Pretraga katalitičkih trijada u enzimima

Miličević, Lana

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski sveučilišni studij računarstva

Završni rad

Pretraga katalitičkih trijada u enzimima

Rijeka, rujan 2022.

Lana Miličević
0069087943

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski sveučilišni studij računarstva

Završni rad

Pretraga katalitičkih trijada u enzimima

Mentor: doc. dr. sc. Goran Mauša

Rijeka, rujan 2022.

Lana Miličević
0069087943

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Uvod u objektno orijentirano programiranje**
Grana: **2.09.04 umjetna inteligencija**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Lana Miličević (0069087943)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Pretraga katalitičkih trijada u enzimima / Search of catalytic triads in enzymes**

Opis zadatka:

Definirati pojam katalitičke trijade u prirodnim enzimima i njegove sastavne dijelove. Izraditi program za pretragu katalitičkih trijada iz zapisa u formatu pdb i za izračun trigonometrijskih parametara koji ih karakteriziraju poput kuteva i duljina stranica trokuta. Predložiti rješenje za pronalazak karakterističnih obrazaca katalitičkih trijada u skupu odabranih enzima.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 14. ožujka 2022.

Mentor:

Doc. Goran Mauša, dipl. ing.

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, rujan 2022.

_____
Lana Miličević

Zahvala

Zahvaljujem laboratoriju DeShPet na pruženoj prilici i stručnoj potpori. Među njima posebno bih izdvojila svog mentora, doc. dr. sc. Gorana Maušu, koji me za vrijeme čitava rada usmjeravao i podržavao.
Na podršci tijekom studiranja zahvaljujem svojoj obitelji i prijateljima.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

Ovaj je završni rad dio uspostavnog istraživačkog projekta Hrvatske zaklade za znanost pod naslovom „Dizajn katalitički aktivnih peptida i peptidnih nanostruktura" pod oznakom UIP-2019-04-7999. Osnovni cilj ovoga rada bilo je stvoriti programski sustav za pronalaženje katalitičkih trijadi iz prostornog zapisa struktura molekula u formatu `pdb`, a kojim će se omogućiti pronalazak karakterističnih strukturnih obrazaca unutar pojedinih enzima. Također, rad pruža prijedloge otkrivanja karakterističnih obrazaca katalitičkih trijadi u skupu odabranih enzima kroz implementaciju genetskog algoritma pronalaženje među njima najčešćeg te enzimima zajedničkog uzorka, teorijska razmatranja i analizu dobivenih rezultata te smjernice za daljnji rad na projektu.

1.1 Opis problema

Rad će predstaviti i analizirati pretragu katalitičkih trijadi enzima zapisanih u `pdb` datotekama. Polazi od objašnjenja što su navedene trijade, koja je njihova važnost te koja svojstva moraju zadovoljavati. Iz toga proizlaze četiri različite perspektive pristupa pojmu katalitičkih trijada, to jest u programskom smislu - četiri metode njihove pretrage.

Nadalje, rad daje uvid u programsko razmatranje karakteristika navedenih tri-

jada koje daju optimalne rezultate u katalitičkim reakcijama, a što je realizirano korištenjem genetskog algoritma. Predstavlja i objašnjava tehnologije, postupke i algoritme koji su doveli do svih u radu donesenih zaključaka, kao i spoznaje i pitanja s područja biotehnologije koja su temelj za njegovo provođenje. Naposljetku, bit će predstavljeni temeljni zaključci doneseni u radu i prijedlozi za daljnje istraživanje ove teme iz perspektive računarstva.

1.2 Motivacija

Katalitičkim trijadama smatraju se trijade kiselinsko-bazno-nukleofilnih ostataka. Njihov značaj leži u ulozi koju imaju za izvođenje reakcije hidrolize estera, omogućenoj prostornom raspoređenošću aktivnih atoma [1].
Budući da je prostorni raspored atoma ključ u pronalaženju, definiranju i funkcioniranju katalitičke trijade, a može se promatrati preko jasno definiranih numeričkih podataka, nameće se ideja programskog pretraživanja zapisa enzima. Implementirane su četiri metode pronalaženja katalitičkih trijadi detaljnije opisane u Poglavlju 3, a koje se razlikuju po tome što u svojem polazištu definiraju kao katalitičku trijadu. Nad rezultatima pretrage spomenutih metoda u ovom se i daljnjem radu nastoji uočiti karakteristične obrasce, također na načine detaljnije opisane u nastavku.

Pronalazak karakterističnih prostornih obrazaca podrazumijeva neograničeno područje pretraživanja (eng. *search space*), zbog čega je odabran genetski algoritam kao metoda usmjerenog pretraživanja. Glavni je cilj genetskih algoritama pronalazak optimalnog rješenja za problem čije je pordučje pretraživanja preveliko za iscrpnu pretragu ili za problem kod kojeg nedostaje znanje iz njegove domene. U kontekstu ovog rada, riječ je o trijadama koje najbolje zadovoljavaju neko od zadanih mjerila kvalitete.

