mjestu aminokiseline u jednom proteinu, u drugom nalazi praznina. Umjesto linearnog modela kažnjavanja praznina, u kojem kazna za prazninu linearno ovisi o njezinoj veličini (broju praznina – u prikazu crtica), uglavnom se koristi različita kazna za otvaranje i širenje praznine, na način da je kazna za otvaranje praznina prilično veća od kazne za širenje, npr. -10 za otvaranje i – 0.2 za širenje. Funkcija kažnjavanja stoga izgleda kao što je prikazano izrazom 4.1, gdje je ukupna kazna - K_u - jednaka zbroju kazne za otvaranje praznine - K_o - i kazne za širenje(ekstenziju) praznine – K_e.

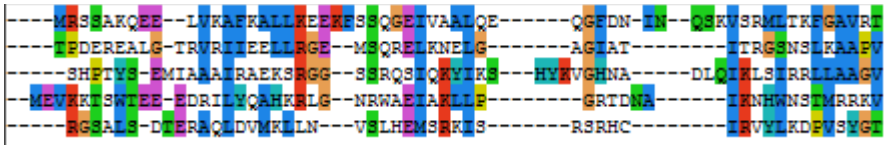
$$K_u = K_o + K_e \tag{4.1}$$

4.2 Višestruko poravnavanje nizova

U odnosu na opisano u prethodnom potpoglavlju, **višestruko poravnavanje nizova** (MSA), odnosi se na poravnavanje tri ili više nizova. U većini slučajeva, razmatraju se nizovi za koje se smatra da posjeduju određenu razinu evolucijske srodnosti, po kojoj dijele lozu ili zajedničkog pretka[30]. Iz rezultirajućeg poravnanja nadalje se može zaključivati o homolognosti nizova ili se izvršiti filogenetska analiza, kako bi se opisalo zajedničko evolucijsko porijeklo. Jedno višestruko poravnanje dostupno je na slici 4.3.

Višestruko poravnavanje se često koristi i za ispitivanje očuvanosti domena proteina, sekundarne i tercijarne strukture, pa čak i individualnih aminokiselina (ili nukleotida u slučaju DNK/RNK).

U usporedbi s dvostrukim poravnavanjem, višestruko poravnavanje je prilično računalno kompleksnije, te zahtijeva sofisticiranije algoritme. Većina postojećih programa koristi heurističke metode, jer deterministički algoritmi, već na više od nekoliko nizova, imaju preveliku cijenu u smislu računalnog vremena. Zašto je navedeni problem težak? Višestrukim poravnavanjem pokušavamo utvrditi evolucijske, strukturne i funkcionalne povezanosti među nizovima koji su možda divergirali milijunima, katkad milijardama godina. Najbolji način za otkrivanje tih veza temelji se na dobrom poznavanju evolucijske povijesti i strukturnih svojstava nizova, međutim takvo je znanje, nažalost, vrlo rijetko dostupno. Umjesto toga, koriste se generički modeli evolucije proteina temeljeni na sličnosti nizova, uz sustav ocjenjivanja prikladnosti temeljenom na kažnjavanju supstitucija i umetanja praznina.



Slika 4.3-primjer višestrukog poravnanja pet proteina

4.3 Standardni algoritmi

Standardni optimizacijski algoritmi za višestruko poravnavanje mogu se podijeliti u tri kategorije: egzaktni, progresivni i iterativni.

Egzaktni algoritmi pokušavaju pronaći optimalno poravnanje temeljeno na dobro definiranim pravilima. Nažalost, takvi algoritmi su prilično ograničeni u pogledu broja nizova koje mogu obraditi u razumnom vremenu, te u pogledu funkcije dobrote (objektivne funkcije) koju optimiziraju.

Progresivni algoritmi su daleko najkorišteniji [1]. Oni se izvode progresivnim sastavljanjem višestrukog poravnanja, na način da se nizovi u poravnanje dodaju jedan po jedan, tako se nikada ne poravnavaju više od dva niza istovremeno. Dvostruka poravnavanja izvode se dinamičkim programiranjem (Needleman-Wunsch). Ovaj pristup ima veliku prednost zbog svoje jednostavnosti i brzine, ali u osnovi je *pohlepna* heuristika koja ne garantira optimizaciju, tj. može se dogoditi da se zaustavi u nekom lokalnom optimumu.

Iterativni algoritmi proizvedu inicijalno poravnavanje i zatim ga poboljšavaju kroz niz ponavljajućih koraka (iteracija), dok se više ne može postići poboljšanje. Iterativne metode mogu biti determinističke i stohastičke. Najjednostavnije metode su determinističke.

Determinističke metode uzimaju niz po niz iz poravnavanja i ponovno ga poravnavaju, dok se poravnanje više ne može poboljšati (konvergencija).

Stohastičke metode uključuju **Skrivene Markovljeve modele** (eng. *Hidden Markov Models; HMM),* **simulirano kaljenje**, **evolucijsko računarstvo (genetski algoritmi)**, **te evolucijsko programiranje (EP)**. Glavna prednost stohastičkih metoda je u jednostavnom razdvajanju optimizacijskog procesa i kriterija evaluacije (funkcije dobrote).

4.4 SAGA

Sequence Alignment by Genetic Algorithm najpoznatiji je i najkorišteniji program za višestruko poravnavanje proteina koji koristi evolucijsko računarstvo.[31][32]

Struktura jedinke je jednostavna dvodimenzionalna matrica u kojoj svaki red predstavlja jedan niz u poravnanju.

U SAGA-i, populacija ima konstantan broj i ne sadrži duplikate. Početna generacija sastoji se od 100 slučajno stvorenih jedinki. Jedinke se stvaraju na način da se za svaki niz u poravnanju odabere inicijalna početna praznina (eng. *offset*), duljine od nula do duljine najdužeg niza. Dakle, svaki niz pomaknut je u desno za broj mjesta dodijeljen posmakom. Razlike u duljinama nizova na krajevima popunjavaju se prazninama, tako da svaki niz u poravnanju ima istu duljinu.

Selekcija se obavlja proporcionalnim odabirom, i nad odabranom populacijom se primjenjuju operatori varijacije. SAGA koristi 22 različita operatora koji se i sami optimiziraju tijekom izvođenja korištenjem dinamičkog raspoređivanja (eng. *dynamic scheduling)*. Operatori se mogu podijeliti u dvije kategorije: jedno-roditeljski (mutacije) i više-roditeljski (križanja).

Funkcija dobrote koju koristi SAGA je težinska suma parova (eng. *weighted sum-of-pairs*), detaljnije opisana u sljedećem poglavlju.

