

SVEUČILIŠTE U ZAGREBU  
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 3616

**Fraktalna vizualizacija evolucijskim  
algoritmima**

Mirjam Škarica

Zagreb, svibanj 2014.

Umjesto ove stranice umetnите izvornik Vašeg rada.

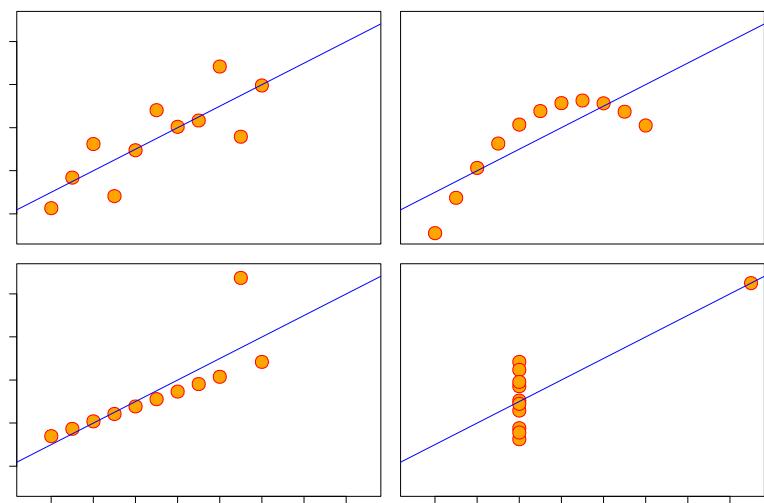


# Sadržaj

1. Uvod.....	1
2. Uvod u bioinformatiku.....	3
3. Uvod u evolucijske algoritme.....	5
4 . IFS fraktali .....	6
4.1. IFS fraktali generirani ulaznim podacima .....	7
4.2. Sintetiziranje DNA podataka Markovljevim modelom.....	9
4.3. Struktura jedinke.....	12
4.4. IFS fraktali generirani evolucijskim algoritmima.....	14
4.4.1 Operator selekcije.....	15
4.4.2 Operator križanja.....	16
4.4.3 Operator mutacije.....	17
4.4.4 Funkcija dobrote.....	17
5. Rezultati.....	20
6. Zaključak.....	24
Literatura.....	25

# 1. Uvod

Jedna od karakteristika današnjeg vremena je svakodnevna proizvodnja i pohrana velikih količina podataka. Područje bioinformatike nije izuzetak. Zato nije iznenadujuće što neprestano tražimo učinkovite načine obrađivanja, reduciranja i prikazivanja podataka u svrhu dolaženja do novih spoznaja. Većinom se to radi tako što se podaci analiziraju nekakvim statističkim modelom koji kao rezultat daje još statističkih podataka. Problem u ovakvom načinu je mogućnost da taj proces ogoli bitne informacije koje podaci kriju u sebi. Ovu ideju ilustrirao je Frank Anscombe 1973. na primjeru Anscombevevog kvarteta<sup>1</sup>. Kvartet se sastoji od 4 skupa podataka po 11 točaka ( $x,y$ ). Skupovi imaju iste, ili gotovo identične statističke vrijednosti (prosjek, devijaciju, korelacije između  $x$  i  $y$ ...), ali izgledaju vrlo različito kada se nacrtaju (slika 1.1). Ljudi brzo i s lakoćom uočavaju vizualne različitosti i uzorke koji su netom prije bili skriveni među statističkim rezultatima. Anscombe je ovim primjerom ilustrirao važnost vizualizacije, uz analizu podataka. To je i misao kojom se ovaj rad vodi.



Slika 1.1 Anscombe's quartet

---

<sup>1</sup> Anscombe's quartet

U nastavku rada opisani su postupci generiranja fraktala iz DNA nizova u nadi da će se iz takvih vizualizacija doći do novih spoznaja. Sljedeća dva poglavlja opisuju osnovne pojmove iz područja bioinformatike, odnosno evolucijskih algoritama. U četvrtom poglavlju detaljno je opisan postupak vizualizacije DNA podataka IFS fraktalima pomoću evolucijskih algoritama. U petom poglavlju dan je pregled i analiza dobivenih rezultata. Na kraju, šesto poglavlje je zaključak s prijedlozima za moguća poboljšanja.

## 2. Uvod u bioinformatiku

Bioinformatika je širok pojam, sljedeća definicija je predložena od strane NCBI<sup>2</sup>.

*Bioinformatika svodi pojmove biologije na razinu makromolekula te potom primjenjuje informatičke tehnike (izvedene iz disciplina kao što su primijenjena matematika, računarska znanost i statistika) kako bi se razumjele i organizirale informacije povezane s tim molekulama, u velikim razmjerima.<sup>3</sup>*

Svaka stanica je zapravo dinamičan sustav koji se sastoji od molekula, kemijskih reakcija i kopije genetskog materijala, odnosno genoma tog organizma. Makromolekule nukleinskih kiselina, proteina i ugljikohidrata su presudne za funkcioniranje svih poznatih živih organizama.

DNA (eng. *deoxyribonucleic acid*) i RNA (engl. *ribonucleic acid*) su nukleinske kiseline. DNA kontrolira aktivnosti u stanici određivanjem enzima i drugih proteina koji će se sintetizirati.

DNA se sastoji od dvostrukе zavojnice koje se sastoje od manjih gradivnih jedinica, nukleotida. Svaki nukleotid se sastoji od fosfata, šećera i nukleinskih baza adenin, citozin, gvanin i timin koje označavamo slovima A, C, G i T. DNA je organizirana u strukture koje nazivamo kromosomima.

RNA obično ima samo jedan lanac, iako postoje i RNA s dva lanca. On se sastoji od šećera, fosfata i nukleinskih baza adenin, citozin, gvanin i uracil koje označavamo sa slovima A, C, G i U.

Proteini su makromolekule sastavljene od jednoga ili više lanaca aminokiselina. One čine većinu biomase organizma i obavljaju različite funkcije npr. ubrzavanje metaboličkih reakcija, umnažanje i prepisivanje DNA, itd.

---

2 The National Center for Biotechnology Information

3 „Bioinformatics is conceptualizing biology in terms of macromolecules (in the sense of physical-chemistry) and then applying "informatics" techniques (derived from disciplines such as applied maths, computer science, and statistics) to understand and organize the information associated with these molecules, on a large-scale.” [1]

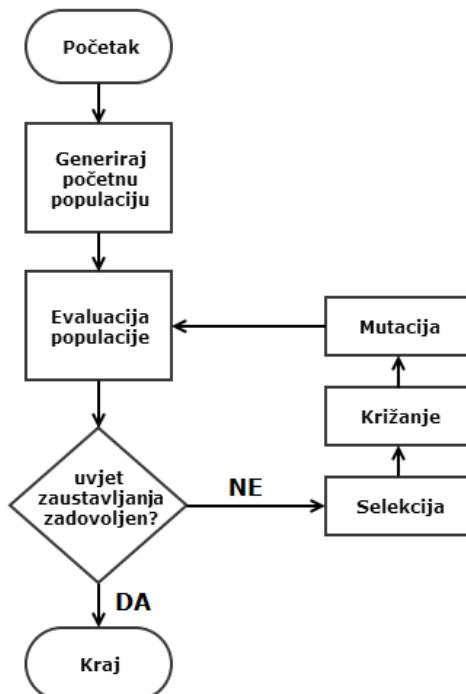
Objašnjene toka genetskih informacija između DNA, RNA i proteina naziva se centralna dogma molekularne biologije.<sup>[2]</sup> Opći prijenos je umnažanje DNA, kopiranje informacije iz DNA u RNA. Ta sekvenca RNA dalje nosi informaciju u obliku tzv. kodona. Kodon je slijed 3 uzastopne nukleinske baze koje određuju umetanje određene aminokiseline u polipeptidni lanac tijekom sinteze proteina, ili signaliziraju početak i prestanak sinteze istih. Postoji 64 različitih kodona, npr. AGU, CUC, GUU.

U korijenu bioinformatike je želja za određivanjem novih gena u stanicama organizama i funkcija proteina koje ti geni kodiraju. Automatizacija DNA sekvenciranja je dovela do nevjerovatnog rasta količine dostupnih podataka. Tradicionalno, funkcije proteina se pokušavaju doznati detaljnim analizama DNA nizova u tekstualnom formatu. To je još uvijek spor proces. Ideja ovog rada je pronaći vizualnu reprezentaciju niza koja bi jednostavno i brzo otkrivala neke informacije, odnosno karakteristike tog niza.

### 3. Uvod u evolucijske algoritme

Evolucijski algoritmi traže rješenje problema simulirajući proces evolucije. Ideju EA je predstavio I. Rechenberg 1960-tih godina u svom djelu „*Evolution strategie*“ („Evolutionsstrategie“). Danas postoje mnoge verzije EA, ali je većina bazirana na istim principima.