Izučavanje biologije proteina počiva na ispitivanju i razumijevanju uzročno-posljedične povezanosti strukturnih elemenata i promatranih funkcija [2], [3]. Kad

je riječ o enzimima s katalitičkim trijadama, takav pristup u praktičnom smislu ima dvojak učinak: (i) povezivanje uočenih funkcija enzima s konkretnim katalitičkim trijadama i aktivnim mjestima te (ii) donošenje generalnih zaključaka o samim trijadama uočavanjem uzoraka i pravila u dobivenim rezultatima.

Valja istaknuti kako su oba područja na kojima počiva predmet istraživanja ovog rada - računarstvo i biotehnologija - relativno nova, kao i sama tema proučavanja. Trenutni je trend u odabiru metoda za rješavanje sličnih problema još uvijek „ručni" pristup temeljen na postojećim spoznajama o strukturama proteina [2], [4]. Samim time, moglo bi se reći kako je razina interdisciplinarnosti u vrijeme pisanja rada u aktualnim znanstvenoistraživačkim radovima nedovoljno istražena te se ovim projektom želi promovirati mogućnosti suvremenih programskih pristupa u rješavanju problema kemije enzima. Iz tog je razloga u provedbi ovog rada temeljna motivacija bila prevazići navedenu barijeru i problem usko vezan uz biotehnologiju riješiti programski, očekujući pritom veću fleksibilnost u radu i pronalaženje rezultata koji klasičnim pristupom potencijalno ne bi bili ostvareni.

1.3 Programska podrška

Sav programski kod implementiran je u programskom jeziku Python verzije 3.8. Za obradu `pdb` datoteka enzima od velike je koristi bila Biopython knjižnica. Biopython u radu s `pdb` datotekama prepoznaje značajke strukture s kojom radi (u više detalja nego što je potrebno u ovom radu), a nakon pojednostavljivanja neke se takve informacije gube. Ipak, ocijenjeno je da su jedine potrebne informacije one koje se tiču vrhova trijadi jer su temelji algoritma isključivo prostorno-geometrijski. Za implementaciju genetskih algoritama korišten je mikro razvojni okvir NiaPy, čije se mogućnosti i korištenje nad u radu zadanim problemima detaljno analiziraju u nastavku rada.

Poglavlje 2

Katalitičke trijade

2.1 Definicija

Hidrolaze su u prirodi sveprisutne biomolekule koje koristeći vodu razbijaju kemijske veze te su zaslužne za degradativne procese u ljudskom tijelu. Aktivno mjesto hidrolaze, direktno odgovorno za njenu katalitičku funkcionalnost, često sadrži katalitičku trijadu [5].

Katalitičkim se trijadama smatraju skupovi *nukleofil-baza-kiselina* (*Nuc-Acid-Base)* na čijem međudjelovanju počivaju katalitička svojstva enzima koji ih sadrže [6]. Točnije, takav skup čine atomi iz aminokiselina navedenih tipova, a koji zadovoljavaju određena svojstva u vidu tipova atoma, aminokiselina u kojima se oni nalaze te prostornog rasporeda. Najčešće su to aminokiseline serin (nukleofil), histidin (baza) i asparaginska kiselina. U ovom se radu, ovisno o metodi pretrage, katalitičkom trijadom smatra više različitih kombinacija atoma, detaljnije opisanih u Poglavlju 3.

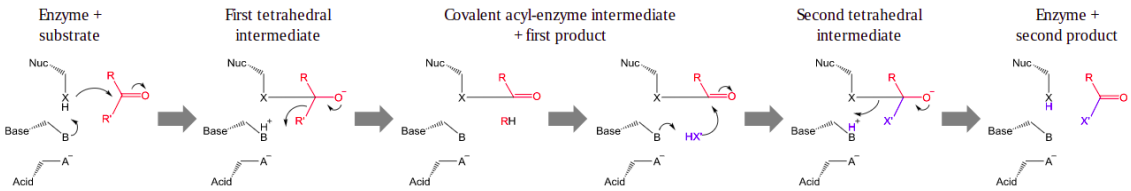
2.2 Uloga i značaj

Kataliza (ubrzavanje) kemijskih reakcija jedan je od temeljnih koncepata čitavog područja biokemije, a rad na povećanju enzimatske stope akceleracije temelji se upravo na katalitičkim trijadama. Pritom, unatoč vrlo razvijenim i kompleksnim spoznajama o evoluciji enzima, zbog njezine ponekad nasumične i nepredvidive prirode neka temeljna pitanja ostaju neodgovorena. Put ka rješenju tih pitanja leži u primjeni i eksperimentalnog i računskog pristupa ispitivanju svojstava katalitičkih struktura [3].

Iako slični u sekvenci i trodimenzionalnoj strukturi, enzimi koji posjeduju katalitičke trijade zaslužni su za katalizu mnogobrojnih i međusobno diverznih reakcijskih mehanizama [7].