Algoritam završava kada nije došlo do poboljšanja kroz 100 generacija.

5 Ostvarenje programskog rješenja

U članku "*A Simple Genetic Algorithm for Multiple Sequence Alignment*" [31], opisan je jednostavan genetski algoritam za višestruko poravnavanje nizova proteina. Članak dovodi u pitanje potrebu za velikim brojem kompleksnih operatora s dinamičkim raspoređivanjem, kao što je to slučaj u programu SAGA, opisanom u potpoglavlju 4.4 , te predlaže brojna pojednostavljenja. Ostvarenje praktičnog dijela ovog rada vođeno je idejama opisanima u navedenom članku.

5.1 ECF

Praktično ostvarenje rada sagrađeno je na *Evolutionary Computation Framework* -u (ECF), platformi za evolucijsko računarstvo razvijanu na matičnom fakultetu.[33] ECF je općenita platforma za primjenu bilo koje vrste evolucijskog računarstva. Sagrađena je u jeziku C++, uz korištenje C++ STL (eng. *Standard Template Library)* i Boost C++ biblioteka.

Dostupni, ugrađeni genotipovi uključuju:

- *za genetske algoritme:* bitstring, binarno kodirane realne vrijednosti, vektore realnih brojeva s pomičnom decimalnom točkom, permutacijske vektore
- *za genetsko programiranje:* stablo

Valja napomenuti da jedinka može sadržavati različite genotipove u bilo kojem broju. Ugrađeni algoritmi selekcije su:

- *turnirska selekcija*
- *generacijska jednostavna selekcija (generational roulette-wheel)*
- *optimizacija rojem čestica (eng. particle swarm optimization)*

Platforma također podržava paralelizaciju putem *Message Passing Interface* (MPI) API-ja.

Jedini parametar koji se **mora** definirati prije pokretanja je struktura jedinke, dok će ostali parametri koristiti predodređene vrijednosti, ukoliko nisu eksplicitno navedeni u konfiguracijskoj datoteci, koja se dohvaća i obrađuje pri inicijalizaciji programa. Konfiguracijska datoteka je u XML (eng. *Extensible Markup Language*) formatu, te je njen primjer dan na slici 5.1.

```
<ECF>
  <Algorithm>
    <SteadyStateTournament>
      <Entry key="tsize">3</Entry>
    </SteadyStateTournament>
  </Algorithm>

  <Genotype>
    <BitString>
      <Entry key="size">20</Entry>
    </BitString>
    <Tree>
      <Entry key="functionset">sin cos + - / *</Entry>
      <Entry key="terminalset">X Y</Entry>
    </Tree>
  </Genotype>

  <Registry>
    <Entry key="population.size">30</Entry>
    <Entry key="term.maxgen">100</Entry>
  </Registry>
</ECF>
```

Slika 5.1-osnovni primjer ECF konfiguracijske datoteke

U programskom rješenju ECF je nadgrađen vlastitim komponentama:

- *vlastiti genotip*
- *vlastiti operatori*
- *vlastita funkcija dobro*te

5.2 Prikaz jedinke

Jedinka je predstavljena dvodimenzionalnom matricom u kojoj svaki red predstavlja jedan niz u višestrukom poravnanju. Osim dvadeset simbola koji predstavljaju svaku od aminokiselina koje mogu graditi protein, postoji i simbol za prazninu: "-". Moguć izgled jedinke prikazan je na slici 5.2.

```
MEVKKTSWTE-EEDRILYQAHK--RLGNRWAEIAKLLPGRTDNAIKNHWNS---------TMRRKV----
---SHPTYSEMIAAAIRAEKSRG---GSSRQSIQKYIKSHYKVGHNADLQIKL-------SIRRLLAAGV
---RGSALS--DTERAQLDVMK--LLNVSLHEMSRKI-SRSRHCIRVYLKDP--VSYGT-----------
MRSSAKQEE---LVKAFKALLKEEKFSSQG-EIVAALQEQGFDNINQSKVSRMLTKFGAV--RT------
TPDEREALG--TRVRIIEELLRG-EMSQR--ELKNEL-GAGIATITRGSNS---LKAAPV----------
```

Slika 5.2-vizualizacija jedinke programskog rješenja

Jedinka je programski ostvarena kao nadgradnja ECF-a razredom `MSA`, koji nasljeđuje ECF-ov razred `Genotype`.

5.3 Stvaranje početne populacije

Svaka jedinka predstavlja jedno višestruko poravnavanje. Nakon pokretanja programa i učitavanja datoteke s neporavnatim nizovima u FASTA formatu[34], program gradi svaku jedinku na način prikazan u pseudokodu 5.1:

Pseudokod 5.1

```
kopiraj ulazne nizove u jedinku
za svaki niz jedinke
     na početak niza dodaj slučajno generirani pomak(praznine)
kraj
```

Veličina posmaka je slučajno generirani broj između nula i polovice duljine niza.

5.4 Operatori

U programsko rješenje ugrađeno je pet operatora: dva operatora križanja i tri operatora mutacije.

5.4.1 Operatori križanja

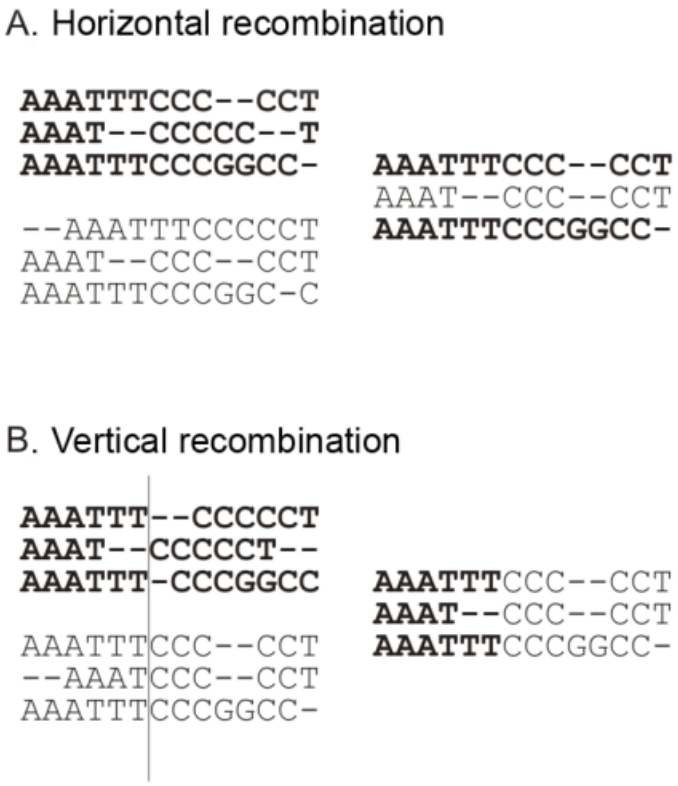
Potomci se stvaraju jednim od dva operatora križanja: *horizontalno križanje* i *vertikalno križanje*, kao što je prikazano na slici 5.3.