Algoritam započinje stvaranjem određenog broja jedinki. Jedinke se generiraju nasumično i svaka od njih predstavlja moguće rješenje problema. Skup svih jedinki jedne generacije čini populaciju. Sljedeći korak je evaluacija populacije koja se radi pomoću funkcije dobrote. Funkcija dobrote svakoj jedinki pridružuje faktor dobrote, s tim da bolje jedinke imaju veći faktor dobrote. Izdvajajući određeni broj rješenja (jedinki), formira se nova populacija. Nova populacija se mijenja operatorima križanja i mutacije. Algoritam ponavlja ovaj proces sve dok nije pronađeno rješenje problema, odnosno dok jedna jedinka nije zadovoljila uvjete evolucije. Opisani postupak predstavljen je grafički dijagramom 3.1.



Dijagram 3.1

## 4 . IFS fraktali

Fraktal, kao pojam, skovao je Benoit Mandelbrot 1975. Fraktal je prirodna tvorevina, a opisujemo ga kao geometrijski uzorak koji se ponavlja (barem približno) na svim skalama umanjenosti tvoreći nepravilne oblike i površine koje se ne mogu predstaviti klasičnom geometrijom. Drugim riječima, fraktali su samoslični bilo da ih gledamo iz bliza ili iz daleka. Osobito se koriste u računalnom modeliranju nepravilnih uzoraka i struktura u prirodi.

IFS (engl. *iterated function systems*) su metode konstrukcije fraktala. Rezultirajuće konstrukcije su uvijek samoslične. IFS fraktali mogu biti bilo koje dimenzije, ali se najčešće računaju i crtaju u 2D. U principu, koristi se skupina jednostavnih transformacija kao što su rotacija, skaliranje i translacija kako bi se pomicala točka. Orbita koju dobijemo iterativnom primjenom definiranih transformacija na neku početnu točku je upravo IFS fraktal.

Želimo li omeđenu orbitu, odnosno fraktal konačne površine, ne mogu se koristiti bilo kakve transformacije, već samo one za koje vrijedi da se za bilo koji par točaka  $(p,q)$  njihova međusobna udaljenost smanjuje s primjenom te transformacije. Takve transformacije nazivamo kontrakcijskim mapama (engl. *contraction maps*). Formalno,  $f: \mathbb{R}^2 \rightarrow \mathbb{R}^2$  je kontrakcijska mapa ako vrijedi:

$$d(p, q) \geq d(f(p), f(q)), \quad \forall p, q \in \mathbb{R}^2 \quad (4.1)$$

Jedan od najpoznatijih primjera je IFS za trokut Sierpinskog. Neka su dane sljedeće transformacije (svaki redak predstavlja jednu od transformacija).

$a$	$b$	$c$	$d$	$e$	$f$	$p_i$
0.5	0.0	0.0	0.5	0.0	0.0	1/3
0.5	0.0	0.0	0.5	1.28	0.0	1/3
0.5	0.0	0.0	0.5	0.64	0.8	1/3

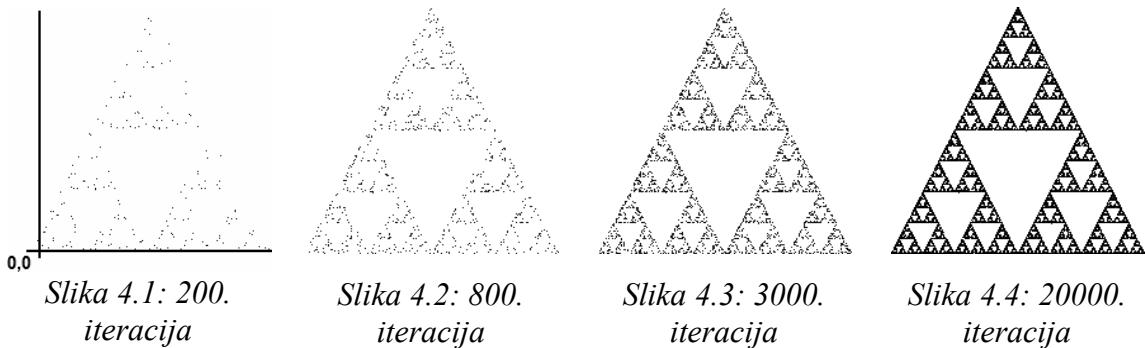
Tablica 4.1: transformacije za trokut Sierpinskog

Ove transformacije točku  $T_i = (x_i, y_i)$  preslikavaju u točku  $T_{i+1} = (x_{i+1}, y_{i+1})$ , gdje se  $x_{i+1}, y_{i+1}$  računaju pomoću funkcija:

$$x_{i+1} = a \cdot x_i + b \cdot y_i + e \quad (4.2)$$

$$y_{i+1} = c \cdot x_i + d \cdot y_i + f \quad (4.3)$$

Fraktal, u ovom slučaju trokut Sierpinskog, dobijemo tako da na početnu točku iterativno primjenjujemo jednu od 3 transformacije iz tablice 4.1, s vjerojatnošću  $p_i$ . Općenito mora vrijediti  $\sum_{i=1}^n p_i = 1$ . Počevši od  $T_0 = (0,0)$ , i ponavljajući postupak preslikavanja velik broj puta dobiju se rezultati na slikama 4.1- 4.4.



Lako je zaključiti kako se korištenjem drugih vrijednosti koeficijenata, vjerojatnosti ili broja transformacija mogu dobiti vrlo različiti rezultati, pa time i fraktali.

## 4.1. IFS fraktali generirani ulaznim podacima

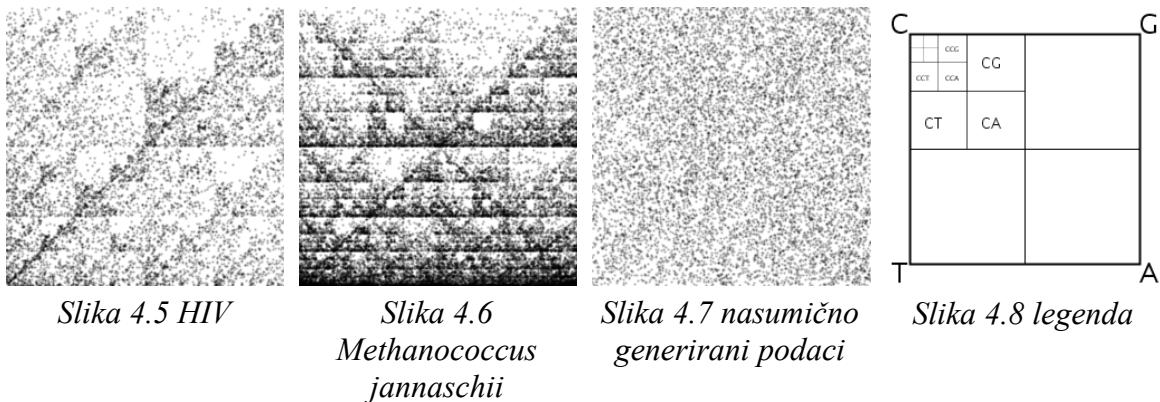
Na primjeru IFS za trokut Sierpinskog je pokazan način generiranja fraktala pomoću transformacija čiji izbor određuje slučajnost, odnosno njihove vjerojatnosti. Ovaj rad se nadalje bavi generiranjem fraktala pomoću transformacija čiji izbor određuju ulazni podaci. Da povučemo paralelu s DNA podacima, najjednostavniji primjer bio bi da imamo definirane neke 4 transformacije, da nam je ulazni niz upravo niz nukleinskih baza (A, C, G, T) te da pojava svake od baza uvijek rezultira primjenom odgovarajuće transformacije.

Algoritam bi tada slijedno prolazio kroz niz te na točku primjenjivao transformaciju koja odgovara zadnjem pročitanom znaku,  $znak \in \{A, C, T, G\}$ . Primjer<sup>4</sup> ovih transformacija prikazan je tablicom 4.2.

$a$	$b$	$c$	$d$	$e$	$f$	$znak$
0.5	0.0	0.0	0.5	0.0	0.5	A
0.5	0.0	0.0	0.5	0.5	0.5	T
0.5	0.0	0.0	0.5	0.5	0.0	G
0.5	0.0	0.0	0.5	0.0	0.0	C

Tablica 4.2: Transformacije za IFS fraktal generiran DNA nizom

Što ove transformacije rade možemo lakše predočiti slikom 4.8. Svaka nukleinska baza "privlači" točku u svoj kut, tako da trenutnu točku preslika u novu na pola puta između "svog" kuta i trenutne pozicije točke. Na slikama 4.5,4.6 prikazani su IFS fraktali generirani ulaznim podacima *HIV* genoma, *Methanococcus jannaschii* genoma te s nasumično generiranim DNA podacima (slika 4.7), radi usporedbe.