2.3 Načela funkcioniranja

Katalitičke su trijade samo jedan od mnogih mehanizama enzimatske katalize. Ipak, svi navedeni mehanizmi idejno se temelje na istom principu - smanjivanju energije aktivacije potrebne da se neka reakcija provede [8]. Sniženju energije aktivacije u pozadini leži više mogućih faktora, općenito vezanih uz naboje atoma i veze među njima. Ukratko, katalitičke trijade (*nukleofil-baza-kiselina*) služe generiranju nukleofilnih ostataka za katalizu. Ponašaju se kao svojevrstan relej naboja koji polarizira nukleofil i započinje njegovu aktivnost. Uloga baze se pritom u najvećoj mjeri odnosi na polariziranje i orijentiranje nukleofila, a uloga kiseline na njegovu stabilizaciju [9].



Slika 2.1 Reakcijski mehanizam katalitičke trijade

Izvor: Evolvability of a viral protease: experimental evolution of catalysis, robustness and specificity [9]

2.4 Pretraga katalitičkih trijadi

Pretraga katalitičkih trijadi podrazumijeva pripremu i obradu podataka o enzimima zapisanim u `pdb` datoteke te zapis dobivenih rezultata u `csv` format. Sve navedeno realizirano je programom pisanom u programskom jeziku Python te uvelike počiva na Biopython knjižnici.

Skup podataka na kojem počiva ovaj rad sastoji se od 22 `pdb` datoteke u kojima su zapisani odabrani enzimi, odabrani da budu hidrolaze (skupina EC 3.1.) koje posjeduju katalitičku trijadu. Skup podataka preuzet je sa sustava Protein Data Bank. S ciljem optimizacije programski je obrađen i pročišćen: odbačeni su atomi koji nikako ne mogu biti kandidat za trijadu, a oni koji mogu podijeljeni su u zasebne `nuc`, `acid` i `base` datoteke.

2.5 Cilj

Cilj pretrage katalitičkih trijada u enzimima (prema bilo kojoj od metoda iz poglavlja 3) jest identificirati:

1. **prihvaćene katalitičke trijade** (one koje zadovoljavaju uvjete nametnute nekom od metoda)
2. (opcionalno) trijade koje su po geometriji trokuti slični prihvaćenim trijadama i sadrže iste atome, ali ne zadovoljavaju neki od geometrijskih uvjeta

da bi ih se smatralo trijadom (u daljnjem tekstu: „**slični trokuti**")

Uz prihvaćene trijade, koje su glavna motivacija provođenja rada, važna je i informacija o sličnim trokutima jer se njome može dobiti uvid u to postoje li statistički značajne sličnosti (ili razlike) njih i prihvaćenih trijadi.

Već iz neobrađenih podataka o identificiranim trijadama i sličnim trokutima mogu se izvesti zanimljivi zaključci o enzimima i broju trijada koje sadrže, omjerima broja prihvaćenih trijada i sličnih trokuta te provesti usporedbe s rezultatima referentnog istraživanja [5]. Ipak, te su informacije van domene onoga što ovaj rad proučava te su dane na raspolaganje suradnicima za daljnje istraživanje.

2.6 Algoritam pretrage

Neovisno o metodi pretrage, pretraga trijada uvijek prati jednostavnu ideju: za sve pročišćene `pdb` datoteke enzima pronaći trijade te saznanja formatirati i pohraniti u obliku `csv` datoteke za daljnju obradu.

Pritom se razlikuju dva pristupa pretrazi: onaj kad se slični trokuti uzimaju u obzir, i onaj kad ne. Budući da se sličnost trokuta promatra preko odnosa omjera, a u ovom ju slučaju ima smisla provesti provjerom svih triju kutova, za kandidata za trijadu potrebno je ispitati sve moguće *Nuc*, *Acid* i *Base* kombinacije. Danoj se kombinaciji stoga nužno prvo provjeravaju rasponi kutova (kako bi se utvrdilo može li biti i od kakvog interesa), a zatim rasponi udaljenosti atoma (kako bi se utvrdilo je li riječ o katalitičkoj trijadi ili sličnom trokutu). U programskom smislu govorimo o složenosti $O(n^m)$, gdje je n najveći broj atoma kandidata za neki vrh, a m broj atoma koji se provjeravaju (ovdje 3 ili 4, ovisno o broju atoma koje dana metoda promatra).

S druge strane, u pristupu koji zanemaruje slične trokute, ispituju se rasponi udaljenosti (a oni, samim time, podrazumijevaju i kuteve). Radi optimizacije, za razliku od prethodno opisanog, ovaj pristup nije proveden tako da se nad svim mogućim kombinacijama vrše provjere, već inkrementalno. To znači da se trijada

slaže „vrh po vrh”, odnosno odbacuje čim vrh nadodan na listu vrhova kandidata ne zadovoljava zadani uvjet. Ukoliko ni u kom trenutku ne bude odbačena, sve su udaljenosti zadovoljile uvjet te biva prihvaćena kao katalitička trijada. Složenost je $O(n)$, gdje je n najveći broj atoma kandidata za neki vrh.