Horizontalno križanje nasumce odabire svaki niz iz jednog od roditelja i kopira ga u potomka.

Vertikalno križanje nasumce odabire točku prekida, istu kod oba roditelja, te se svi nizovi prije točke prekida potomka kopiraju iz jednog roditelja, a nakon točke prekida iz drugog roditelja. Pozicija i broj praznina uzima se u obzir kako bi se zadržao integritet niza (kako ne bi došlo do situacije u kojoj u potomku niz ima višak ili manjak aminokiselina).

Svaki od operatora ima nezavisnu vjerojatnost odabira. Empirijska promatranja pokazala su da je jednaka šansa odabiranja oba operatora, uz 30% vjerojatnosti izvođenja horizontalnog križanja, te 50% vjerojatnosti izvođenja vertikalnog križanja, uglavnom prihvatljiva. Točnije, oba operatora biraju se s jednakom vjerojatnošću, međutim, horizontalno križanje s vjerojatnošću izvođenja od

30% znači da će se križanje izvesti u 30% slučajeva, a u ostalim slučajevima potomak će biti identičan jednom od roditelja (slučajno odabranom). Isti postupak vrijedi i za vertikalno križanje. [31]



Slika 5.3 -primjeri operacija križanja; preuzeto iz [31]

Operatori su ostvareni kao razredi `HorizontalCrossover` i `VerticalCrossover`, koji nasljeđuju ECF-ov razred `CrossoverOp`.

5.4.2 Operatori mutacije

Operatori mutacije mogu djelovati samo na prazninama, i to kroz četiri općenite operacije:

- *otvori prazninu*
- *proširi prazninu*
- *smanji prazninu*
- *obriši prazninu*

Navedene operacije ostvarene su pomoću tri operatora.

Za otvaranje nove praznine, nasumično se odabire pozicija u poravnanju, te se umeće praznina slučajno odabrane veličine, u intervalu od jedan do 5% duljine niza u kojeg se umeće praznina.

Za proširenje praznine, nasumično se odabire blok praznina u poravnanju i dodaje se jedna praznina u blok.

Treći operator, smanjenje praznine, djeluje tako da se iz nasumično odabranog bloka praznina ukloni jedna praznina. Ako je blok praznina sastojao od samo jedne praznine, ona će biti uklonjena.

Programska ostvarenja operatora mutacija su razredi koji nasljeđuju ECF razred `MutationOp`, poimence `GapOpenMutation`, `GapExpandMutation`, `GapReduceMutation`.

5.4.3 Funkcija dobrote

Kao što je navedeno u opisu problema višestrukog poravnavanja proteina, za ocjenjivanje dobrote pojedinog poravnanja koristimo dvije komponente: supstitucijsku matricu kojom vrednujemo očuvanja ili supstitucije aminokiselina na odgovarajućoj poziciji u dva proteinska niza, te kažnjavanje praznina koje nam omogućava numeričku kvantifikaciju umetanja i brisanja aminokiselina u proteinima.

Budući da je smjer supstitucija nepoznat, matrice su simetrične. Kao što je navedeno u poglavlju 4 , najčešće korištene matrice su iz PAM, BLOSUM i Gonnet porodica. Očuvanja i supstitucije koje se pojavljuju češće nego što očekivano slučajnošću, u matricama su vrednovane pozitivnim; one koje se pojavljuju rjeđe od slučajno očekivanih imaju negativne vrijednosti. U ovom programskom rješenju implementirane su BLOSUM62 i Gonnet250 matrice.

Kažnjavanje praznina (tj. blokova praznina) izvršava se prema jednadžbi 4.1. Numeričke

vrijednosti kazni za otvaranje i proširenje praznina ovise o supstitucijskoj matrici koja se koristi, i moraju biti uravnotežene s vrijednostima u matrici. Previsoka kazna sprječavat će pojavu praznina i onda kada su one potrebne. S druge strane, preniska kazna će uzrokovati pojavljivanje praznina posvuda u poravnanju. Primjerice, u Clustal programu su zadane vrijednosti za BLOSUM matricu -10 za otvaranje praznine, te -0.2 za svaku proširenu poziciju. Jednake ili približno jednake vrijednosti korištene su i u ovom radu.

Postoji nekoliko metoda evaluacije dobrote, međutim često se koristi suma dobrota svih mogućih dvostrukih poravnanja nizova – suma parova –.[31] Ovaj pristup pretpostavlja da su nizovi međusobno neovisni, te precjenjuje broj zamjena aminokiselina. Kako bi se poništio ovaj efekt, koristi se **težinska** suma parova (eng. *weighted sum-of-pairs*), koja umanjuje utjecaj najpovezanijih nizova. Težinska suma parova definirana je jednadžbom 5.1, gdje je *k* broj nizova, *s* je suma para, a *w* težina pripadajućeg para.

$$Težinska\ Suma\ Parova = \sum_{i=2}^{k} \sum_{j=1}^{i-1} w_{ij}\, s_{ij} \tag{5.1}$$

