SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Primjena genetskih algoritama u bioinformatici

Ivona Škorjanc

Voditelj: *Domagoj Jakobović*

Zagreb, travanj 2016.

Sadržaj

1. Uvod

S napretkom tehnologije, računala i računarske znanosti, obujam i težina problema koje rješavamo i pokušavamo riješiti eksponencijalno raste. Zato je nastala potreba za novim i različitim načinima ostvarivanja programskih rješenja. Mnogi se problemi ne mogu riješiti jednostavnim, determinističkim algoritmima i tu započinje priča o razvoju područja umjetne inteligencije. Kako doći do dovoljno dobrog rješenja nekog problema? Znanstvenici su se dosjetili principa evolucije. Upravo je ta ideja poslužila i za stvaranje genetskih algoritama. Ako priroda ima način izdvajanja najboljih pojedinaca, možda se taj način može iskoristiti i za izdvajanje najboljih programskih rješenja. I zaista, primjenom principa evolucije i darvinizma („opstanak najjačih“), razvili su se evolucijsko računarstvo, genetski algoritmi (GA) i genetsko programiranje.

Jedno od područja znanosti u kojem su primjenjivi genetski algoritmi je genetika. Zbog potrebe za obradom velikog broja podataka i vrlo specifičnim skupovima podataka „obični“ su algoritmi često kompliciranii neefikasni u konačnom vremenu. Zato će se u ovom radu opisati i proučiti neke od konkretnih primjena genetskih algoritama u bioinformatici.

2. O genetskim algoritimima

2.1 Ideja genetskih algoritama

Princip rada genetskih algoritama (GA) analogan je principu evolucije. Za najlakše shvaćanje GA treba ponoviti osnovna načela evolucije. Pojedinac ima veću mogućnost preživljavanja ako je dobro prilagođen određenom okruženju. Genetika nas uči da se geni, odnosno svojstva roditelja nasljeđuju (nastavljaju na potomke). Ako preživljavaju pojedinci bolje prilagođeni okolišu, to znači da će preko njihovih potomaka opstati dobri, prilagođeni geni, dok će oni neprilagođeni izumrijeti zajedno s jedinkama koje su ih nosile. Dobri geni se miješaju i tako nastaju prilagođeni pojedinci. Postoje još neke pojave u genetici koje se mogu primijeniti na izradu genetskih algoritama. To su križanje i mutacija, promjene do kojih dolazi tijekom nasljeđivanja gena, a detaljnije će biti objašnjene u nastavku teksta.

Genetski algoritam traži rješenje nekog problema upravo imitirajući evoluciju i procese nasljeđivanja u genetici. Jedna jedinka u svijetu genetskih algoritama je jedno potencijalno rješenje, a algoritam započinje prvom populacijom koju čini n rješenja/jedinki. Ocjenjujemo koliko je to rješenje dobro ili loše, što je analogno stupnju prilagođenosti jedinke na svoje okruženje i uvjete. Za sljedeću populaciju biramo „potomke" bolje ocijenjenih rješenja, i tako nastavljamo reprodukciju jedinki koje se sve više približavaju konačnom rješenju sve dok nismo zadovoljni.

Jasno, trenutni načini na koji se implementiraju GA puno su grublji i manje precizni od ogromne kompleksnosti gena koji nose informacije o svojstvima organizama u prirodi. Bez obzira na to, GA čine vrlo moćan alat u mnogim granama računarskog inženjerstva i znanosti.

2.2 Princip rada genetskih algoritama

Slijedi definicija potrebnih podatkovnih struktura za realizaciju jednostavnog genetskog algoritma. Krećemo od početne populacije jedinki. To su najčešće nasumce izabrana potencijalna rješenja. Svaka jedinka u populaciji naziva se kromosomom. U biologiji kromosom je struktura u kojoj se nalazi dvostruka nit DNK, koja se onda razdvaja za vrijeme mejoze i mitoze. U DNK su spremljeni geni, osnovne jedinice nasljedne informacije. Dakle, u kontekstu biologije i u kontekstu GA, kromosomi su nositelji svojstava jedinke. Za svaku jedinku moramo izraziti i koliko je blizu ili daleko od željenog konačnog rješenja. Za to nam služi mjera koju nazivamo dobrota *(eng. fitness)*, a funkcija kojom definiramo kolika je dobrota rješenja nazivamo funkcija dobrote *(eng. fitness function)*.