Poglavlje 3

Metode pretrage

Rad obuhvaća četiri različite metode pronalaženja katalitičkih trijada, koje se razlikuju po tome koje atome uzimaju u obzir za *Nuc-Acid-Base* trojku. Također, razlikuju se po dopuštenoj geometriji, tj. rasponima dužina stranica i veličina kutova trokuta koji opisuju prihvaćenu trijadu. Iz perspektive biotehnologije, svaka od predloženih metoda daje informaciju o tome je li za katalitičku reakciju tog enzima bitniji atom ili aminokiselina, govori o važnosti geometrije te trijade u usporedbi s drugim kemijskim katalitičkim faktorima te daje podatke za daljnje istraživanje. Implementacija predloženih i u nastavku opisanih metoda može ubrzati proces otkrivanja trijadnih mehanizama, koji je inače dugotrajan i skup.

3.1 Originalna metoda

Originalna metoda (eng. *old method*) metoda je čija provedba odgovara pretpostavkama postavljenim u inicijalnim fazama ovog rada. Točnije, da bi se skup atoma smatrao katalitičkom trijadom, mora zadovoljiti kriterije koje pretpostavlja istraživanje [5], čiji je ovaj rad nastavak, a koji su u svojoj definiciji stroži od onih koje pretpostavljaju ostale metode. Samim time, za očekivati je manji broj trokuta koji su prihvaćeni kao katalitičke trijade u odnosu na druge metode.

Vrh	**Aminokiselina**	**Atom**
Nuc	Ser	OG
Nuc	Cys	SG
Acid	Trp/Asp/Glu	OD1
Acid	Trp/Asp/Glu	OD2
Base	Asp/Glu/His	CG

Tablica 3.1 Atomi (vrhovi) koje pretražuje originalna metoda

Originalna metoda za svaki enzim pronalazi sve potencijalne *Nuc-Acid-Base* trojke. Na svakoj od njih vrši se provjera kutova i stranica, koje će odrediti klasifikaciju dane trojke. Ako uvjet eksperimentalno utvrđenih raspona kutova nije zadovoljen, trojka se odbacuje i prelazi se na sljedećeg kandidata.
Ako je pak uvjet zadovoljen, provjerava se je li trojka katalitička trijada ili sličan trokut. To se vrši provjerom zadovoljava li geometrija trojke eksperimentalno određene raspone dužina stranica. Zadovoljava li, smatra se katalitičkom trijadom; u suprotnom, sličnim trokutom.

Dozvoljeni se rasponi pritom izražavaju preko sljedeće zakonitosti:

$$\text{raspon} = \text{prosječna vrijednost} \pm \text{STDEVx3} \tag{3.1}$$

te koriste vrijednosti zapisane u Tablicama 3.2 i 3.3.

Stranica	**Prosječna dužina [Å]**	**STDEVx3**
Nuc-Acid	7.15	1.42
Acid-Base	3.64	0.98
Base-Nuc	4.86	1.06

Tablica 3.2 Raspon dozvoljenih udaljenosti u katalitičkoj trijadi prema originalnoj metodi

Kut	**Prosječna veličina [°]**	**STDEVx3**
Nuc	26.84	15.38
Acid	38.01	27.84
Base	115.14	41.27

Tablica 3.3 Raspon dozvoljenih kutova u katalitičkoj trijadi prema originalnoj metodi

3.2 Metoda zasnovana na atomima

Motivacija iza metode zasnovane na atomima (eng. *atom-based method*) leži u pitanju može li se katalitička funkcija kemijske reakcije ekstrapolirati na geometriju triju atoma umjesto na cijele aminokiseline.
Glavna joj je značajka da u obzir uzima samo vrstu atoma, ne i aminokiselinu u kojoj se on nalazi. Dakle, atom mora zadovoljavati samo vrstu, tj. ne postoji kriterij aminokiseline. Time se kriteriji za kandidate za trijade znatno proširuju. Posljedično, za očekivati je i veći broj pronađenih trijadi u svim enzimima.

Vrh	**Atom**	**Naziv atoma**
Acid	bilo koji O	Acid O
Base N1	bilo koji N	Base N1
Base N2	bilo koji N	Base N2
Nuc	bilo koji O	Nuc O
Nuc	bilo koji S	Nuc S

Tablica 3.4 Atomi (vrhovi) koje pretražuje metoda zasnovana na atomima

U implementaciji ove metode nije se pridavala pozornost pronalaženju sličnih trokuta. Iz tog razloga bilo je moguće pretragu izvoditi na inkrementalan način opisan u Potpoglavlju 2.6, što složenost sa $O(n^m) = O(n^4)$ snižava na $O(n)$ te uvelike ubrzava izvođenje programa.