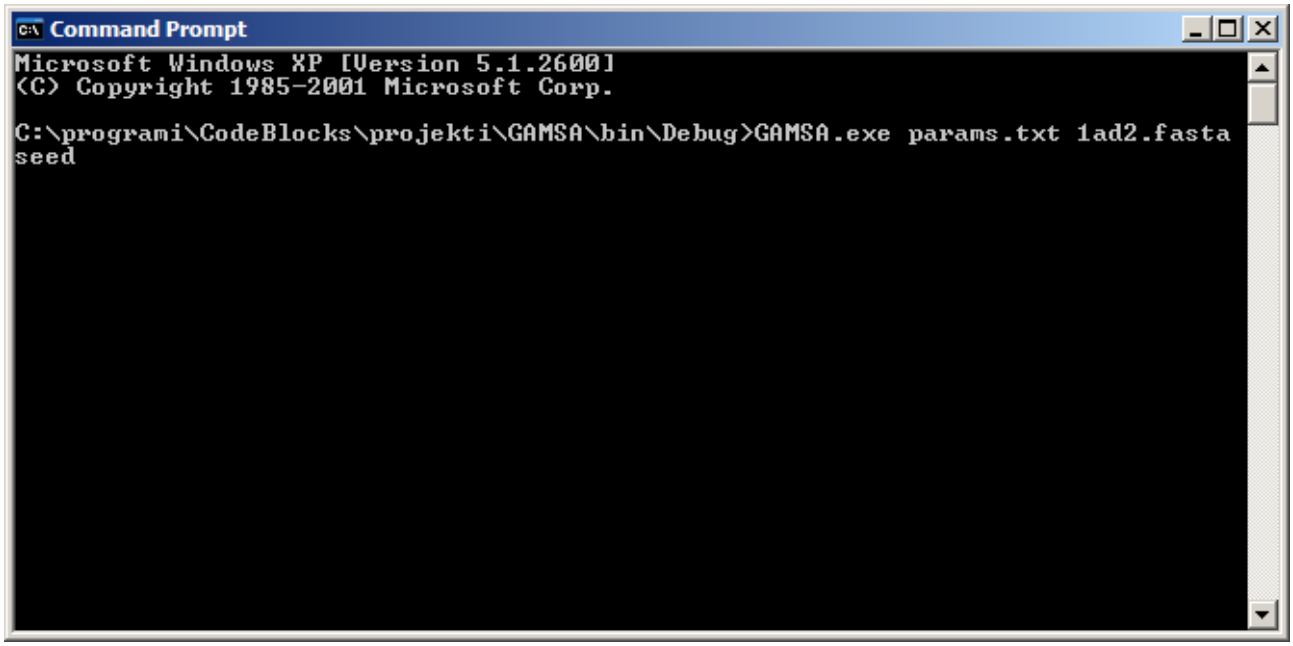
ClustalW[35], program koji implementira progresivni algoritam, koristi težinsku sumu parova. Program ostvaren u ovom radu koristi ClustalW kako bi dobio matricu udaljenosti (eng. *distance matrix*), iz koje dobiva težine.

5.4.4 Pokretanje programa GAMSA

Programsko rješenje `GAMSA` ostvareno je u CodeBlocks IDE sučelju, na operacijskom sustavu Windows XP Proffesional. Kao i sam ECF, program je ostvaren u C++ programskom jeziku, bez korištenja dodatnih biblioteka, osim onih koje koristi i sam ECF – STL i Boost. Program `GAMSA.exe` se pokreće iz konzole i prima sljedeće parametre, u navedenom redoslijedu:

- konfiguracijska datoteka
- neporavnati nizovi u datoteci FASTA formata
- opcionalni parametar "seed"; ako je naveden tada će ClustalW prvo poravnati nizove, a poravnanje će se iskoristiti kao predložak za kreiranje početne populacije. Jedna jedinka u početnoj populaciji bit će poravnanje iz ClustalW-a.

Primjer pokretanja vidljiv je na slici 5.4:



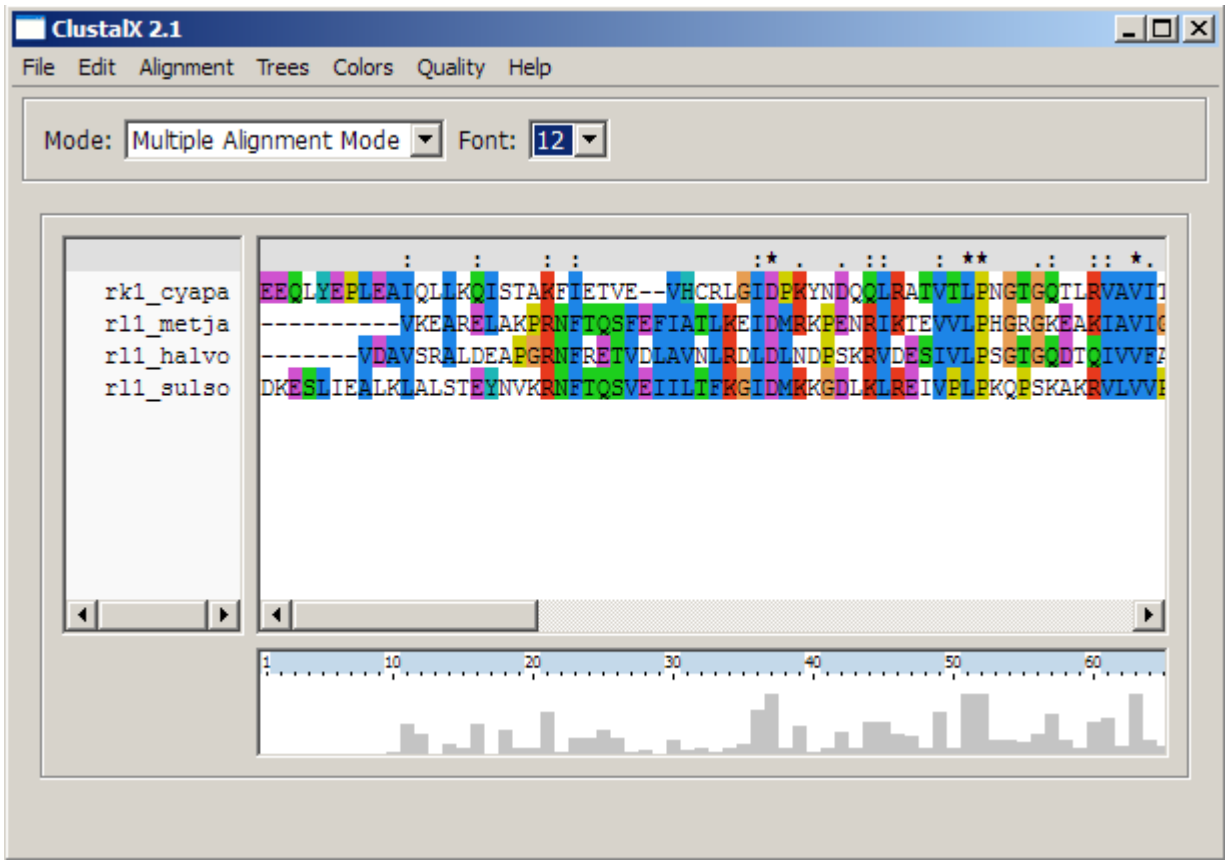
Slika 5.4-eventualni način pokretanja programa

5.4.5 Vizualizacija rješenja

Nakon što program završi s izvođenjem, najbolju jedinku sprema u datoteku FASTA formata. Ako je ulazna datoteka s nizovima u *fasta* formatu imenovana kao *nizovi.fasta*, tada će se izlazna datoteka s rješenjem zvati *nizoviGAMSA.fasta.*

Osim običnim preglednicima teksta, rješenje se može odlično vizualizirati u ClustalX programu. ClustalX je program identične namjene i uporabe kao i ClustalW, osim što, za razliku od ClustalW-a, posjeduje grafičko sučelje. ClustalX je, kao i ClustalW, besplatan i slobodno dostupan na web adresi [35].