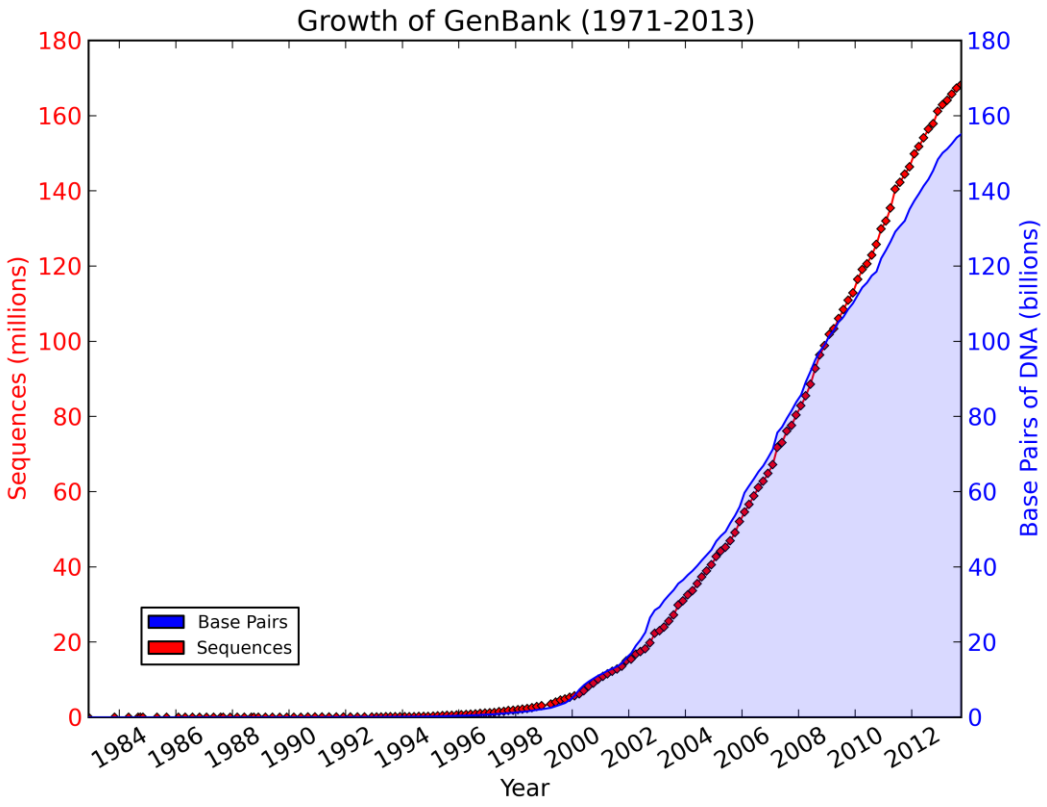
Kada smo sve jedinke podvrgli sudu funkcije dobrote, moramo odlučiti koliko, kako i koje će se jedinke reproducirati i stvoriti potomke. To se naziva postupkom selekcije. Odabrane jedinke nastavljaju dalje i stvaraju novu populaciju, a ostale odumiru. Postoje dva načina za dobivanje novih jedinki. Prvi je križanje, kombiniranje svojstava dviju ili više jedinki. Ovdje se termin križanja razilazi od termina u genetici, gdje je križanjem kromosoma (*eng. crossing-over*) mogu dobiti potpuno nova svojstva. Drugi način je mutacija koja, za razliku od križanja, dijeli svoj smisao sa svojim analogonom u genetici i predstavlja slučajnu promjenu manjeg dijela genetske informacije. Selekciju, križanje i mutaciju nazivamo genetskim operatorima jer njima djelujemo na svaku jedinku. Nakon što su sve jedinke prošle ova tri koraka, dobili smo novu populaciju koja je (vrlo vjerojatno) malo bliže konačnom rješenju. Opisani postupak ponavljat ćemo u petlji sve dok nije zadovoljen postavljeni uvjet završetka, ili dok ne istekne određeno vrijeme. Rezultat bi trebao biti blizu optimalnog rješenja.

```
Genetski_algoritam
{
      t = 0
      generiraj početnu populaciju potencijalnih rješenja P(0);
      sve dok nije zadovoljen uvjet završetka evolucijskog procesa
      {
            t = t + 1;
            selektiraj P'(t) iz P(t-1);
            križaj jedinke iz P'(t) i djecu spremi u P(t);
            mutiraj jedinke iz P(t);
      }
      ispiši rješenje;
}
```

3. O bioinformatici i povezanost s GA

Samo ime nam kaže da bioinformatika spaja dvije znanosti – biologiju i informatiku. Preciznije, bioinformatika primjenjuje računalnu tehnologiju za upravljanje i analizu bioloških i genetskih informacija, sve u svrhu povećanja razumijevanja bioloških procesa. Glavnina istraživanja u ovoj znanosti fokusirana je na prepoznavanje gena, sastavljanje nizova genoma i usporedbu različitih genoma, istraživanje, otkrivanje i dizajniranje lijekova, predviđanje strukture proteina i RNK, poravnavanje nizova DNK, RNK i proteina te modeliranje evolucije.

Povećana potreba za ovom granom znanosti nastala je nakon eksponencijalnog rasta javno dostupnih informacija o ljudskom genomu proizašlog iz projekta ljudskog genoma (*eng. Human Genome Project*) [4]. Cilj ovog poznatog internacionalnog projekta bio je da se identificira i analizira sav DNK u čovjeku, za svaki gen proučiti njegovu specifičnu strukturu i odrediti mu funkciju. Projekt je dovršen u travnju 2003. godine i čovječanstvu je omogućio uvid u način na koji priroda stvara i oblikuje ljudska bića. Na znanstvenicima je da iskoriste dobivene podatke za dobivanje novih znanja u područjima medicine i farmakologije (za razumijevanje ljudskih bolesti i razvijanje lijekova), ali i za daljnje istraživanje u biologiji i genetici.



Slika 3.1 Rast baze podataka o ljudskom genomu, preuzeto s [10]

Bioinformatika može se podijeliti na dvije važne pod-discipline, tj. zadaće:

I. Razvoj i implementacija programa koji omogućuju dostupnost, korištenje i upravljanje različitim biološkim i genetskim informacijama.

II. Razvoj novih algoritama i statističkih metoda koji ocjenjuju povezanost dijelova velikih skupova podataka. Takvi su, na primjer, algoritmi za prepoznavanje određenog gena unutar dugačkih sekvenci, predviđanje strukture i funkciju proteina, te za grupiranje sekvenci proteina u povezane cjeline.

Razvijanje ovakvih algoritama nije nimalo lak posao, i jedna vrsta algoritama koja se primjenjuje su upravo genetski algoritmi. Budući da najveći problem predstavlja masivna količina podataka koje treba procesirati (Slika 3.1), GA mogu biti vrlo korisni prvenstveno za sužavanje prostora pretrage u nekom istraživanju. Na taj način mogu se odabrati kandidati za daljnje pretrage, analizu i eksperimente, što značajno smanjuje troškove i količinu uloženog vremena.