Stranica	**Maksimalna dužina [Å]**
Nuc-Acid	9
Acid-Base N1	4
Base N1-Base N2	2.5
Base N2-Nuc	4.5

Tablica 3.5 Maksimalne dozvoljene udaljenosti u katalitičkoj trijadi prema metodi zasnovanoj na atomima

Kut	**Maksimalna veličina [°]**
Nuc	33
Acid	50
Base N1	180
Base N2	180

Tablica 3.6 Maksimalni dozvoljeni kutova u katalitičkoj trijadi prema metodi zasnovanoj na atomima

3.3 Metoda zasnovana na aminokiselinama

Posebnost metode zasnovane na aminokiselinama (eng. *residue-based method*) leži u tome što vrhovima trijade promatra samo aminokiselinu u kojoj se nalaze, ne i vrstu atoma - dakle, na neki način čini protutežu metodi zasnovanoj na atomima. Po implementaciji je najsličnija originalnoj metodi, jer iziskuje samo pronalazak trokuta čije vrhove čini *Nuc-Acid-Base* trojke te provjeru geometrije tog trokuta.

Vrh	**Aminokiselina**	**Atom**
Acid	Asp	CB
Acid	Glu	CB
Base	His	CB
Nuc	Cys	CB
Nuc	Ser	CB

Tablica 3.7 Atomi (vrhovi) koje pretražuje metoda zasnovana na aminokiselinama

Budući da je po svemu osim uvjeta za vrstu atoma ista kao i originalna metoda, u metodi zasnovanoj na aminokiselinama se kao i u originalnoj metodi traže i katalitičke trijade i slični trokuti. Složenost izvođenja pritom iznosi $O(n^m) = O(n^3)$.

Po pronalasku trokuta prvo se vrši provjera njegovih kutova, a zatim i stranica. Ako je zadovoljio propisane uvjete kuteva, pronađeni se trokut podliježe provjeri duljina stranica. Ovisno o ishodu provjere stranica, svrstava ga se ili u listu prihvaćenih trijadi, ili u listu sličnih trokuta.

Stranica	**Maksimalna dužina [Å]**
Nuc-Acid	11
Acid-Base	8
Base-Nuc	8

Tablica 3.8 Maksimalne dozvoljene udaljenosti u katalitičkoj trijadi prema metodi zasnovanoj na aminokiselinama

Kut	**Maksimalna veličina [°]**
Nuc	40
Acid	55
Base	130

Tablica 3.9 Maksimalni dozvoljeni kutova u katalitičkoj trijadi prema metodi zasnovanoj na aminokiselinama

3.4 Hibridna metoda

Hibridna metoda (eng. *hybrid method*) objedinjuje ideje metode zasnovane na atomima (Potpoglavlje 3.2) i one zasnovane na aminokiselinama (Potpoglavlje 3.3). U implementaciji po svemu nalikuje metodi zasnovanoj na atomima, uz dodatne kriterije koji atomi određenog elementa mogu biti dijelom trijade.

Vrh	**Atom**
Acid	OD1
Acid	OD2
Acid	O
Acid	OE1
Acid	OE2
Base	ND1
Base	NE2
Nuc	OG
Nuc	SG

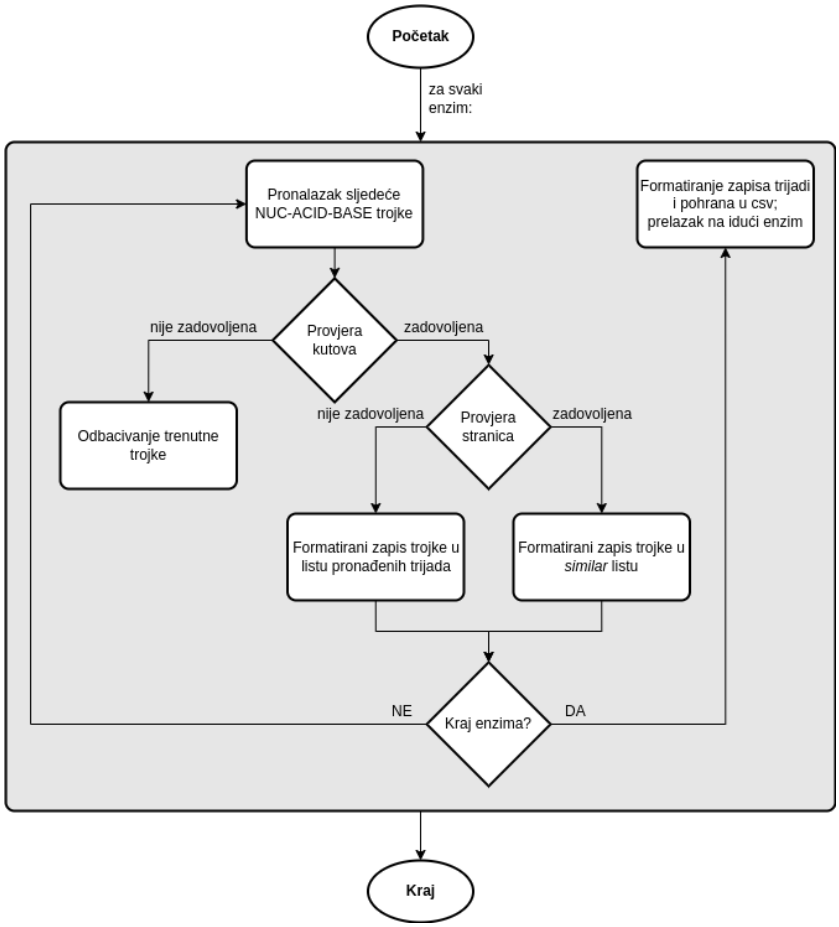
Tablica 3.10 Atomi (vrhovi) koje pretražuje hibridna metoda