Zanimljivo je uočiti uzorce dijagonala, odnosno izostanak istih te tzv. *double scoop* koji se može lako primijetiti na slikama 4.5 i 4.6 kao relativno prazan prostor u svakom podkvadrantu GC. Pojava *double scoop* uzorka je prvi put zabilježena u ljudskoj beta-globin regiji i ukazuje na relativnu rijetnost uzastopne pojave gvanina i citozina.<sup>[3]</sup>

Upravo ovo brzo i jednostavno dolaženje do spoznaja je glavni motiv za

---

4 U literaturi se može pronaći pod nazivima *DNA driven four-cornered chaos game* ili *chaos game representation algorithm*.

vizualizaciju DNA podataka. To je također i motiv za daljnji pokušaj pronađaska drugih korisnih skupova transformacija koje bi nam s lakoćom otkrivale različite informacije.

## 4.2. Sintetiziranje DNA podataka Markovljevim modelom

Jedan od mogućih problema koji se mogu javiti je kratak ulazni DNA niz. Naravno, mogli bismo proći kroz dani niz više puta i spremati broj pojavljivanja svake dobivene točke te zatim tu informaciju koristiti kao mjeru učestalosti nekog uzorka. Ali, u praksi se u ovu svrhu koriste Markovljevi lanci koji su specijalni slučaj Markovljevih procesa.

Markovljevi procesi su oni stohastički (slučajni) procesi čije buduće stanje ovisi samo o trenutnom stanju. To svojstvo zovemo svojstvo odsustva pamćenja. Markovljevi procesi mogu imati diskretan ili kontinuiran skup stanja. Procese s diskretnim stanjima nazivamo lancima.

Neka Markovljev lanac  $X_0, X_1, X_2, \dots, X_n$  može poprimiti vrijednosti iz skupa diskretnih stanja  $S = \{s_0, s_1, s_2, \dots\}$ . Skup stanja može biti beskonačan ili konačan. Oznaka  $X_n = i, i \in S$  neka znači da se Markovljev lanac u  $n$ -tom koraku nalazi u stanju  $i$ . Za opći Markovljev lanac vrijedi sljedeće:

$$P(X_{n+1}=j | X_0=i_0, X_1=i_1, \dots, X_n=i_n) = P(X_{n+1}=j | X_n=i_n) \quad (4.4)$$

za svaki  $n \geq 0$  i za sve  $i_0, \dots, i_{n-1}, i, j \in S$ .

Svojstvo u relaciji (4.4) je upravo svojstvo odsustva pamćenja, odnosno Markovljevo svojstvo. Prepostavimo da se nalazimo u vremenskom trenutku  $n$ . Tada vrijeme  $n+1$  predstavlja neposrednu budućnost, dok vremena  $0, 1, \dots, n-1$  predstavljaju prošlost. Markovljevo svojstvo nam govori da je ponašanje Markovljevog lanca u neposrednoj budućnosti, uvjetno na sadašnjost i prošlost, jednako ponašanju Markovljevog lanca u neposrednoj budućnosti, uvjetno samo na sadašnjost. Prijelaz iz stanja  $i$  u koraku  $n-1$  u stanje  $j$  u koraku  $n$  opisan je prijelaznom vjerojatnošću:

$$p_{ij}^{(n)} = P(X_n=j | X_{n-1}=i) \quad (4.5)$$

U općem slučaju ta se vjerojatnost mijenja ovisno o koraku  $n$ .

Nas će zanimati samo homogeni Markovljevi lanci. To su oni za koje prijelazne vjerojatnosti ne ovise o koraku, odnosno vremenu  $n \geq 1$ . Vrijedi da je:

$$p_{ij}^{(n)} = p_{ij}, \quad \forall i, j, n \quad (4.6)$$

U vezi s tim, uvodimo pojam stohastičke matrice. Matrica  $P = (p_{ij} : i, j \in S)$  se naziva stohastičkom matricom ako je  $p_{ij} \geq 0, \forall i, j \in S$  te ako vrijedi:

$$\sum_{j \in S} p_{ij} = 1, \quad \forall i \in S \quad (4.7)$$

Ako je broj stanja u  $S$  konačan, tada je  $P$  „prava“ (konačna) matrica. S druge strane, ako je  $S$  beskonačan skup, tada će  $P$  biti beskonačna matrica.

Ideja je pomoći stohastičke matrice opisati vjerojatnosti pojavljivanja pojedinih uzorka u ulaznom DNA nizu te pomoći tih vjerojatnosti generirati DNA niz proizvoljne duljine koji bi bio vjerna reprezentacija ulaznog niza. U mnogim slučajevima, kao što je i naš, nije lako uočiti i odrediti prijelazne vrijednosti između stanja. U tu svrhu koristimo skrivene Markovljeve modele koji opisuju statističku vezu između promatranog niza i skupa stanja  $S$ .

Markovljev model  $k$ -toga reda DNA niza se dobije tako što se za svaki mogući podniz duljine  $k$  izračuna kolika je vjerojatnost da ga slijedi baza C, G, A, odnosno T.<sup>[4]</sup> To se radi na sljedeći način. Ulagni niz se čita redom, za svaki podniz duljine  $k$ , računa se koliko puta je taj podniz slijedila svaka od četiri baze. Vjerojatnosti se dobiju dijeljenjem tih brojeva s ukupnim brojem pojavljivanja pojedinog podniza. U slučaju da se neki od podnizova ne pojavi, koriste se jednake vjerojatnosti za sve četiri baze, odnosno  $p_{podnizC} = p_{podnizG} = p_{podnizA} = p_{podnizT} = 0.25$ .

Ilustrirajmo postupak na primjeru ulaznog niza:

AGCAACTAGGCCACCCGGACTACTACCTGCAGGTCCCTAGCATGTATCAA

Kako bi izračunali Markovljev model 2. reda ( $k=2$ ) prolazimo slijedno kroz niz koristeći klizeći prozor veličine 2 kako je prikazano na ilustraciji 4.1

AG	CAACTAGGCCACCCGGACTACTACCTGCAGGTCCCTAGCATGTATCAA
A	GC AACTAGGCCACCCGGACTACTACCTGCAGGTCCCTAGCATGTATCAA
AG	CA ACTAGGCCACCCGGACTACTACCTGCAGGTCCCTAGCATGTATCAA

*Ilustracija 4.1*

Za svaki trenutni podniz unutar klizećeg prozora broje se ponavljanja sljedeće nukleinske baze. Rezultat toga za ovaj konkretni primjer prikazan je tablicom 4.3. Sve dobivene vjerojatnosti prijelaza prikazane su tablicom 4.4.

podniz	N <sub>C</sub>	N <sub>G</sub>	N <sub>A</sub>	N <sub>T</sub>
CC	2	1	1	2
CG	0	1	0	0
CA	1	1	2	1
CT	0	1	4	0
GC	1	0	3	0
GG	1	0	1	1
GA	1	0	0	0
GT	1	0	1	0
AC	2	0	0	3
AG	1	2	0	0
AA	1	0	0	0
AT	1	1	0	0
TC	1	0	1	0
TG	1	0	0	1
TA	2	2	0	1
TT	0	0	0	0

Tablica 4.3

podniz	P <sub>C</sub>	P <sub>G</sub>	P <sub>A</sub>	P <sub>T</sub>
CC	0.333	0.167	0.167	0.333
CG	0.0	1.0	0.0	0.0
CA	0.2	0.2	0.4	0.2
CT	0.0	0.2	0.8	0.0
GC	0.25	0.0	0.75	0.0
GG	0.333	0.0	0.333	0.333
GA	1.0	0.0	0.0	0.0
GTs	0.5	0.0	0.5	0.0
AC	0.4	0.0	0.0	0.6
AG	0.333	0.667	0.0	0.0
AA	1.0	0.0	0.0	0.0
AT	0.5	0.5	0.0	0.0
TC	0.5	0.0	0.5	0.0
TG	0.5	0.0	0.0	0.5
TA	0.4	0.4	0.0	0.2
TT	0.25	0.25	0.25	0.25

Tablica 4.4  
Markovljev model 2. reda za dani primjer

Sada, uz izračunati Markovljev model, možemo generirati tzv. sintetičke DNA nizove proizvoljne duljine po algoritmu:

Za početnu vrijednost klizećeg prozora slučajno izabratи podniz duljine  $k$  iz ulaznog DNA niza

Dok novi niz nije željene duljine:

generiraj novu nukleinsku bazu s obzirom na trenutni sadržaj klizećeg prozora koristeći prijelazne vjerojatnosti

pomakni klizeći prozor za jedno mjesto

Primjer generiranog DNA niza duljine 100 za dani ulazni primjer i klizeći prozor veličine 2:

```
ATCATCATCAGGACTACCCTAGGTACCCAACTAGCAGGCCACTAGGCAGG  
ACTAGGCATCCTGCCCGGCCGGTCAACCAGGTAGGACCTACCCGGACCT
```

Radi vjerodostojnosti podataka, dalje u radu koriste se Markovljevi modeli reda 6.