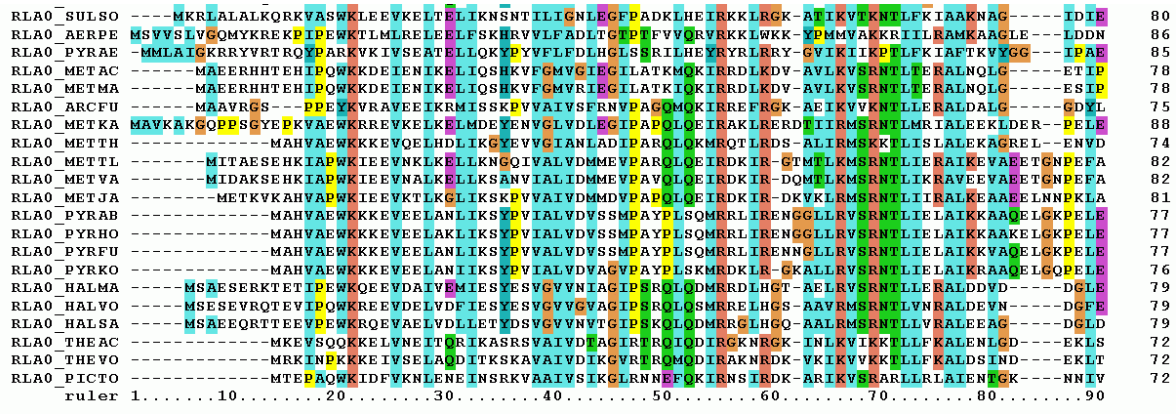
4. Primjene genetskih algoritama u bioinformatici

Postoje mnogi načini na koje se genetski algoritmi mogu primijeniti u bioinformatici i računalnoj biologiji. Na primjer, GA za otkrivanje motiva može koristiti za pronalaženje određenog gena u sekvenci ili kao pomoć u razlikovanju pravih uzoraka podsekvenci od pozadinskih sekvenci. GA za poravnavanje višestrukih sekvenci koristi se za utvrđivanje sličnosti među više različitih sekvenci. Predviđanje strukture proteina koristi se za predviđanje tercijarne 3D strukture sekvenci aminokiselina. Stvaranje filogenetskog stabla može se primijeniti na proučavanje hijerarhijske evolucijske veze između različitih vrsta, recimo usporedbu gena između čovjeka i životinja. Na problem rasporeda lijekova (npr. za pacijente na kemoterapiji) također se mogu primijeniti GA i tako na efikasan način pomoći u kliničkom raspoređivanju lijekova.

U daljnjem tekstu navest ću i objasniti neke od zanimljivijih i šire upotrijebljenih primjena genetskih algoritama u bioinformatici.

4.1 GA za poravnavanje višestrukih sekvenci

Sekvence su nizovi nukleinskih ostataka (nukleotida ili aminokiselina) koji grade DNK, RNK i proteine. U bioinformatici, poravnavanje sekvenci (*eng. sequence alignment*) je način uspoređivanja sekvenci kako bi se otkrili slični nizovi i dijelovi sekvence (podsekvence) koji bi mogli biti posljedica funkcionalne, strukturalne ili evolucijske veze između sekvenci. Poravnavanje višestrukih sekvenci podrazumijeva poravnavanje više od dvije sekvence. Grafički se poravnate sekvence najčešće prikazuju u obliku matrice, gdje se odgovarajuće komponente nalaze jedna ispod druge (Slika 4.1). Metode višestrukog poravnavanja nastoje poravnati sve sekvence u određenom setu sekvenci. Često se koriste za identifikaciju istih podsekvenci *(eng. conserved sequence regions)* u grupi sekvenci za koje se pretpostavlja da su evolucijski povezane. Također su korisne za utvrđivanje evolucijskih veza izgradnjom filogenetskih stabala (hijerarhijski dijagram evolucijskih veza).



Slika 4.1 Prikaz višestrukih sekvenci, preuzeto s [11]

Jedan od softverskih rješenja poravnavanja višestrukih sekvenci koji koristi genetski algoritam zove se SAGA *(Sequence Alignment by Genetic Algorithm)*[5]. Ideja je da se uzme funkcija koja mjeri kvalitetu (dobrotu) višestrukog poravnavanja i zatim ju se optimizira koristeći genetski algoritam. Skup dobro poznatih testnih slučajeva koristi se kao referenca za evaluaciju efikasnosti optimizacije.

Pseudokod algoritma je sljedeći:

```
Inicijalizacija     stvori početnu populaciju G0
                    n = broj generacija, i=0
Evaluacija          dok je (i<n) čini {
                          ocijeni populaciju generacije Gi
                          ako je populacija stabilna idi na kraj
                          izaberi jedinke koje ćeš zamijeniti
                          ocijeni očekivano potomstvo
Razmnožavanje             za (svakog potomka) čini {
                                odaberi roditelje iz Gi
                                odaberi operator
                                generiraj potomka
                                zadrži ili odbaci novog potomka u Gi+1
                                i++
                          }
                    }
Kraj
```

Inicijalizacija. Prvi korak u algoritmu jest stvaranje početne populacije jedinki (*eng. generation zero*). U SAGA-i podatkovna struktura jedinke je jednostavna: dvodimenzionalna matrica u kojoj svaki red predstavlja jednu poravnatu podsekvencu (kao na Slika 4.1). Svaki se niz sastoji od dvadeset simbola koji označavaju aminokiseline koje mogu graditi protein, i simbola za prazninu. Početne jedinke sadrže samo praznine ('-'). Ukupan broj generacija je unaprijed određen i fiksan.