Primjer vizualizacije ClustalX-om nalazi se na slici 5.5:



Slika 5.5-ClustalX

6 Rezultati ispitivanja

6.1 Uvod u ispitivanja

Budući da ishod genetskog algoritma prilično ovisi o izboru parametara, ispitivanja su načinjena s tim na umu. Za skup testnih podataka odabrano je šest testnih skupina proteina, koje se mogu pronaći na web adresi BaliBase *benchmarka* [36]. Šest skupina podijeljeno je u tri kategorije:

- *dvije skupine proteina s međusobnim identitetom nizova manjim od 25%*
- *dvije skupine proteina s međusobnim identitetom nizova između 20% i 40%*
- *dvije skupine proteina s međusobnim identitetom nizova većim od 35%*

Ispitivanja su izvedena tako da se ispita utjecaj veličine populacije, broja generacija te vjerojatnosti mutacije na dobrotu najbolje jedinke.

Također, budući da funkcija dobrote ne predstavlja i biološki optimalan kriterij za poravnavanje, dobiveni rezultati uspoređeni su s poravnanjima iz baze podataka BaliBase *benchmarka.*

Ispitivanja o utjecaju parametara GA na konačnu dobrotu vršena su na *1idy* skupini proteina uz BLOSUM62 supstitucijsku matricu, kaznu za otvaranje praznine -5, te kaznu za širenje praznine -0.1, dok su sve ostale skupine proteina uspoređene na BaliBase *benchmarku*. Važno je napomenuti kako *1idy* skupina dolazi iz kategorije s međusobnim identitetom nizova manjim od 25%.

Budući da su rezultati ispitivanja mnogo bolji i brže konvergiraju kada se početna populacija slučajno generira iz već pretporavnate jedinke, za što se koristi Clustal W, samo ti rezultati su prikazani. Pokazat će se da genetski algoritam ovog programskog ostvarenja, prema BaliBase *benchmarku,* može poboljšati poravnanja dobivena ClustalW-om, te također nadmašiti *Simple MSA-GA* program opisan u članku [31].

6.2 Rezultati ispitivanja

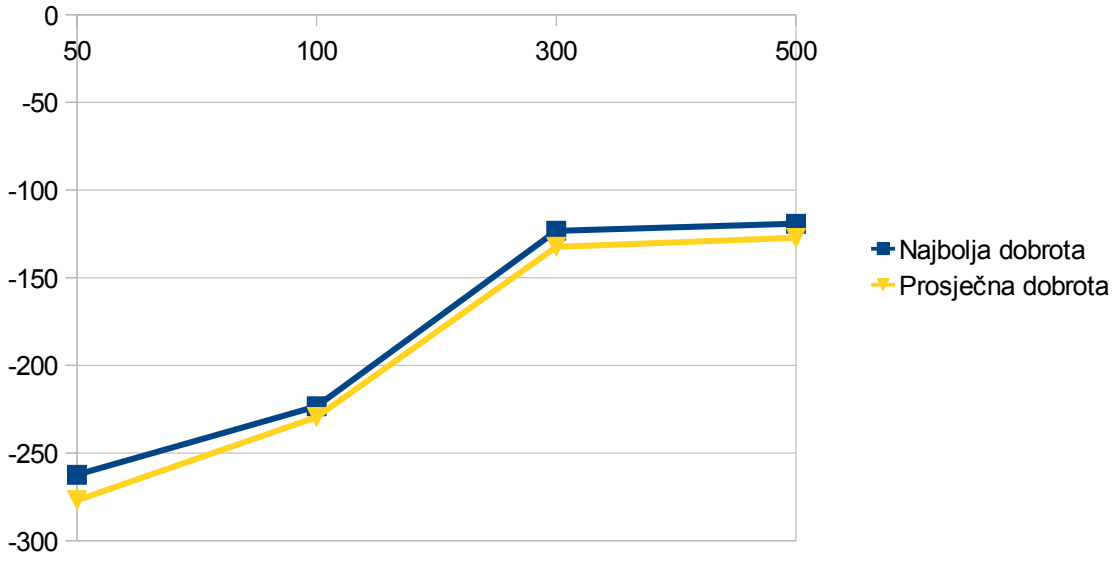
Algoritam selekcije korišten u ispitivanjima je 3-turnirska *steady-state* selekcija. Sva ispitivanja izvedena su s 10 ponavljanja, te su prikazane prosječne vrijednosti. Kriterij zaustavljanja za svaku skupinu ispitivanja je broj generacija, koji iznosi 500, osim kod ispitivanja utjecaja broja generacija, kada je, naravno, promjenjiv.

6.2.1 Utjecaj veličine populacije

Iz tablice 6.1 i grafa 6.1 vidljivo je kako veća populacija ima pozitivan utjecaj na dobrotu najbolje jedinke, te također na prosječnu dobrotu, međutim taj utjecaj se smanjuje rastom populacije.

Tablica 6.1

Skupina proteina	1idy			
Broj generacija	500			
Vjerojatnost mutacije	0.1			
Veličina populacije	50	100	300	500
Najbolja dobrota	-262.353	-223.178	-123.245	-119.173
Prosječna dobrota	-276.846	-229.407	-132.362	-127.263
Standardna devijacija dobrote	38.954	25.17	32.835	32.449