### 4.3. Struktura jedinke

Spomenimo još jednom kako svaka jedinka u evolucijskim algoritmima predstavlja jedno moguće rješenje problema. Zato je izbor i modeliranje strukture jedinke jedno od najvažnijih koraka.

Idealno bi bilo kada bi ista fraktalna reprezentacija radila na jednak način s DNA podacima, proteinima i kodonima. Iako se svi ovi oblici mogu dobiti iz bilo kojeg drugog, svaki od njih se pojavljuje u drugom stadiju biološkog procesa. Sirov DNA ima najviše informacija, ali najmanji stupanj interpretabilnosti, dok dijeljenjem DNA podataka u kodone, ona postaje interpretabilnija (npr. kodon sadrži informaciju o termalnoj stabilnosti DNA)<sup>[5]</sup>. Dalje u radu, po prijedlogu [Ashclock, Golden], izbor transformacije će ovisiti o kodonima koji dijele DNA niz u 64 moguće trojke.

Jedinka se sastoji od 2 strukture podataka:

- liste transformacija,
- liste duljine 64 čiji elementi sadrže redne brojeve transformacija

Lista transformacija je prikazana tablicom 4.5. Transformacije su definirane rednim brojem i s još 4 realna broja:

- kutom rotacije  $\Theta$  izraženog u radijanima,
- translacijom po x-osi  $\Delta x$  ( $-1 \leq \Delta x \leq 1$ ),
- translacijom po y-osi  $\Delta y$  ( $-1 \leq \Delta y \leq 1$ ),
- faktorom skaliranja s ( $0 \leq s \leq 1$ ).

Sukladno tome, točka  $T_i = (x_i, y_i)$  se preslikava u točku  $T_{i+1} = (x_{i+1}, y_{i+1})$ , gdje se  $x_{i+1}, y_{i+1}$  računaju pomoću funkcija:

$$x_{i+1} = s \cdot (x_i \cdot \cos \theta - y_i \cdot \sin \theta + \Delta x) \quad (4.8)$$

$$y_{i+1} = s \cdot (x_i \cdot \sin \theta + y_i \cdot \cos \theta + \Delta y) \quad (4.9)$$

Drugi dio strukture jedinke je prikazan tablicom 4.6. Lista je duljine 64 jer svaki element odgovara točno jednom mogućem kodonu, a sadržaj svakog elementa je redni broj transformacije koja će se primijeniti u slučaju nailaska na dotični kodon. Npr. gledajući tablicu 4.6. ako algoritam pročita kodon "CCG", na trenutnu točku primijenit će se 2. transformacija.

rbr.	kut rotacije u radijanima ( $\Theta$ )	translatacija po x-osi ( $\Delta x$ )	translatacija po y-osi ( $\Delta y$ )	skaliranje (s)
1	6.0176	0.8754	0.5239	0.9086
2	1.0304	0.2503	-0.657	0.9729
3	5.9167	-0.525	0.1340	0.6029
4	0.1123	0.3145	0.5260	0.6262
5	0.3586	0.5329	0.3257	0.5669
6	5.7666	0.4117	-0.817	0.9305
7	0.4247	0.9795	0.9789	0.9910
8	3.0488	-0.055	-0.881	0.9690

Tablica 4.5 Primjer liste transformacija

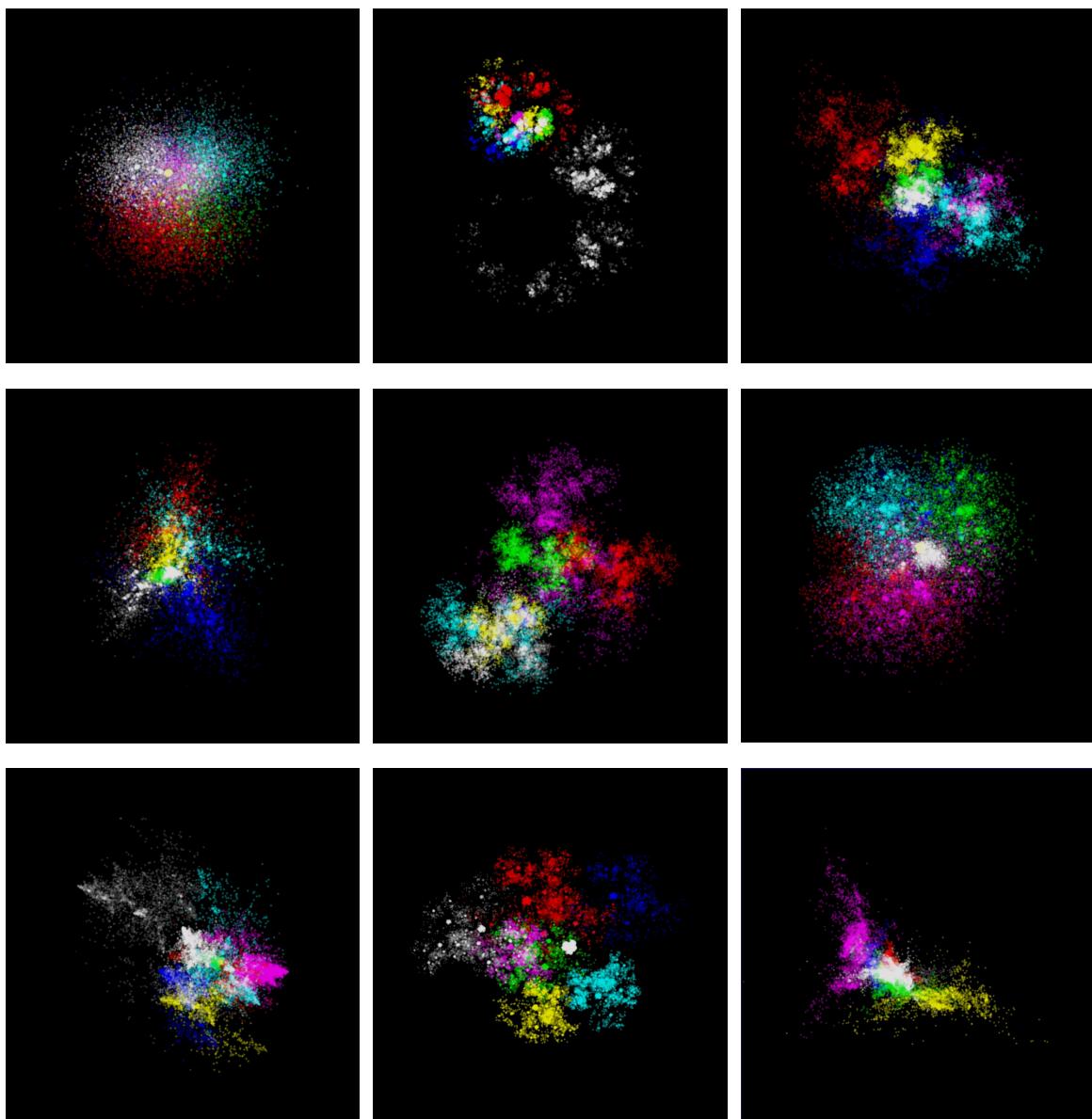
CCC	CCG	CCA	CCT	CGC	CGG	CGA	CGT	...	TTA	TTT
8	2	2	7	4	4	1	3	...	5	3

Tablica 4.6 Primjer liste s koja određuje odnos kodon → transformacija

#### 4.4. IFS fraktali generirani evolucijskim algoritmima

Ukratko, za vizualizaciju DNA podataka IFS fraktalima, prvo koristimo Markovljev model 6. reda kako bismo sintetizirali proizvoljno dug i reprezentativan DNA niz. Taj se niz zatim dijeli u kodone. Prolazimo slijedno kodone te, pomoću informacija sadržanih u strukturama jedinke, biramo i primjenjujemo odgovarajuće transformacije na pomicnu točku. Trag pomicne točke je upravo IFS fraktal.

Na slici 4.9 se mogu vidjeti fraktali generirani opisanim postupkom.



Slika 4.9 Fraktali dobiveni pomoći jedinki sa slučajno odabranim vrijednostima. Ovakav prikaz je dobiven tako da je svakoj od transformacija pridružena jedna boja.