Evaluacija. Svaka jedinka se boduje ovisno uspješnosti poravnavanja. To se radi na način da se pridruži određeni trošak za svaki poravnati nukleinski ostatak u stupcu (*eng. supstitution cost*), i za sve praznine (*eng. gap cost*). Zbrojeni, ovi troškovi daju globalni trošak tog poravnanja. Nadalje, svakom paru sekvenci pridružuje se njihova težina koja je određena sličnošću s drugim parovima. Ukupni trošak višestrukog poravnavanja (A) je

$$A = \sum_{i=2}^{N} \sum_{j=1}^{i-1} W_{i,j} \, \mathrm{COST}(A_i, A_j)$$

gdje je COST ukupni trošak između dviju poravnatih sekvenci (A_i i A_j), a $W_{i,j}$ je njihova težina. Jedinke s boljim rezultatom (njih oko 50%) nedirnute nastavljaju u novu generaciju, a one lošije će zamijeniti potomci. Ovaj postupak naziva se preklapanjem generacija (*eng. generation overlapping*).

Razmnožavanje. Prvo se nova generacija direktno puni s 50% najprilagođenijih jedinki prošle generacije. Ostalih 50% stvara se odabirom i promjenom roditelja. U SAGA-i se koristi čak 22 različita operatora za promjenu roditelja, a mogu se globalno svrstati u operatore mutacije (jedan roditelj) i križanja (dva roditelja).

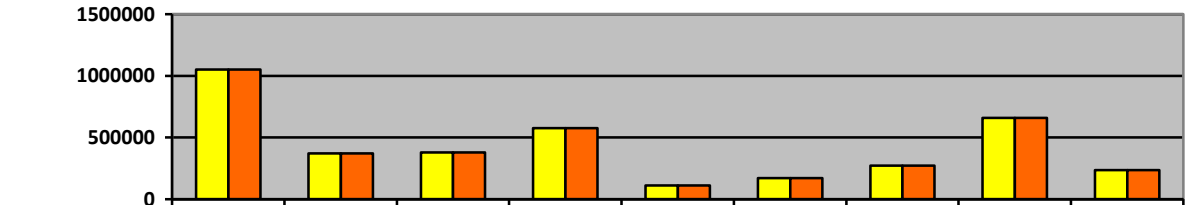
- Križanje. Postoje dvije implementacije; prva je križanje u jednoj točki *(eng. one-point crossover)* gdje se u nasumično odabranoj točki sekvenca dijeli na dva dijela. Lijevi dio spaja se sa desnim dijelom druge sekvence i obrnuto. Bolji od dva tako nastala potomka će „preživjeti“. Drugi način je primjena uniformnog križanja (*eng. uniform crossover*) koje podržava suptilnije višestruke razmjene podsekvenci između roditelja.
- Mutacija. Mnogo je implementiranih načina za manje ili veće promjene jednog roditelja. Dok križanja kombiniraju uzorke, operatori mutacije zaduženi su za kreiranje tih uzoraka. U grubo, to su operatori za umetanje praznina ili blokova praznina te premetanje blokova.

Za bolje performanse programa postoje još i operatori za pretraživanje blokova i lokalno preraspoređivanje.

Svi operatori na početku izvođenja programa imaju jednaku vjerojatnost korištenja, što ne mora biti jednako učinkovito za pojedine konkretne uzorke višestrukih sekvenci. Zato se koristi dinamičko raspoređivanje (*eng. dynamic scheduling*) operatora. To znači da je vjerojatnost korištenja nekog od operatora funkcija njegove efikasnosti u zadnjih nekoliko generacija. Da bi se izbjegla situacija da se jedan ili nekoliko operatora uopće ne koriste, postoji granica minimalne vjerojatnosti korištenja koja je dodijeljena operatoru.

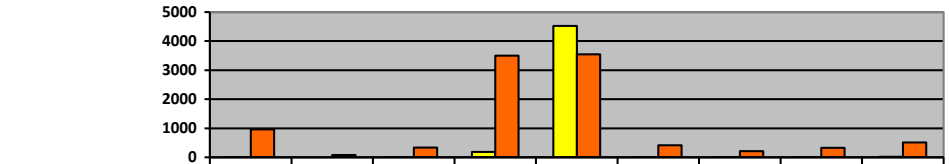
Za testiranje programa u ovom primjeru koristi se 13 testnih slučajeva poravnavanja poznatih sekvenci odabranih iz Pascarella baze podataka [9]. Rezultati su uspoređeni s rezultatima programa MSA i CLUSTAL W koji koriste nestohastički pristup rješavanja problema poravnavanja višestrukih sekvenci. MSA nastoji suziti prostor rješenja na relativno malo područje gdje se najbolje poravnanje vjerojatno nalazi. Program je zbog toga ograničen na manje uzorke od najviše sedam ili osam sekvenci. Za veće uzorke koristi se CLUSTAL W.

Tablica 4.1 Usporedba rezultata MSA-e i SAGA-e



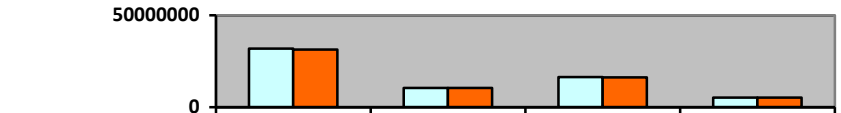
	Cyt c	Gcr	Ac	S protease	Chtp	Dfr secstr	Sbt	Globin	Plasto
MSA rezultat	1051257	371875	379997	574884	111924	171979	271747	659036	236343
SAGA rezultat	1051257	371650	379997	574884	111579	171975	271747	659036	236195

Tablica 4.2 Usporedba vremena izvođenja u sekundama (MSA i SAGA)