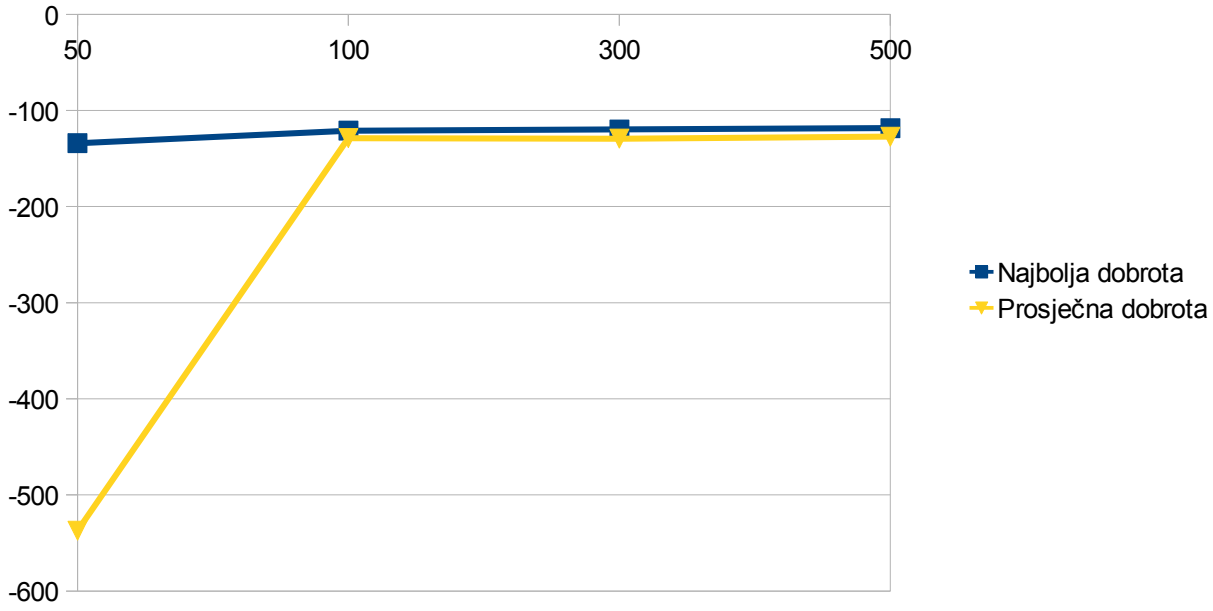
Graf 6.1 - ovisnost dobrote (y-os) o veličini populacije (x-os)

6.2.2 Utjecaj broja generacija

Kad govorimo o utjecaju broja generacija, primjetna je promjena u prosječnoj dobroti i standardnoj devijaciji kad broj generacija povećamo s 50 na 100. Razlike od 100 do 500 generacija nisu toliko značajne. Navedeno je prikazano tablicom 6.2 i grafom 6.2.

Tablica 6.2

Skupina proteina		1idy		
Veličina populacije		500		
Vjerojatnost mutacije		0.1		
Broj generacija	50	100	300	500
Najbolja dobrota	-134.027	-120.982	-119.696	-118.451
Prosječna dobrota	-537.239	-128.731	-129.344	-127.275
Standardna devijacija dobrote	51.947	35.583	33.468	37.22



Graf 6.2 - ovisnost dobrote (y-os) o broju generacija (x-os)

6.2.3 Utjecaj vjerojatnosti mutacije

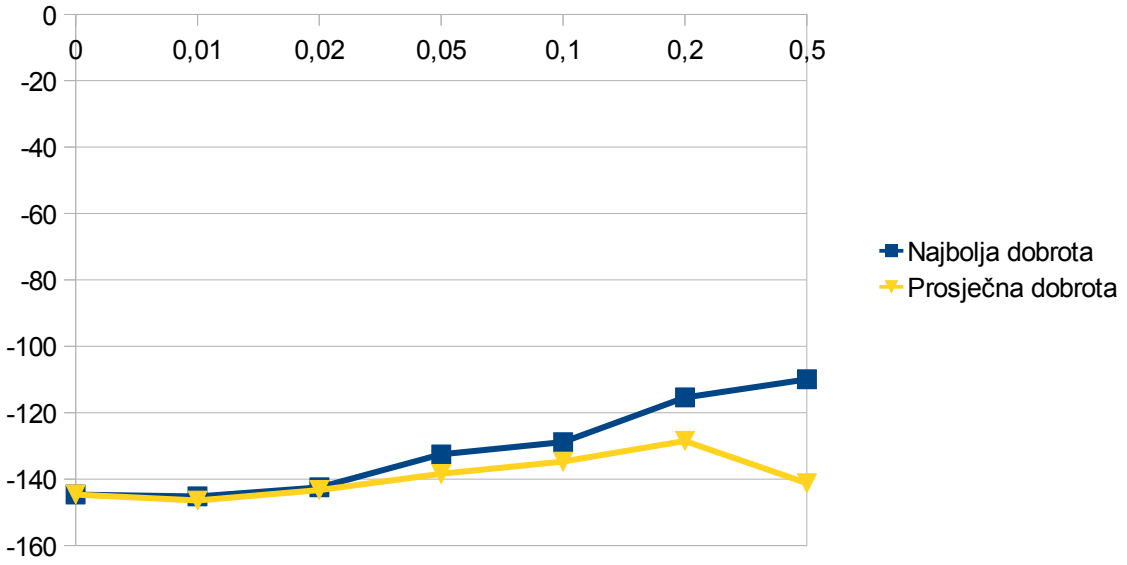
Tablica 6.3 i graf 6.3 prikazuju ovisnost dobrote o vjerojatnosti mutacije jedinke. Iz tablice je vidljivo kako bez mutacije sve jedinke postaju jednake najboljoj. Uvođenjem mutacije unosimo raznolikost u genetski kôd populacije, međutim razlike između pojedinih razina vjerojatnosti mutacije ne upućuju na konkretne zaključke. Kad vjerojatnost prijeđe 0.2, populacija počinje gubiti na kvaliteti.

Kod niskih vrijednosti vjerojatnosti mutacije (0.01, 0.02) možemo vidjeti kako se povećava raznolikost populacije u odnosu na nultu vjerojatnost, ali ta vjerojatnost očito nije dovoljna da poboljša rezultate. Od vjerojatnosti 0.05 na više uočavamo poboljšanja najbolje i prosječne dobrote.

Kod vjerojatnosti 0.5 primjećuje se slabija prosječna dobrota, no u više navrata algoritam dolazi do kvalitetnije najbolje jedinke u odnosu na ostale vjerojatnosti.

Tablica 6.3

Skupina proteina	1idy						
Veličina populacije	300						
Broj generacija	500						
Vjerojatnost mutacije	0	0.01	0.02	0.05	0.1	0.2	0.5
Najbolja dobrota	-144.67	-145.228	-142.492	-132.58	-128.859	-115.394	-109.935
Prosječna dobrota	-144.67	-146.458	-143.281	-138.437	-134.736	-128.536	-141.249
Std. devijacija dobrote	~0	10.209	13.579	23.877	42.282	42.173	63.97



Graf 6.3 - ovisnost dobrote (y-os) o vjerojatnosti mutacije (x-os)

6.2.4 BaliBase benchmark

Referentna poravnanja u BaliBase bazi podataka pomno su odabrana i ručno podešena kako bi bila biološki što smislenija, te služe upravo za ispitivanje raznih programa za višestruko poravnavanje proteina. Osim toga, na web adresi *benchmarka* nalaze se rezultati ispitivanja nekolicine programa za višestruko poravnavanje, što uvelike olakšava usporedbu ovog praktičnog rada s drugim programima.