Svaki on njih je dobiven pomoću jedne jedinke čije su vrijednosti slučajno odabrane. Za svaki se koristio isti ulaz, niz duljine 100 000 s nasumično generiranim elementima iz skupa  $\{C, G, A, T\}$ . Kada bismo ponovili ovaj postupak s konkretnim DNA podacima, vjerojatnost je da bismo dobili jako slične rezultate. To nije pretjerano korisno, s obzirom na to da nas zanimaju informacije koje bismo mogli dobiti iz vizualizacije.

Želimo takav skup transformacija koji će za različite DNA nizove generirati različite fraktale. Iz takvih vizualizacija bi se zatim dalo nešto zaključiti o ulaznom nizu. Za pretragu prostora mogućih rješenja problema koristit ćemo evolucijske algoritme. U tu svrhu moramo definirati operatore selekcije, križanja, mutacije te funkciju dobrote koja svakoj jedinci pridružuje ocjenu kvalitete. Tijek evolucijskog algoritma prikazan je dijagramom 3.1.

#### 4.4.1 Operator selekcije

Selekcija je mehanizam očuvanja dobrih svojstava pojedinih rješenja, odnosno jedinki, iz generacije u generaciju. Postupkom selekcije odabiru se jedinice iz trenutne populacije koje će sudjelovati u sljedećem koraku algoritma, a ostale se odbacuju.

Postoje mnoge varijante operatora selekcije. U ovom radu su implementirane:

- $k$ -turnirska selekcija s mogućnošću elitizma
- eliminacijska proporcionalna selekcija

$K$ -turnirska selekcija ( $k = 2, 3, 4, \dots, N$ , gdje je  $N = \text{velicina populacije}$ ) s jednakom vjerojatnošću odabire  $k$  jedinki iz trenutne populacije. Izabere se najbolja među njima te se kopira u međupopulaciju. Ovaj se postupak ponavlja  $N$  puta, nakon čega međupopulacija postaje populacija koja nastavlja dalje. Nedostatak ove selekcije je generiranje duplikata jedinki i mogućnost gubitka najbolje jedinke. U ovom radu je implementirana 2-turnirska selekcija s mogućnošću elitizma (očuvanja najbolje jedinke).

Eliminacijska proporcionalna selekcija radi na suprotnom principu. Umjesto da bira dobre jedinice koje nastavljaju dalje, odabire loše koje se eliminiraju i zamjenjuju novima. Svakoj jedinci pridružuje se vjerojatnost da bude

odabrana, odnosno eliminirana. Vjerojatnost eliminacije  $i$ -te jedinke  $p_i$  je obrnuto proporcionalna funkciji dobrote  $d$  i iznosi:

$$p_i = \frac{d_{\max} - d_i}{N \cdot d_{\max} - \sum_{i=1}^N d_i} \quad (4.10)$$

gdje je  $N$  veličina populacije,  $d_i$  dobrota  $i$ -te, a  $d_{\max}$  dobrota najbolje jedinke.

Iz (4.10) se vidi da je vjerojatnost odabira najboljeg rješenja jednaka nuli pa je time i vjerojatnost eliminacije istog jednaka nuli. Ovim je riješen problem mogućnosti gubitka najbolje jedinke. Također je riješen i problem pojavljivanja duplikata jer se jednom eliminirana jedinka ne može više odabratи.

#### 4.4.2 Operator križanja

Operator križanja mijenja i razmjenjuje strukturu jedinke s ostalima u populaciji. Izabiru se dvije jedinke, *roditelji*. Miješanjem njihovih svojstava nastaju jedna ili dvije nove jedinke, *djeca*. Budući da *djeca* nasljeđuju svojstva *roditelja*, ako su *roditelji* bili dobri, velika je vjerojatnost da će i *djeca* biti dobra, ako ne i bolja.

Jedna vrsta operatora križanja je križanje s  $n$  točaka prekida. Općenito, slučajno se izabere  $n$  točaka unutar *roditelja* koje svakog od njih podijele na  $n+1$  dijelova. *Djeca* nastaju naizmjeničnim kombiniranjem njihovih dijelova.

U radu se koriste dva neovisna operatora križanja, po jedan za svaku od struktura jedinke. Prvo se iz populacije slučajno odaberu dvije jedinke nad kojima zatim slijedno djeluju ta dva operatora.

Za prvu strukturu, listu transformacija (tablica 4.5), koristi se križanje s jednom točkom prekida. Slučajno se odabire točka prekida  $k$  iz skupa  $\{1, 2, \dots, m\}$ ,  $m = \text{broj transformacija}$ . Prvo dijete nastaje kopiranjem prvih  $k$  transformacija prvog roditelja i preostalih  $n-k$  drugog roditelja. Drugo dijete nastaje analogno, ali obrnutim redoslijedom kopiranja.

Za drugu strukturu (lista rednih brojeva transformacija, tablica 4.6) koristi se križanje s dvije točke prekida. Slučajno se odabiru točke prekida  $k, l$  iz skupa  $\{1, 2, \dots, 64\}$ . Prvo dijete nastaje kopiranjem prvih  $k$  elemenata prvog roditelja, sljedećih  $l$  elemenata drugog i preostalih  $64-k-l$  ponovno od prvog roditelja. Drugo dijete nastaje analogno, ali obrnutim redoslijedom kopiranja.

#### 4.4.3 Operator mutacije

Operator mutacije radi male i slučajne promjene na strukturi slučajno odabrane jedinke. Koristi se kako bi se povećala raznolikost populacije, ali je također bitan i za obnavljanje izgubljenog genetskog materijala.

Kao i kod križanja, u radu se koriste dva neovisna operatora mutacije, po jedan za svaku od struktura jedinke. Kod mutacije prve strukture (tablica 4.5), slučajno se odabere jedna od transformacija i jedan od parametra iz skupa  $\{\Theta, \Delta x, \Delta y, s\}$ . Tom parametru se pribraja slučajno odabrana vrijednost iz intervala [-0.1, 0.1]. Faktor skaliranja se drži unutar granica [0, 1] tako što se u slučaju  $s > 1$ , vrijednost  $s$  zamijeni sa  $2 - s$ , a u slučaju  $s < 0$  sa  $-s$ .

Operator mutacije druge strukture jedinke (tablica 4.6) slučajno izabere jedan element liste i njegov sadržaj izmijeni slučajno odabranom vrijednošću iz skupa  $\{1, 2, \dots, m\}$ ,  $m = \text{broj transformacija}$ .

#### 4.4.4 Funkcija dobrote

Funkcija dobrote<sup>5</sup> je ocjena kvalitete jedinke. Što je jedinka kvalitetnija, to ima veću vjerojatnost preživljavanja. Ne postoji univerzalan oblik funkcije dobrote, već se ona definira posebno za svaki problem. Ona je ključna za postupak selekcije tijekom procesa evolucije jer temeljem njene vrijednosti algoritam samostalno odlučuje koje jedinke zadržati, a koje odbaciti. Iz tog razloga, definiranje funkcije dobrote je jedan od najzahtjevnijih koraka u modeliranju evolucijskog algoritma i o njemu ovisi uspješnost algoritma.

Budući da želimo različit izgled fraktala za različite ulazne podatke, pokušat ćemo pronaći takve jedinke koje će generirati vizualno razdvojene skupove točaka, odnosno fraktala. Dobrota jedinke ovisit će o dva proizvoljno zadana DNA niza. Ideja je sljedeća. Jedinka se podvrgne procesu generiranja IFS fraktala za svaki ulazni niz posebno, uz jednu iznimku. Pomicanje točke uslijed primjene transformacija se ne iscrtava, već se njena pozicija  $T_i = (x_i, y_i)$  prati i pamti. Pri završetku iteracije svakog od ulaznih nizova, računa se prosječna pozicija pomicne točke  $\mu^{ulaz} = (\mu_x^{ulaz}, \mu_y^{ulaz})$ .

---

<sup>5</sup> U literaturi se koriste i nazivi funkcija cilja, *fitness* funkcija itd.

S obzirom na prosječne pozicije pomične točke u radu su implementirane i isprobane tri verzije funkcije dobrote:

- euklidska udaljenost,
- *manhattan* udaljenost,
- linearna interpolacija gornje navedenih udaljenosti.