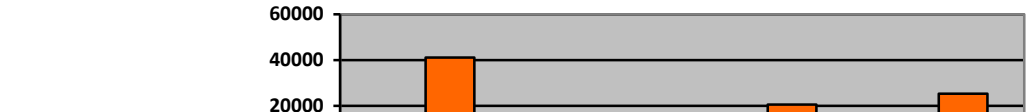
	Cyt c	Gcr	Ac	S protease	Chtp	Dfr secstr	Sbt	Globin	Plasto
MSA vrijeme izvođenja	7	3	13	184	4524	5	7	7	22
SAGA vrijeme izvođenja	960	75	331	3500	3542	411	210	330	510

Tablica 4.3 Usporedba rezultata CLUSTAL W i SAGA-e



	Igb	Ac protease2	S protease2	Globin2
CLUSTAL W rezultat	31812824	10514101	16354800	5249682
SAGA rezultat	31417736	10393145	16282179	5233058

Tablica 4.4 Usporedba vremena izvođenja u sekundama (CLUSTAL W i SAGA)



	Igb	Ac protease2	S protease2	Globin2
CLUSTAL W sličnost strukture (%)	60	16	21	18
SAGA sličnost strukture (%)	41135	12236	20537	25380

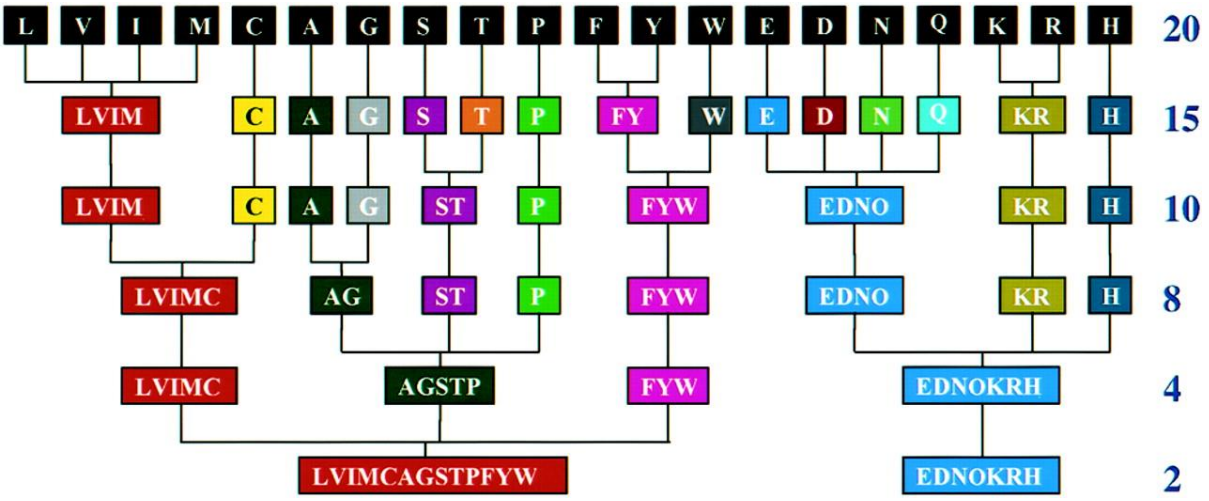
Za prikaz rezultata u tablicama Tablica 4.1,Tablica 4.2,Tablica 4.3 iTablica 4.4 odabrani su oni najbolji iz tri pokušaja izvođenja. U svim slučajevima SAGA je postigla barem jednako dobre rezultate kao MSA (manji broj bodova znači bolji

rezultat). U četiri slučaja (Gcr, Chtp, Dfr secstr i Plasto) rezultati su bili bolji. Da bi zaista utvrdili jesu li to i u biološkom smislu bolji rezultati, ta četiri slučaja uspoređena su s referentnim strukturalnim poravnanjima. Ova analiza pokazala je da poboljšanje optimizacije dosljedno povezano s poboljšanjem točnosti rezultata: ta su četiri slučaja sličnija referentnima od onih dobivenih MSA-om. Pokazuje se i da SAGA radi poravnanja točnije od CLUSTAL W algoritma na uzorcima realističnijih dužina.

S druge strane, vidljive su velike razlike u vremenima izvođenja ovih algoritama. SAGA, kao i drugi geneski algoritmi i stohastički procesi, najčešće zahtijeva mnogo više procesorskog vremena. Prema grafovima se vidi da su te razlike nekad i u više redova veličina (zato su neki stupci „nevidljivi“).

4.2 GA za pojednostavljenje abecede aminokiselina

Proteini se najčešće prikazuju kao niz (sekvenca) simbola, koji zajedno čine abecedu. Kako se proteini sastoje od aminokiselina, postoje 20 mogućih simbola koji se mogu pojaviti u sekvenci. Pojednostavljene abecede se rade na način da se dva ili više simbola spoje i zamijene jednim simbolom koji predstavlja taj skup aminokiselina (Slika 4.2). Pojednostavljene abecede u bioinformatici se koriste za predviđanje strukture i funkcije proteina. Broj mogućih simplifikacija abecede vrlo je velik. Nije praktično proučavati sva moguća pojednostavljenja kako bi pronašli ono koje je pogodno za određenu primjenu. Ovaj se problem svodi na problem razdvajanja skupa (svih dvadeset aminokiselina) na podskupove („clustere” dviju ili više aminokiselina).