BaliBase *benchmark* daje ocjenu sličnosti testiranog poravnanja s vlastitim referentnim poravnanjem, normaliziranu na interval *[0,1],* preciznosti tri decimalna mjesta. Na slici 6.1 prikazano primjer BaliBase poravnanja u MSF formatu.

Kako bi se dobila ocjena višestrukog poravnavanja u odnosu na referentno BaliBase poravnavanje, potrebno je dohvatiti (eng. *download*) program `bali_score.c`, te ga izvršiti, npr. na sljedeći način:

```
c:>\bali_score\bali_score.exe refMSA.msf testMSA.msf
```

U gornjoj konzolnoj naredbi, `refMSA.msf` se odnosi na referentno poravnanje u MSF formatu, koje se također može dohvatiti s BaliBase web stranice. `testMSA.msf` se odnosi na datoteku u kojoj se nalazi poravnanje koje uspoređujemo s referentnim. Ono također mora biti u MSF formatu, a kako je izlaz programskog rješenja ovog rada u FASTA formatu, potrebno je pretvoriti datoteku iz FASTA u MSF format. To možemo učiniti pomoću ClustalW konzolne aplikacije, npr. na sljedeći način:

```
c:>\clustalW\clustalW.exe -CONVERT -INFILE=testMSA.fasta -OUTPUT=gcg
```

```
1ycc            tefkagsakk gatlfktrcl qchtvekg.. ....gphkvg pnlhgifgrh
1c2r            .....gdaak gekefnk.ck tchsiiapdg teivkgaktg pnlygvvgrt
1etp            ....agdaea gqgkvav.cg achgvd.... .....gnspa pn....fpkl
3c2c            ....EGDAAA GEKVSKK.CL ACHTFDQGG. .....ANKVG PNLFGVFENT


1ycc            sgqaegysyt dan..ikk.. .nvlwdennm .seyltnpkk yip.......
1c2r            agtypefkyk dsi..valga sgfawteedi .atyvkdpga flkekldd..
1etp            agqgeryllk qlq.dikag. .stpgapegv grkvlemtgm ldp.......
3c2c            AAHKDNYAYS ESYTEMKak. .gLTWTEANL .AAYVKNPKA FVLeksgdpk


1ycc            gtkmafgglk kekdrndlit ylkkace
1c2r            kkaktgmafk lakggedvaa ylasvvk
1etp            .........l sdqdlediaa yfssqkg
3c2c            aKSKMTFKLT KDDEIENVIA YLKTLK.
```

Slika 6.1-primjer referentnog BaliBase poravnanja

U sljedećim tablicama dani su rezultati BaliBase *benchmark* testiranja na programskom rješenju. U tablice dohvaćene s BaliBase web stranice dodani su rezultati za programsko rješenje ovog rada (s ClustalW pretporavnanjem za stvaranje početne populacije), te rezultati Simple MSA-GA s ClustalW pretporavnanjem za stvaranje populacije, iz članka[31]. Rezultati za Clustal zamijenjeni su novim podacima, dobivenih obavljanjem ispitivanja.

Pri interpretaciji rezultata, bitno je imati na umu kako SAGA **ne koristi** pretporavnate nizove, već slučajno generira početnu populaciju, što je pristup koji kod GAMSA i Simple MSA-GA

U tablici 6.4 možemo vidjeti kako je GAMSA poboljšala rezultat Clustal-a, čija je rješenje koristila kao pretporavnanje u izgradnji populacije. Također, GAMSA je nadmašila i Simple MSA-GA koji počiva na sličnim idejama. Nadalje, uvjetno rečeno, GAMSA ima kvalitetnije rješenje od SAGA-e, najpoznatijeg i mnogo kompleksnijeg GA rješenja za višestruko poravnavanje proteina. Štoviše, za *1idy* skupinu, GAMSA ima najviši rezultat, a za *1tvxA* drugi najviši (nakon *PRRP*).

Tablica 6.5 prikazuje rezultate dobivene na skupinama proteina koje imaju između 20% i 40% identiteta. Za skupinu *1ycc* GAMSA ponovno poboljšava rješenje Clustal-a, neznatno je lošija od SAGA-e, uglavnom je u rangu s drugim rješenjima, a još uvijek značajno bolja od Simple MSA-GA, na temelju čijih ideja je i ostvarena. Za skupinu *3cyr* poboljšan je rezultat Clustal-a, gotovo je jednak kao i Simple MSA-GA, dok SAGA ima nešto veću prednost. No, rezultat je u rangu s ostalim programima, te je stoga zadovoljavajuć.

U tablici 6.6 nalaze se rezultati za skupine proteina koje imaju više od 35% identiteta. Za skupinu *1krn* GAMSA daje rezultat istovjetan Clustal-u. SAGA i Simple MSA-GA imaju neznatno lošije rezultate. Za skupinu *2fxb* GAMS-a ima rezultat istovjetan Clustalu, a bolji od SAGA-e i Simple MSA-GA.

Vrijednosti u tablicama kreću se u intervalu *[0,1],* te predstavljaju ocjenu kvalitete poravnanja u usporedbi s referentnim poravnanjem prema BaliBase *benchmark-u.* Što je poravnanje sličnije referentnom, to je ocjena viša, tj. poravnanje kvalitetnije.

Opis programa navedenih u tablicama, te njihova usporedba, opisani su u članku [37].

Tablica 6.4 -skupine proteina s manje od 25% identiteta

	1idy	1tvxA
PRRP	0.606	0.378
CLUSTAL	0.546	0.223
SAGA	0.342	0.278
DIALIGN	0.018	0.306
SB_PIMA	0.145	0.344
ML_PIMA	0.062	0.344
MULTIALIGN	0.566	0.228
PILEUP8	0.080	0.344
MULTAL	0.080	0.244
HMMT	0.138	0.108
GAMSA	0.671	0.386
Simple MSA-GA	0.438	0.209

Tablica 6.5 -skupine proteina s 20% do 40% identiteta

	1ycc	3cyr
PRRP	0.856	0.907
CLUSTAL	0.810	0.722
SAGA	0.837	0.908
DIALIGN	0.749	0.750
SB_PIMA	0.815	0.887
ML_PIMA	0.815	0.887
MULTIALIGN	0.932	0.858
PILEUP8	0.939	0.854
MULTAL	0.940	0.841
HMMT	0.217	0.454
GAMSA	0.816	0.780
Simple MSA-GA	0.653	0.789

Tablica 6.6-skupine proteina s više od 35% identiteta

	1krn	2fxb
PRRP	1.000	0.945
CLUSTAL	0.975	0.993
SAGA	0.993	0.951
DIALIGN	1.000	0.945
SB_PIMA	0.986	0.945
ML_PIMA	0.986	0.945
MULTIALIGN	0.993	0.945
PILEUP8	1.000	0.945
MULTAL	1.000	0.945
HMMT	0.944	0.930
GAMSA	0.975	0.993
Simple MSA-GA	0.895	0.985

7 Zaključak

Genetski algoritmi stohastička su optimizacijska metoda evolucijskog računarstva primjenjiva na čitav niz problema koji su egzaktno nerješivi ili su teško rješivi. Genetski algoritmi u pravilu proizvode približna rješenja, koja često u inženjerskom pristupu smatramo dovoljno dobrima.