Euklidska udaljenost točaka  $p = (p_1, p_2, \dots, p_n)$  i  $q = (q_1, q_2, \dots, q_n)$  jednaka je duljini dužine koja ih spaja. U  $n$ -dimenzionalnom prostoru definirana je formulom

$$d_{eucl}(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (4.11)$$

U našem 2-dimenzionalnom slučaju (slika 4.10) vrijedi:

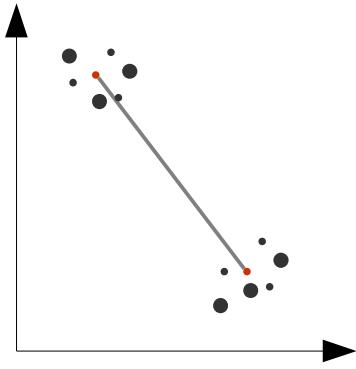
$$d_{eucl}(\mu^{ulaz1}, \mu^{ulaz2}) = \sqrt{(\mu_x^{ulaz1} - \mu_x^{ulaz2})^2 + (\mu_y^{ulaz1} - \mu_y^{ulaz2})^2} \quad (4.12)$$

*Manhattan*<sup>6</sup> udaljenost točaka  $p$  i  $q$  u  $n$ -dimenzionalnom prostoru definirana je:

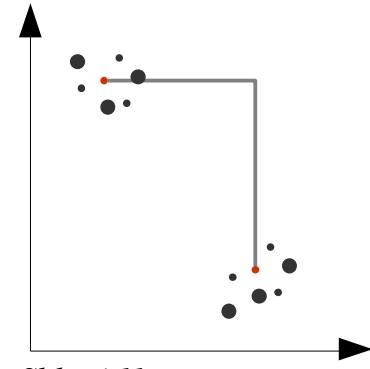
$$d_{manh}(p, q) = \sum_{i=1}^n |q_i - p_i| \quad (4.13)$$

U našem slučaju (slika 4.11) vrijedi:

$$d_{manh}(\mu^{ulaz1}, \mu^{ulaz2}) = |\mu_x^{ulaz1} - \mu_x^{ulaz2}| + |\mu_y^{ulaz1} - \mu_y^{ulaz2}| \quad (4.14)$$



Slika 4.10  
Euklidska udaljenost



Slika 4.11  
Manhattan udaljenost

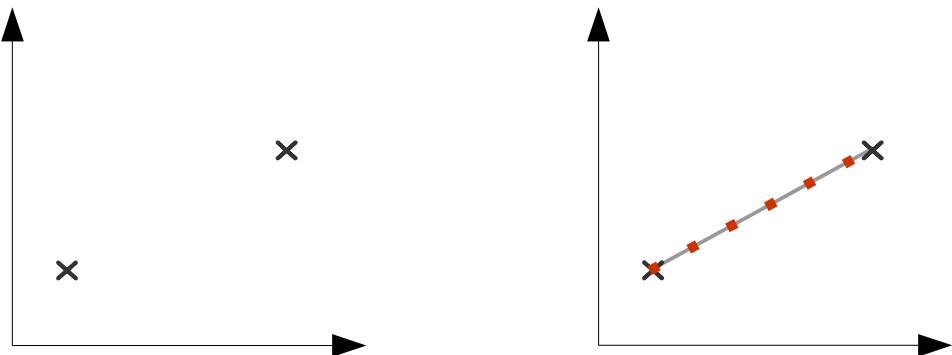
Općenito, linearna interpolacija uzima dvije točke i pronalazi međuvrijednosti koje leže na pravcu između te dvije točke (slika 4.12). Mi želimo interpolirati vrijednosti

---

6 Također zvana  $L_1$  udaljenost *city block distance*, *taxicab metric* itd.

euklidske i *manhattan* udaljenosti ( $d_{eucl}, d_{manh}$ ) prosječnih pozicija pomicne točke  $\mu^{ulaz1}, \mu^{ulaz2}$ . Uvodimo parametar  $t$  koji poprima vrijednosti iz intervala [0,1] te računamo linearnu interpolaciju  $l$  udaljenosti prema formuli:

$$l(\mu^{ulaz1}, \mu^{ulaz2}) = (1-t) \cdot d_{eucl} + t \cdot d_{manh} \quad (4.15)$$

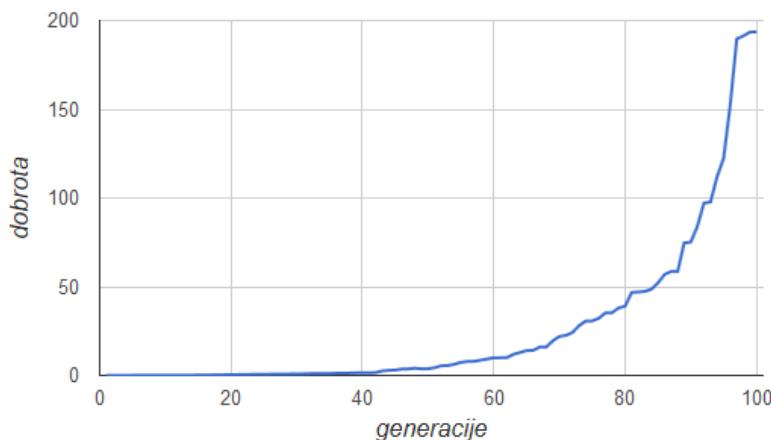


Slika 4.12 primjer linearne interpolacije

## 5. Rezultati

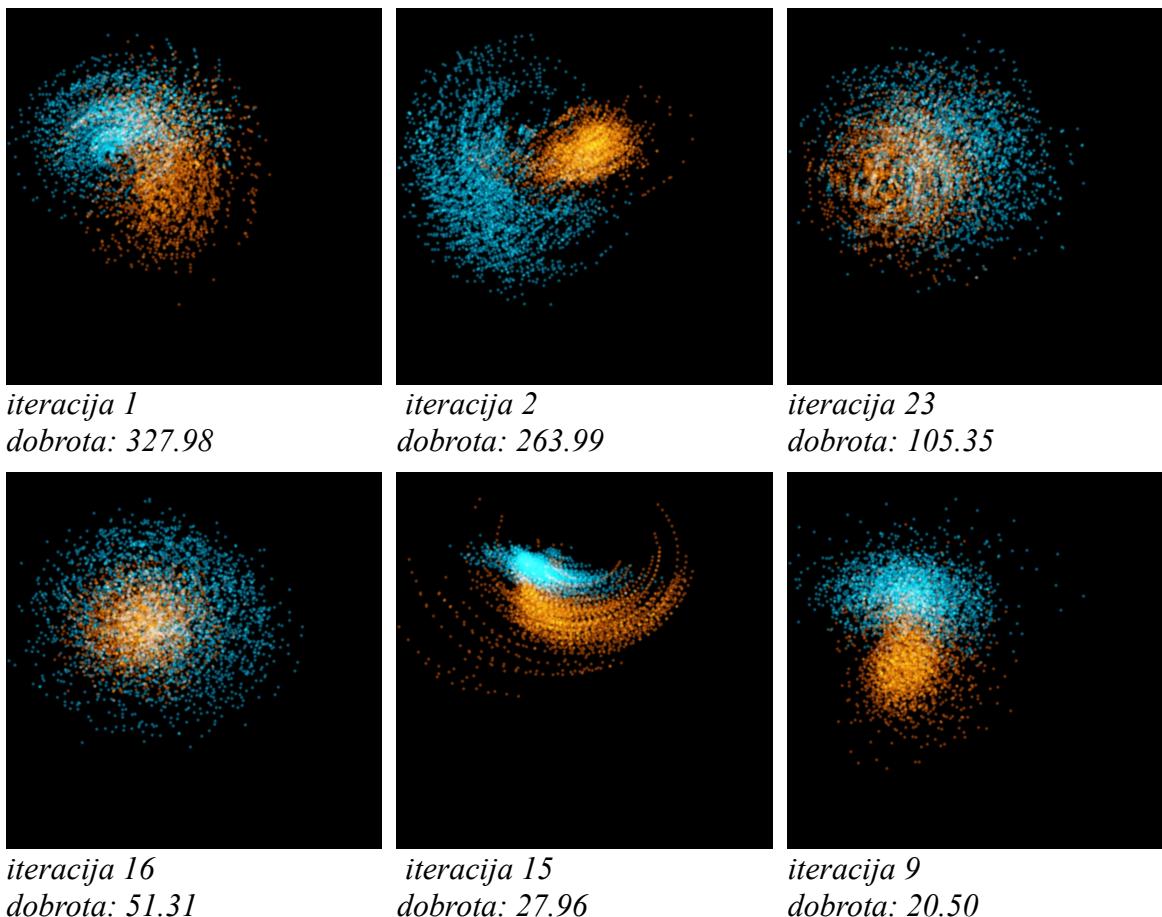
U nastavku su prikazani rezultati dobiveni korištenjem svake od opisanih funkcija dobrote. Za svaku od njih pokrenut je evolucijski algoritam 30 puta s populacijom veličine 100, maksimalnim brojem generacija 100 i faktorom mutacije 0.2. Kao ulazni nizovi koristili su se DNA nizovi *HIV* genoma i *Methanocaldococcus Jannaschii* genoma duljine 10000 generirani Markovljevim modelom 6. reda.