Slika 4.2 Primjer pojednostavljene abecede različitih duljina, preuzeto s [12]

U ovom poglavlju uglavnom ću se bazirati na prilično jednostavnom algoritmu i računalnom programu opisanom u djelu [6]. Algoritam stvara pojednostavljenu abecedu željene (unaprijed određene) veličine (<20). Početak je stvaranje nasumične populacije rješenja. Iz te populacije dva se roditelja nasumično biraju za križanje (u ovom algoritmu ne koriste se mutacije). Grupe simbola svakog od roditelja dijele se na dva dijela tako da se blisko povezani nizovi odijele u svoju podgrupu. Za križanje se koriste te podgrupe, a ne originalne grupe roditelja. Grupe u rješenju djeteta stvaraju se na način da se nasumce odabere jedna aminokiselina i kombinira s

grupama koje sadrže tu kiselinu u roditeljskim rješenjima. Na primjer, ako roditelj 1 ima grupu {ALMRQ} a roditelj 2 grupu {APS}, odabir aminokiseline 'A' prouzročiti će pojavu, recimo, grupe {ALMPQS} u rješenju djeteta. Ovaj proces se ponavlja dok se sve aminokiseline ne pojave u rješenju djeteta. Ako je željena veličina pojednostavljene abecede postignuta prije nego što rješenje djeteta sadrži sve aminokiseline, ostatak simbola umeće se u grupe tako da ukupna ocjena pojednostavljenosti abecede bude najbolja moguća. Nakon što je stvoreno rješenje djeteta, jedno nasumično rješenje se briše i rješenje djeteta se umeće u populaciju.

Tablica 4.5 Rezultati izvođenja genetskog algoritma

Size	Avg. Score	Best Score Found	Avg. Run Time
4	135.317	129.600	00:01:04
5	118.173	109.362	00:01:02
6	105.997	98.743	00:01:01
7	90.848	83.133	00:01:02
8	79.208	73.095	00:01:03
9	65.300	62.833	00:01:09

Tablica 4.6 Rezultati izvođenja „Branch and Bound" algoritma

Size	Best Score Found	Run Time
4	120.917	02:42:00
5	105.705	08:46:07
6	92.967	13:44:52
7	81.000	11:57:31
8	69.167	05:42:47
9	58.500	02:03:10

Tablica 4.5 prikazuje rezultate izvođenja algoritma. Za svaku veličinu abecede, program se dvadeset puta pokrenuo s populacijom od 10 jedinki i u 50 generacija. Stupac „*Avg. Score*" prikazuje prosjek bodova tih dvadeset pokretanja. Bodovi su rezultat ocjenjivanja pojednostavljenosti abecede (funkcije dobrote u zadnjoj generaciji), a najniža ocjena predstavlja najbolji rezultat.

Tablica 4.6 prikazuje rezultate izvođenja drugog algoritma objašnjenog i provedenog u djelu [7], i služi za usporedbu performansi algoritama. Vidimo da drugi algoritam kontinuirano donosi bolje rezultate, ali ipak se ne radi o velikoj razlici. S druge strane, vrijeme izvođenja je višestruko dulje. Genetski algoritam izvodi se malo dulje od minute bez obzira na veličinu abecede, dok se vrijeme izvođenja „*Branch and Bound*" algoritma mijenja od dva sata do čak 13 sati.

Rezultati pokazuju da se pojednostavljene abecede aminokiselina mogu efikasno stvoriti koristeći genetski algoritam. Treba uzeti u obzir i to da rezultantne

abecede dvaju uspoređenih algoritama često nisu slične, i možda metoda ocjenjivanja nije dovoljna za utvrđivanje prikladnosti abecede.

4.3 GA za otkrivanje motiva u sekvencama DNA

U kontekstu analize sekvenci, motiv je uzorak nukleotidnih baza ili aminokiselina koji predstavlja biološki smisleno svojstvo zajedničko grupi sekvenci nukleinskih kiselina ili proteina. Otkrivanje motiva je proces otkrivanja motiva u biološkim sekvencama. Fokus ovog poglavlja bit će na otkrivanju motiva unutar promotorskih sekvenci gena. Promotor je dio DNK odgovoran za transkripciju (prepisivanje podataka iz DNK u RNK). Smisleni motivi koje želimo pronaći često su vrlo kratki i mogu se pojaviti bilo gdje u sekvencama duljine do oko 10 kb. Vrlo je vjerojatno i da postoji više relevantnih motiva unutar jednog skupa podataka.

Postoji nekoliko problema koji se mogu dogoditi ako problem otkrivanja motiva želimo riješiti pomoću standardnih genetskih algoritama. Prvi je potreba da se pronađe više podataka odjednom (više motiva unutar sekvence) a drugi je činjenica da biološki besmislena rješenja mogu biti bolje ocijenjena funkcijom dobrote od onih „pravih". Standardni GA uglavnom konvergiraju ka točno jednom rješenju ili skupu sličnih rješenja. Posljedično, takvi algoritmi ne mogu generirati višestruka različita rješenja koja su potrebna za efikasno otkrivanje motiva. Osmišljena su mnoga rješenja kako bi se doskočilo ovom problemu (npr. ograničavanje izbora roditelja za razmnožavanje, korištenje višestrukih podpopulacija ili prostorno razdijeljenih populacija). Ovdje ću opisati algoritam [8] koji koristi grupiranje populacije (*eng. population clustering*) za održavanje raznolikosti rješenja.

Ideja je da se populacija prije razmnožavanja (stvaranja nove populacije) podijeli u podpopulacije (particije). Roditelji se onda biraju isključivo međusobno unutar podpopulacija. Ovakvo eksplicitno odjeljivanje rješenja ima mogućnost da selekcija bude jača, rigoroznija unutar podskupina a slabija između različitih particija. To dovodi do optimizacije individualnih rješenja unutar skupine, istovremeno održavajući raznolikost rješenja među skupinama.