Stranica	**Maksimalna dužina [Å]**
Nuc-Acid	9
Acid-ND1	4
ND1-NE2	2.5
NE2-Nuc	4.5

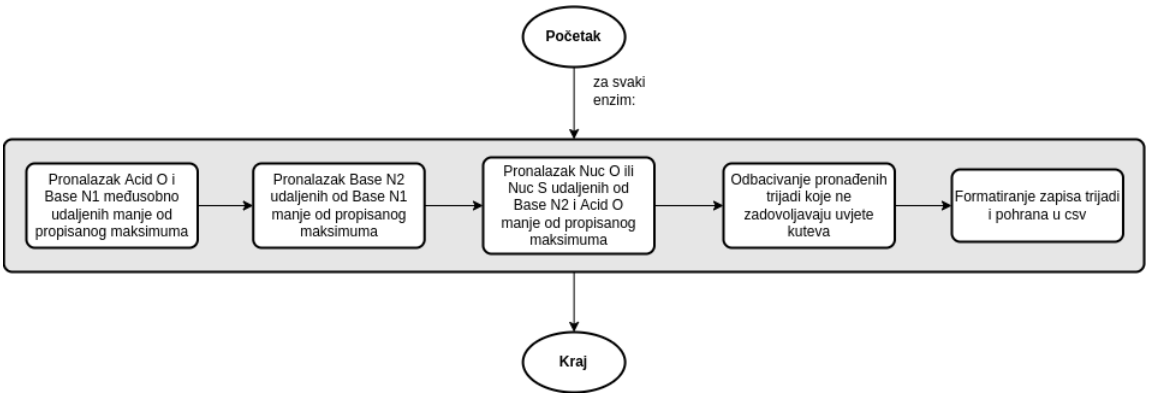
Tablica 3.11 Maksimalne dozvoljene udaljenosti u katalitičkoj trijadi prema hibridnoj metodi

Kut	**Maksimalna veličina [°]**
Nuc	33
Acid	50
ND1	180
NE2	180

Tablica 3.12 Maksimalni dozvoljeni kutova u katalitičkoj trijadi prema hibridnoj metodi



Slika 3.1 Algoritam pretrage trijadi kod metoda koje traže slične trokute



Slika 3.2 Algoritam pretrage trijadi kod metoda koje ne traže slične trokute

Poglavlje 4

Genetski algoritmi

4.1 Karakteristike

Mjestom i vremenom nastanka genetskih algoritama smatra se Sveučilište u Michiganu 60-ih godina prošloga stoljeća, a predvodnicima istraživanja prof. John Holland i njegov tim suradnika. Cilj njihova istraživanja bio je detaljno objasniti i apstrahirati procese prilagodbe prirodnih sustava te zatim po njima modelirati softver [10]. Stoga se o genetskim algoritmima govori kao jednoj od vrsta algoritama inspiriranih prirodom koji se modeliraju po uzoru na stvarne prirodne pojave, a najčešće s ciljem optimizacije.

Da bi se jasno predstavile temeljne karakteristike genetskih algoritama, potrebno je krenuti od motivacije za razvoj tih algoritama. To jest, valja naglasiti ključne razlike u odnosu na tradicionalne metode optimizacije, a to su:

- rad nad skupovima parametara, a ne individualnim parametrima;
- pretraga/kretanje nad skupinom točaka, ne pomicanje iz jedne točke;
- korištenje informacija o isplativosti dobivenih primjenom objektivnih funkcija;
- korištenje stohastičkih načela tranzicije umjesto deterministčkih [10].

Genetski algoritmi spadaju u skupinu metaheurističkih algoritama. To znači da iterativnim metodama iskorištavanja prostora pretrage podređenu heuristiku transformira tako da u svakoj iteraciji ona vodi ka konačnom rješenju bližem onom teoretski optimalnom [11]. Najčešće se primjenjuju u svrhu optimizacije jer su efikasni, posjeduju koncepte ugađanja i rukovanja parametrima te primjenjuju metode sprečavanja stagnacije [12].