Prosječno kretanje najbolje jedinke kroz 100 generacija korištenjem euklidske udaljenosti za računanje dobrote prikazano je slikom 5.1



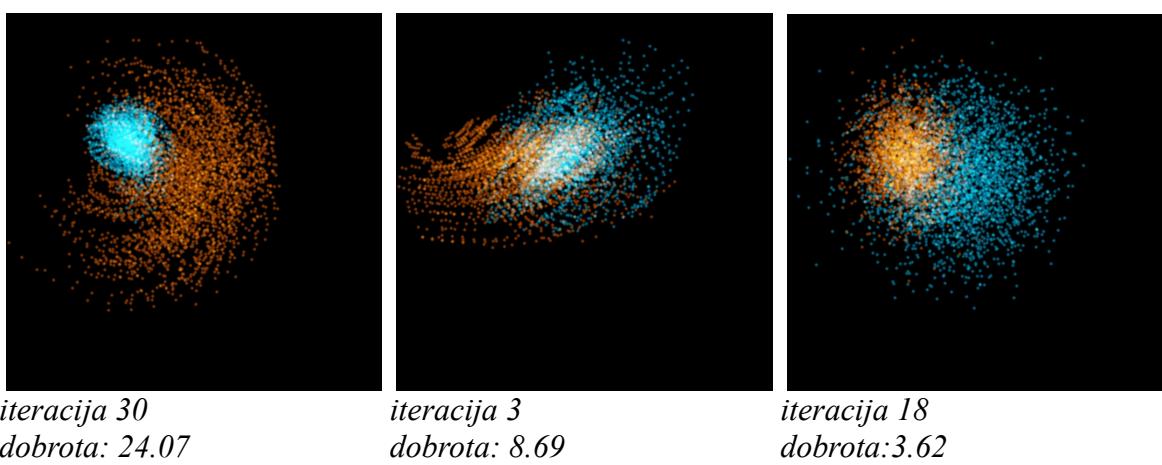
Slika 5.1 prosječno kretanje dobrote najbolje jedinke za pet pokretanja

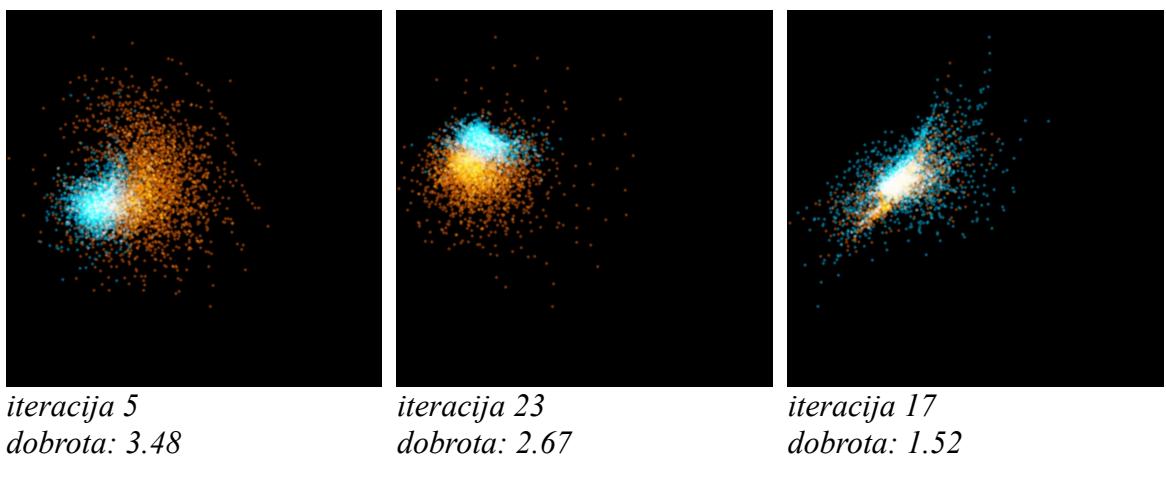
Rezultati dobiveni korištenjem euklidske udaljenosti mogu se vidjeti na slikama 5.2. Na svakoj slici su prikazana dva fraktala različitih boja radi lakšeg razlikovanja. Fraktal *HIV* genoma plave je boje, dok je *Methanocaldococcus J.* narančast. Fraktali su poredani prema silaznim vrijednostima funkcije dobrote. Nakon 30 ponovljenih iteracija najbolja jedinka, barem s obzirom na dobrotu, je dobivena u 1. iteraciji. Iz slike 5.2 se vidi kako algoritam ne uspijeva uvijek razdvojiti skupove. Štoviše, uspoređujući vrijednosti dobrote s dobivenim prikazom, uočavamo kako visoka vrijednost dobrote ne garantira željenu vizualnu reprezentaciju podataka. Npr. fraktali dobiveni 23. iteracijom imaju daleko većudobrote (105.35) od fraktala dobivenih 9. iteracijom (20.5), ali su vizualno potonji udaljeniji.



Slika 5.2 fraktali dobiveni koristeći euklidsku udaljenost

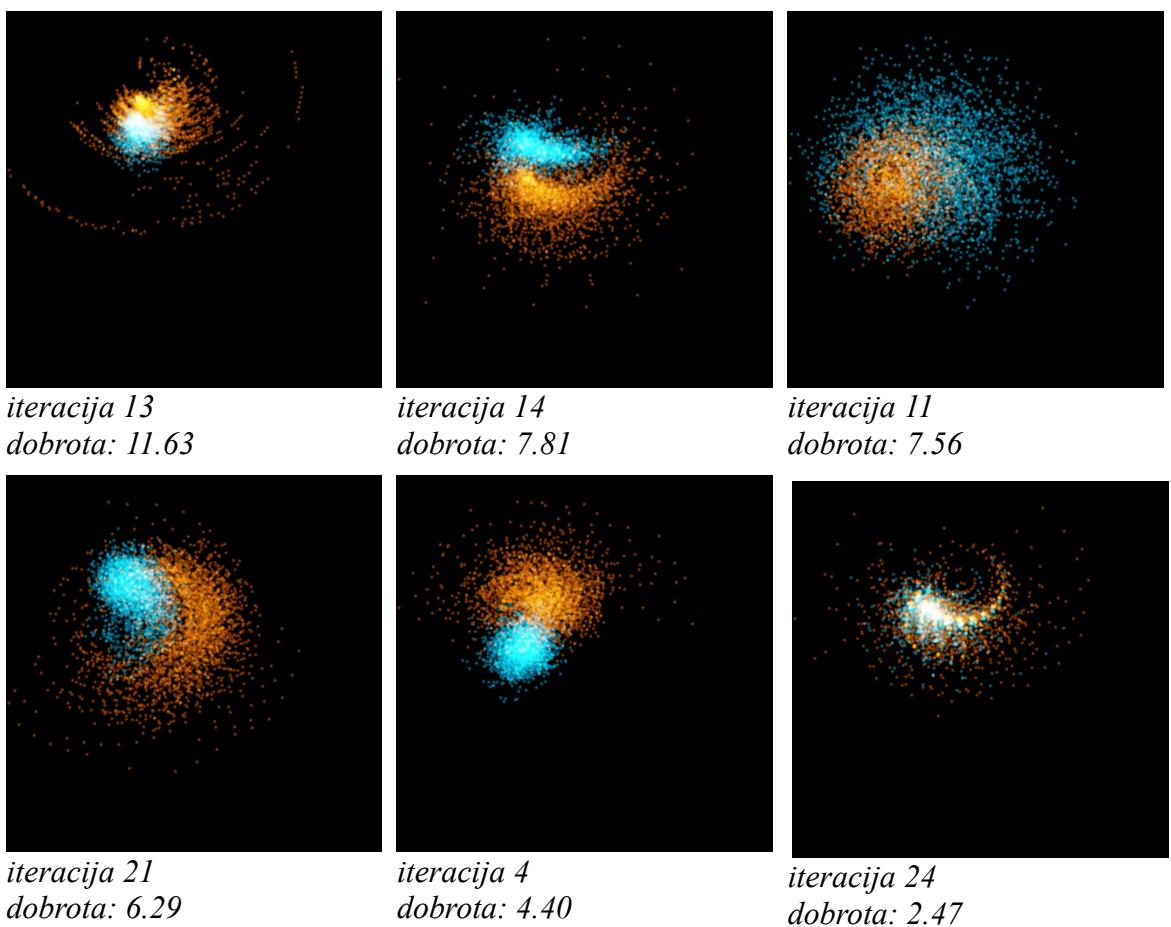
Rezultati na slici 5.3 dobiveni su korištenjem *Manhattan* udaljenosti kao funkcije dobrote. Ova varijanta također ima problem razdvajanja fraktala i vjerodostojnosti dobrote. Zanimljivo je primijetiti kako je dobrota nabolje jedinke (24.07) dobivena uporabom *Manhattan udaljenosti* daleko manja od vrijednosti dobrote najbolje jedinke (327.98) koju smo dobili kada smo koristili euklidsku udaljenost.