Algoritam se sastoji od faze inicijalizacije i iterativne faze koja sadrži grupiranje populacije, razmnožavanje, i ocjenjivanje dobrote. Prvo se populacija puni nasumično generiranim nizovima koji su pretvoreni u poseban matrični oblik koji čuva podatke o učestalosti pojavljivanja određenog nukleinskog ostatka (baze) na nekoj poziciji (*eng. position frequency matrix*). U početnoj populaciji učestalost pojavljivanja u matricama za svaku bazu je uniformno raspoređena. Zatim slijedi grupiranje populacije. Za to se može koristiti jedan od brojnih poznatih algoritama grupiranja. Jedan od jednostavnijih jednom prolazi kroz čitavu populaciju i ovisno o stupnju sličnosti jedinki čini jednu od dvije stvari: pridodaje rješenje jedinku particiji ili ju koristi za stvaranje nove particije.

Razmnožavanje se događa zasebno unutar svake grupe, s brojem djece proporcionalnim relativnoj dobroti rješenja u toj grupi (dobrota grupe u usporedbi s dobrotom drugih grupa). Svaka grupa stvara barem jedno novo rješenje da bi se očuvala raznolikost. Roditelji za razmnožavanje biraju se ovisno o rangu njihove dobrote. Mutacija se primjenjuje ovisno o vjerojatnosti pridijeljenoj svakoj poziciji u

nukleotidu i radi na način da nasumce promijeni učestalost dodijeljenu jednoj ili više baza i zatim normalizirajući ostale učestalosti tako da zbroj ostane uniforman u ostalim matricama (prikazu sekvenci). Križanje se izvodi tako da se odaberu točke (pozicije) s odabranom vjerojatnošću i zatim se zamijeni podskup stupaca matrice između svakog drugog para odabranih točaka. Mutacija i križanje primjenjuju se neovisno jedno od drugog i ovise o unaprijed određenom omjeru mutacija i križanja. Nakon razmnožavanja, novonastale jedinke odmah se ocjenjuju funkcijom dobrote.

Testiranje algoritma obavlja se na dva načina: koristeći sintetičke skupove podataka (umjetno napravljeni podaci) koji sadrže sekvence DNK različitih fiksnih duljina ili koristeći skup mišićno-specifičnih promotorskih sekvenci (promotori koji su aktivni u određenom tipu mišića/stanica) za koje je poznato da sadrže višestruke promotorske elemente (motive).

Prvi test provodi se za motive HFH-1, HLF i c-FOS (Tablica 4.7) nasumično umetnute u dugačke sekvence DNK.

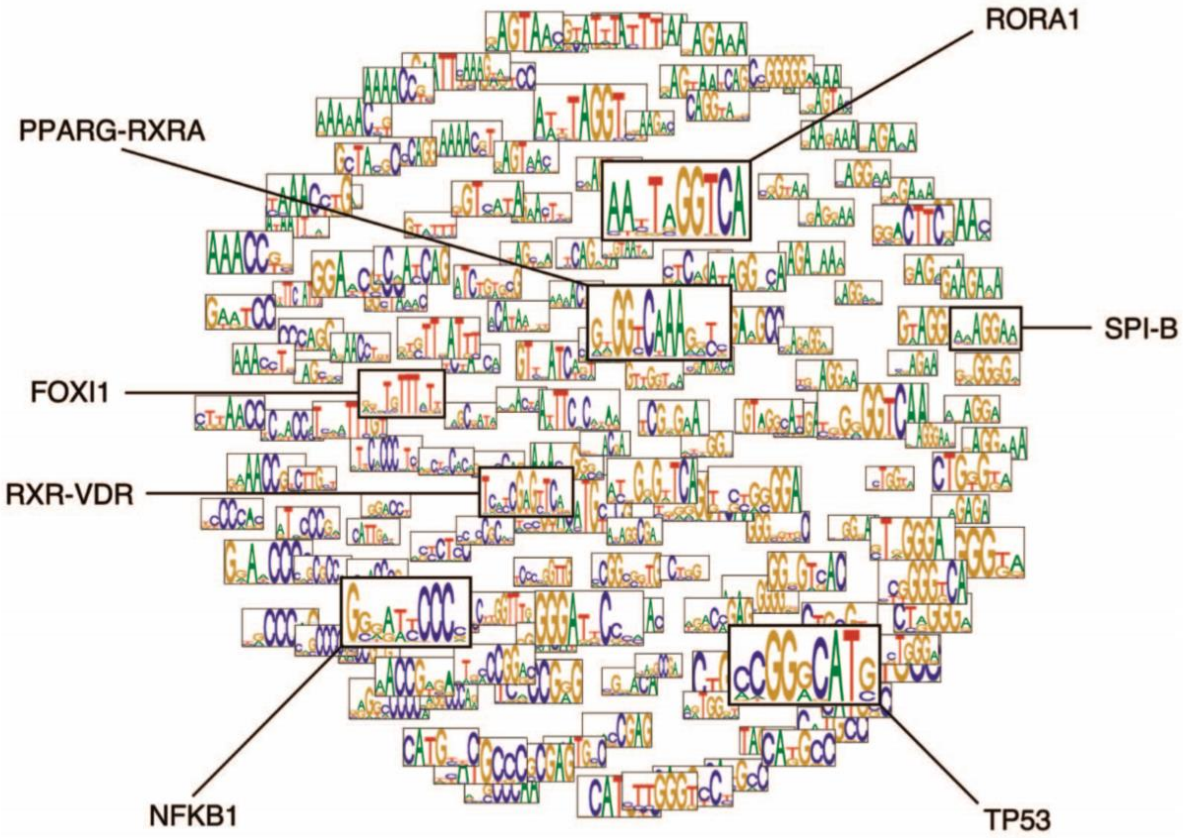
Tablica 4.7 Motivi nad kojima se testira algoritam

Name	ID	Information Content	Size	Sequence Logo
HFH-1	MA0040	14.07	11	
HLF	MA0043	11.15	12	
c-FOS	MA0099	10.67	8	

Prvo ćemo promotriti rad algoritma sa skupom podataka u kojem se nalaze DNK sekvence s umetnutim samo jednim motivom. Algoritam se izvodi 20 puta za svaki sintetički skup podataka. U svakom izvođenju maksimalni broj generacija je 200. Tablica 4.8 pokazuje postotak izvođenja u kojima je pronađen točan motiv, prosječnu duljinu motiva otkrivenih algoritmom i primjer nastalog rješenja za svaki skup podataka.