Problem višestrukog poravnavanja proteina jedan je od takvih zadataka, kod kojeg egzaktne metode daju dobre rezultate samo za nekoliko nizova, a za veći broj nizova su neprimjenjive zbog enormnog vremena izvođenja. Druge metode, npr. progresivne, imaju odličnu brzinu izvođenja, ali često proizvode rješenja koja nisu optimalna, jer im problem predstavljaju lokalni optimumi.

Genetski algoritmi pokazali se se kao potencijalno dobar pristup rješavanju višestrukog poravnavanja proteina. U gotovo svim ispitnim slučajevima, programsko rješenje temeljeno na GA poboljšalo je rezultate najpoznatijeg programa baziranog na progresivnoj metodi, što nikako ne znači da je GAMSA na neki način kvalitetnija od poznatih rješenja koja se razvijaju već desetljećima, čak i kad bismo ignorirali neusporedivo veću vremensku zahtjevnost genetskog algoritma. No, iako u relativnom smislu spomenuta poboljšanja možda nisu velika, i najmanja poboljšanja mogu biti od značaja biolozima, bez obzira na utrošak vremena, a ovisno o primjeni poravnanja. Također, u većini ispitnih slučajeva GAMSA nadmašuje Simple MSA-GA, rješenje koje je poslužilo kao inspiracija za ovaj rad. Čini se kako je GAMSA posebno pogodna za višestruko poravnavanje nizova niskog međusobnog identiteta. Ako se uzme u obzir da su u radu implementirane najjednostavnije ideje, može se zaključiti kako su rezultati u potpunosti zadovoljavajući, te da postoji prostor za buduću optimizaciju i poboljšanja.

Bojan Dunaj

Literatura

1: Fogel, Corne, Evolutionary Computation in Bioinformatics, 2003.

2: Bairoch, A., Bucher, E, and Hofmann, K., Nucl. Acids Res., 25:217-221., 1997

3: C. Notradame, Desmong G. Higgins, SAGA: Sequence Alignment by Genetic Algorithm, http://nar.oxfordjournals.org/content/24/8/1515.full,

4: Wikipedia, Bioinformatics, 2011, http://en.wikipedia.org/wiki/Bioinformatics

5: Volker Sperschneider, Bioinformatics: Problem Solving Paradigms, 2008.

6: Francis Crick, On Protein Synthesis, 1958

7: Wikipedia, Transkripcija, , http://hr.wikipedia.org/wiki/Transkripcija_(biologija)

8: Medicinski fakultet Osijek, Struktura i funkcija proteina, 2011, http://biochem.mefos.hr/biokemija/P2.pdf

9: Worldwide Protein Data Bank, Protein Data Bank, 2011, http://www2.rcsb.org/pdb/

10: National Center for Biotechnology Information, NCBI Protein Database, , http://www.ncbi.nlm.nih.gov/protein

11: Wikipedia, Gene prediction, , http://en.wikipedia.org/wiki/Gene_finding

12: Wikipedia, Biological target, , http://en.wikipedia.org/wiki/Biological_target

13: Wikipedia, Small molecule, , http://en.wikipedia.org/wiki/Small_molecule

14: Griffin, J.P., The Textbook of Pharmaceutical Medicine, 2009

15: PAC, Nomenclature for radioanalytical chemistry (IUPAC Recommendations 1994), 1994

16: William L. Jorgensen, The many roles of computation in drug discovery, 2004

17: Wikipedia, Protein Structure Alignment, 2011,
http://en.wikipedia.org/wiki/Protein_structural_alignment

18: Berg JM, Tymoczko JL, Stryer L., Biochemistry, 2002

19: Christoph Gille, STRAP, 2011, http://www.bioinformatics.org/strap/

20: Wikipedia, Structural alignment software, 2011,
http://en.wikipedia.org/wiki/Structural_alignment_software

21: Wikipedia, Critical Assessment of Techniques for Protein Structure Prediction, 2011,
http://en.wikipedia.org/wiki/CASP

22: Wikipedia, Protein Structure Prediction, 2011,
http://en.wikipedia.org/wiki/Protein_structure_prediction

23: Cyrus Levinthal, How to Fold Graciously, 1969

24: Marek Obitko, Introduction to Genetic Algorithms, 1998,
http://www.obitko.com/tutorials/genetic-algorithms

25: Domagoj Jakobović, Genetski algoritmi - predavanje, 2007

26: Wikipedia, Genetic algorithm, Reproduction, 2011,
http://en.wikipedia.org/wiki/Genetic_algorithm#Reproduction

27: Ivan Kokan, Primjena Genetskog Programiranja Za Rješavanje Problema Prijanjanja Proteina, 2010

28: David Wheeler, Selecting the Right Protein Scoring Matrix, 2003

29: Swiss Institute of Bioinformatics, SwissProt Protein Knowledgebase, 2011, http://expasy.org/sprot/

30: Wikipedia, Multiple Sequence Alignment, 2011, http://en.wikipedia.org/wiki/Multiple_sequence_alignment

31: C Gondro, B.P. Kinghorn, A Simple Genetic Algorithm for Multiple Sequence Alignment, 2007

32: C. Notredame, D.G. Higgins, SAGA: Sequence Alignment by Genetic Algorithm, 1996

33: Domagoj Jakobović, Evolutionary Computation Framework, 2011, http://gp.zemris.fer.hr/ecf/

34: Wikipedia, FASTA Format, 2011, http://en.wikipedia.org/wiki/FASTA_format

35: Conway Institute UCD Dublin, Clustal, 2011, http://www.clustal.org

36: J. Thompson, F. Plewniak, O. Poch, BAliBASE: A benchmark alignments database for the evaluation of multiple sequence alignment programs, 2011, http://bips.u-strasbg.fr/en/Products/Databases/BAliBASE/

37: Thompson et al, A Comprehensive Study of Multiple Sequence Alignment Programs, Nucleic Acids Res. 1999, 27(13), 2682-90, 1999