Kad se govori o genetskim algoritmima, riječ je o skupini algoritama čije implementacije počivaju na upravo opisanom načelu, ali se razlikuju u implementacijskim detaljima vezanim uz odabir karakteristika strukture algoritma opisanih u Potpoglavlju 4.2. Pojam „genetski algoritam" označava jednu specifičnu implementaciju te vrste algoritma.

Zahvaljujući moći koju pokazuju u potrazi za optimalnim rješenjima praćenom računskom jednostavnošću izvedbe, genetski algoritmi nalaze sve širu primjenu u poslovnim, znanstvenim i inženjerskim primjenama [13]. Neovisno o području i prirodi problema nad kojim se primjenjuju, česta je motivacija da se koriste upravo genetski algoritmi potreba za robusnim postupkom koji rješava raznorodne probleme [10].

4.2 Osnovni pojmovi

Inspiracija za konstrukciju genetskih algoritama leži u biološkoj evoluciji - vjerojatnije je da će idućim *generacijama* svoje *gene* prenijeti *jedinke* najveće dobrote (eng. *fitness*) [14]. Stoga, prije opisa algoritama, valja definirati sljedeće pojmove [14]–[17]:

Jedinka

Svaka jedinka jedno je od mogućih rješenja problema kojeg algoritam rješava - u ovom slučaju, to su prihvaćene trijade i slični trokuti. Ipak, nisu sve jedinke jednako uspješne i pretpostavlja se da će samo one koje su po dobroti bolja rješenja

„preživjeti".

Populacija

Populacija je set svih jedinki koje predstavljaju rješenje problema. Rješavanje počinje inicijalnom populacijom, koja može biti unaprijed zadana, nasumično odabrana iz šireg seta podataka, nasumično generirana i drugo. Populacija jedne iteracije naziva se **generacijom**.

Geni

Geni su parametri (varijable) koje karakteriziraju svaku jedinku. Identitet jedinke čine geni koji ju opisuju, a najčešće se bilježe kao stringovi ili polja numeričkih vrijednosti. Skup gena jedinke naziva se i **fenotip**.
Fenotip jedinke dobiva se enkodiranjem, procesom prevođenja informacija o jedinki u zadani oblik. Enkodiranjem se ograničava veličina skupa informacija o jedinki koje se za problem smatraju relevantnim.

Funkcija dobrote (*fitness* funkcija)

Za izračun ocjene sposobnosti jedinke služi funkcija dobrote, funkcija koja matematički opisuje sposobnost te jedinke u nadmetanju s ostatkom populacije. Pritom pruža na cijeloj domeni primjene (ili više njih) uniforman pristup procjeni jedinke. Vjerojatnost odabira jedinke za reprodukciju temelji se na ocjeni njene dobrote.

Odabir

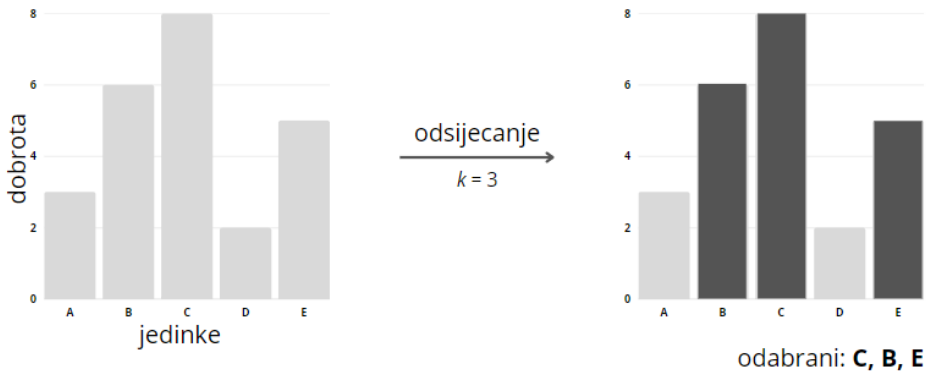
Odabir (eng. *selection*) podrazumijeva proces izdvajanja roditelja, tj. jedinki za reprodukciju koje će pomoći proizvesti iduću generaciju. Pritom razlikujemo više pristupa, a najčešći od njih su:

- **Odabir odsijecanjem** - roditelji su k jedinki iz populacije s najboljim vrijednostima dobrote.
- **Turnirski odabir** - svaki od roditelja pobjednik je turnira, tj. jedinka najbolje vrijednosti dobrote od k nasumično odabranih jedinki iz populacije.