*Slika 5.3 fraktali dobiveni koristeći Manhattan udaljenost*

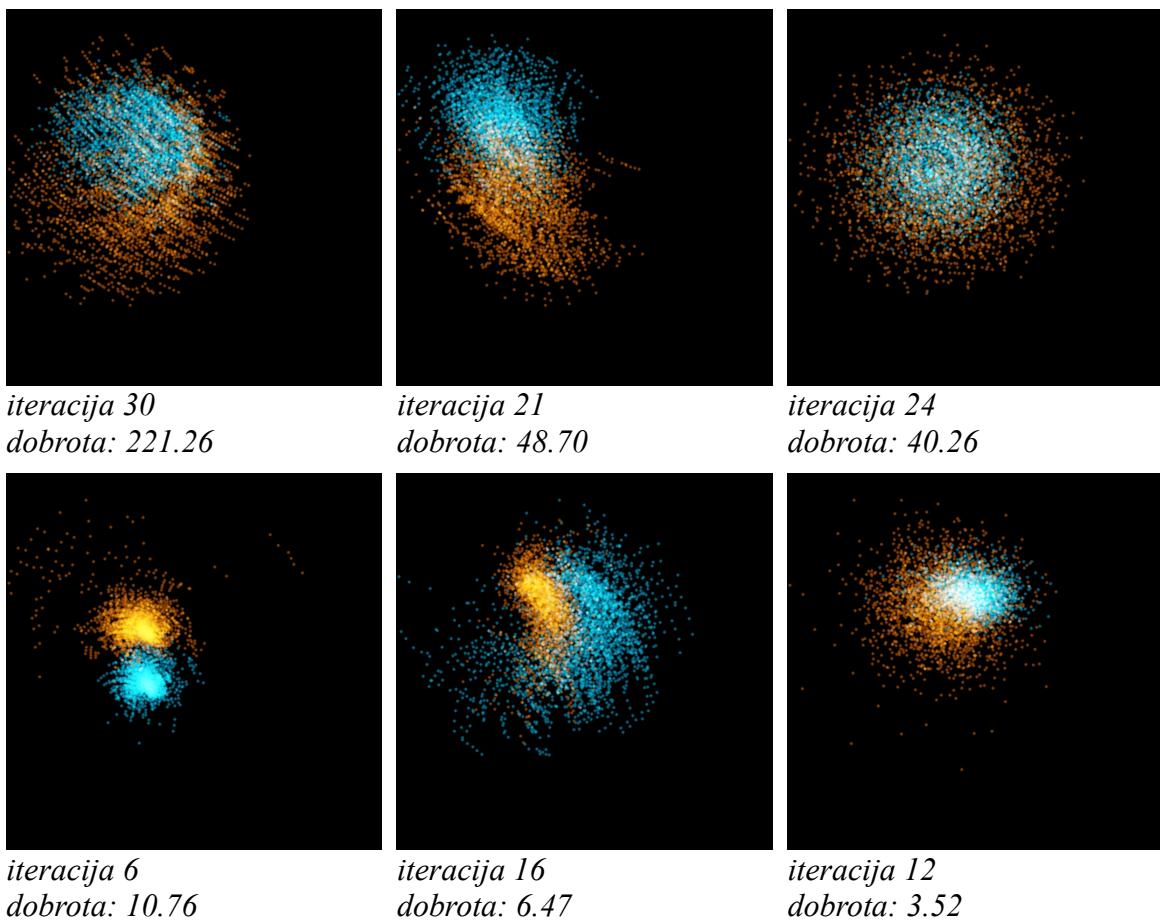
Rezultati na slici 5.4 dobiveni su korištenjem linearne interpolacije za  $t=4$ . I ova skupina fraktala prikazuje iste probleme kao i prethodne dvije.



*Slika 5.4 fraktali dobiveni linearnom interpolacijom*

Opisani postupak vizualizacije ponovljen je za ulazne DNA nizove genoma *Helicobacter Pylori* 2017 i *Human Papillomavirus tip 41* duljine 10000 generiranih Markovljevim modelom 6. reda. Na slici 5.5 prikazani su rezultati dobiveni korištenjem euklidske udaljenosti za računanje funkcije dobrote.

Lako se mogu uočiti isti problemi kao i kod prethodno prikazanih vizualizacija.



Slika 5.5 fraktali genoma *Helicobacter Pylori* 2017 i *Human Papillomavirus tip 41*  
dobiveni koristeći euklidsku udaljenost

## 6. Zaključak

U ovom radu opisan je postupak generiranja IFS fraktala uz pomoć DNA nizova, ali taj se postupak lako generalizira za bilo koji skup diskretnih podataka. Cilj je pronaći takve fraktale koji bi dva različita DNA niza vizualno razdvojili. Ta pretraga se obavlja pomoću evolucijskih algoritama. Implementirane su i ispitane tri verzije funkcije dobrote. Pokazalo se problematičnim vizualno odvojiti dva skupa točaka koje predstavljaju ulazne nizove. Moguće je da je to dijelom zbog sličnosti ulaznih podataka, ali i zbog samoslične prirode IFS fraktala.

Postoje dvije mogućnosti za daljnji nastavak, odnosno nadogradnju ove ideje. Prva je pronalazak učinkovitije funkcije dobrote, dok je druga razvijanje postupka koji bi iz postojećih fraktalnih reprezentacija mogli iščitati korisne informacije.

## Literatura

1. N. M. Luscombe, D. Greenbaum, M. Gerstein (2001) *What is Bioinformatics? A Proposed Definition and Overview of the Field.*  
[http://www.ebi.ac.uk/luscombe/docs/imia\\_review.pdf](http://www.ebi.ac.uk/luscombe/docs/imia_review.pdf)
2. F. Crick (1958), *Central Dogma of Molecular Biology*  
<http://cs.brynmawr.edu/Courses/cs380/fall2012/CrickCentralDogma1970.pdf>
3. Achuthsankar S. Nair, Vrinda V. Nair, Arun K. S., *Bio-sequence Signatures Using Chaos Game Representation*  
[http://deity.gov.in/hindi/sites/upload\\_files/dithindi/files/Bio-sewuence\\_AlpanaDey.pdf](http://deity.gov.in/hindi/sites/upload_files/dithindi/files/Bio-sewuence_AlpanaDey.pdf)
4. D. Ashlock (2003) *Application to Bioinformatics: Chapter 15*  
<https://orion.math.iastate.edu/danwell/ma378/chapter15.pdf>
5. D. Ashlock, J. Golden, *Evolutionary Computation and Fractal Visualization of Sequence Data*  
<http://eldar.mathstat.uoguelph.ca/dashlock/eprints/biochapter.pdf>
6. H.Joel Jeffrey, *Chaos game representation of gene structure*  
<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC330698/pdf/nar00192-0217.pdf>
7. M. Domazet (2013), Bioinformatika [skripta]
8. M. Čupić, Ž. Mihajlović (2014), Interaktivna računalna grafika kroz primjere u OpenGL-u
9. Z. Vondraček (2008), Markovljevi lanci [predavanja]
10. M. Golub G (2004), Genetski algoritam [skripta]
11. J. Batterson (2013), *The order in Chaos*  
<http://www.whitman.edu/mathematics/SeniorProjectArchive/2013/Batterson.pdf>

## **Fraktalna vizualizacija evolucijskim algoritmima**

### **Sažetak**

Ovaj rad opisuje postupak vizualizacije DNA podataka pomoću fraktala. Pretražuje se prostor rješenja za optimalnim oblikom fraktala koji bi prenio što je više moguće korisnih informacija o ulaznom nizu. Pretraga se radi pomoću evolucijskih algoritama. Ulagni podaci su dva DNA niza, a rezultat je njihova fraktalna vizualna reprezentacija. Opisuju se i korišteni evolucijski operatori te se uspoređuju rezultati dobiveni izborom različitih funkcija dobrote.

**Ključne riječi:** vizualizacija, fraktali, evolucijski algoritmi, DNA podaci

## **Fractal visualization with evolutionary algorithms**

### **Abstract**

This thesis describes a method of visualizing DNA sequences with fractals. The solution space is searched for the optimal fractal shape which would convey as much useful information as possible. The search is done using evolutionary algorithms. The input data are two DNA sequences, and the result is their fractal visual representation. The thesis also describes used evolutionary operators and analyzes the results obtained using different fitness functions.

**Keywords:** visualization, fractals, evolutionary algorithms, DNA sequence