Tablica 4.8 Rezultati izvođenja algoritma

Motif	Sequence length	Population size	Background set size	Successful runs	Evolved length	Evolved example
HFH-1	1200	1000	500	95%	68%	
HFH-1	2000	2000	1000	95%	75%	
HFH-1	5000	4000	1500	90%	82%	
HLF	600	500	1000	100%	70%	
HLF	1000	1000	1000	90%	76%	
HLF	1500	3000	1500	95%	65%	
c-FOS	500	1000	1000	90%	96%	
c-FOS	1000	4000	1500	60%	75%	
c-FOS	1500	4000	1500	95%	91%	



Slika 4.3 Testiranje na podacima s višestruko umetnutim motivima

Prava moć algoritma vidi se kad ga testiramo na višestruko umetnutim motivima unutar sekvenci. Slika 4.3 prikazuje grupe unutar populacije jednog takvog testiranja. Svaku pojedinu grupu predstavlja najbolji motiv, a relativna dobrota prikazana je veličinom motiva. Način raspoređivanja prikazuje višedimenzionalno skaliranu udaljenost između grupa. Dodatno su uokvirene najbolje particije za pojedini motiv.

Ovi rezultati pokazuju da predstavljeni algoritam uspijeva otkriti motive u relativno dugačkim DNK sekvencama. Za svaki od tri motiva algoritam je uspio pronaći motive ugrađene u prave, zamršene i dugačke DNK sekvence. Na primjeru testiranja s višestruko ugrađenim motivima vidi se da je algoritam uspio zadržati raznovrsnost jedinki između grupa ali i istovremeno optimizirati individualna rješenja.

5. Zaključak

Proučavajući različite primjene i implementacije genetskih algoritama na području bioinformatike došla sam do zaključka da genetski algoritmi mogu biti zaista moćan alat u rješavanju kompleksnih zadataka. Napraviti algoritam koji će ponuditi egzaktno rješenje ponekad je izvedivo, ali u ovim se primjerima često pokazalo da je takve algoritme ili vrlo teško osmisliti, ili su primjenjivi samo na jednostavnije skupove podataka (npr. kraće sekvence DNK). Bilo je zanimljivo vidjeti i kako se neke specifičnosti mogu ugraditi u genetski algoritam kako bi on udovoljio zahtjevima određenog problema. To ostavlja prostor za prilagodbu algoritama i za potencijalno drugačije potrebe u budućnosti.

Pokazalo se da, ovisno o problemu i podacima, genetski algoritmi su u nekim slučajevima nadmašili prethodne algoritme rješavanja istih problema.

Budući da su i evolucijsko računarstvo i bioinformatika prilično mlade znanosti o kojima se još puno istražuje, biti će zanimljivo vidjeti kakvi će se sve problemi u budućnosti moći riješiti, kako iz svake grane zasebno, tako i u njihovoj točki ispreplitanja.

6. Literatura

[1] Golub, M. Genetski algoritam: Prvi dio, skripta. Izdanje 2004. Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

[2] Dunaj, B. Poravnavanje protienskih nizova uz pomoć genetskih algoritama. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2011.

[3] Wong, K. Evolutionary Algorithms: Concepts, Designs, and Applications in Bioinformatics. Department of Computer Science, University of Toronto, Ontario, Canada

[4] *About the Human Genome Project*
http://web.ornl.gov/sci/techresources/Human_Genome/project/index.shtml

[5] Notredame, C., Higgins, D. G. SAGA: sequence alignment by genetic algorithm. Oxfors University Press. Nucleic Acids Research, 1996, Vol. 24, No. 8.

[6] Palensky, M., Ali, H.. A Genetic Algorithm for Simplifying The Amino Acid Alphabet. Coll. of Inf. Sci. & Technol, Nebraska Univ.. Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE

[7] Cannata, N., Toppo, S., Romualdi, C., Valle, G.. Simplifying amino acid alphabets by means of a branch and bound algorithm and substitution matrices. Bioinformatics, 18(8):1102– 1108, 2002.

[8] Lones, M. A., Tyrrell, A. M.. Regulatory Motif Discovery Using a Population Clustering Evolutionary Algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 4, No. 3, Srpanj 2007

[9] Pascarella, S. Argos, P. (1992) Protein Eng., 5, 121–137.

[10] Benjamin Wicks, osobna stranica
http://www.benjaminwicks.com/portfolio/images/01.png

[11] Wikimedia Commons: *Clustal W alignment*
https://commons.wikimedia.org/wiki/File:RPLP0_90_ClustalW_aln.gif

[12] PEDS: protein engineering design and selection
http://peds.oxfordjournals.org/content/13/3/149/F1.expansion.html

7. Sažetak

Genetski algoritmi su algoritmi koji kroz nekoliko faza (inicijalizacija, ocjena dobrote, razmnožavanje) i više generacija iterativno dolaze do konkretnog rješenja problema. U seminaru je prikazan način rada genetskih algoritama kroz konkretne primjene na području bioinformatike. Radi se o problemima poravnavanja višestrukih sekvenci, pojednostavljivanja abecede aminokiselina i pronalaženja motiva u sekvencama DNK. U svakoj od primjena prikazan je konkretan način realizacije svake od faza GA, rezultati testiranja algoritama na nekoliko primjera kao i usporedba s rezultatima drugih algoritama.