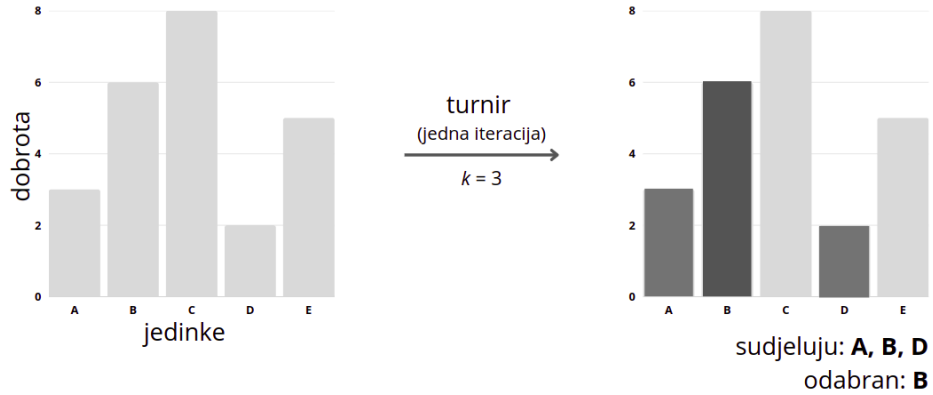
Goldberg i Deb pokazali su kako je turnirski odabir u usporedbi s ostalim metodama odabira efikasniji i manje sklon preranoj konvergenciji [18].

- **Odabir proporcionalan dobroti** (rulet odabir, eng. *roulette wheel*) - vjerojatnost odabira svakog od roditelja proporcionalna je njegovoj relativnoj dobroti (u odnosu na ostatak populacije).

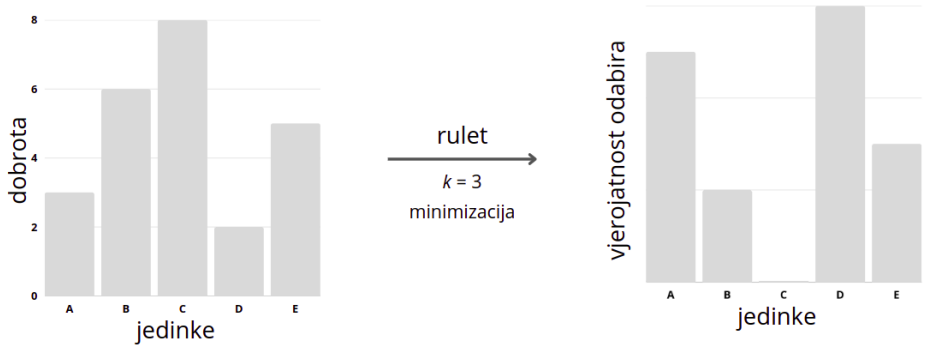
Neovisno o pristupu, valja primijetiti kako je glavna ideja odabira uvijek ista: među odabranim jedinkama, dati prednost onima s boljom vrijednosti dobrote. [17].



Slika 4.1 Odabir odsijecanjem
Odabir $k = 3$ najboljih jedinki iz populacije veličine $m = 5$ na problemu maksimizacije.



Slika 4.2 Turnirski odabir
Jedna iteracija turnira veličine $k = 3$ u populacije veličine $m = 5$ na problemu maksimizacije.



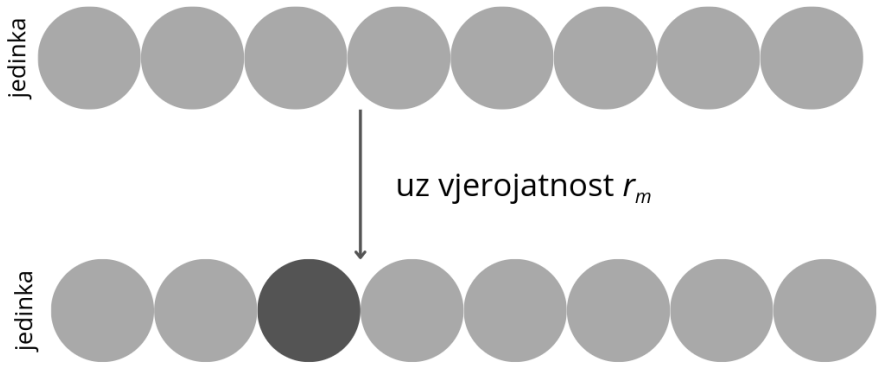
Slika 4.3 Odabir proporcionalan dobroti (rulet odabir)
Vjerojatnosti odabira svake od jedinki iz populacije veličine $m = 5$ na problemu minimizacije.

Pri odabiru mogu se koristiti **heuristike** koje rješavanje problema čine bržim i robusnijim. Primjer je heuristika specijacije, koja se implementira kako bi se spriječilo križanje međusobno presličnih jedinki i time spriječila prerana konvergencija ka suboptimalnom rješenju.

Mutacija

Mutacija je promjena vrijednosti nekog gena ili skupa gena jedinke. Ta promjena mora biti dozvoljena, tj. nova vrijednost mora biti unutar propisanog raspona ili skupa vrijednosti izmijenjenog gena. Obično se implementira mala i nasumična vjerojatnost da će doći do mutacije. Razlog tomu je održavanje diverziteta populacije i sprečavanje preuranjene konvergencije rezultata ka suboptimalnom rješenju.

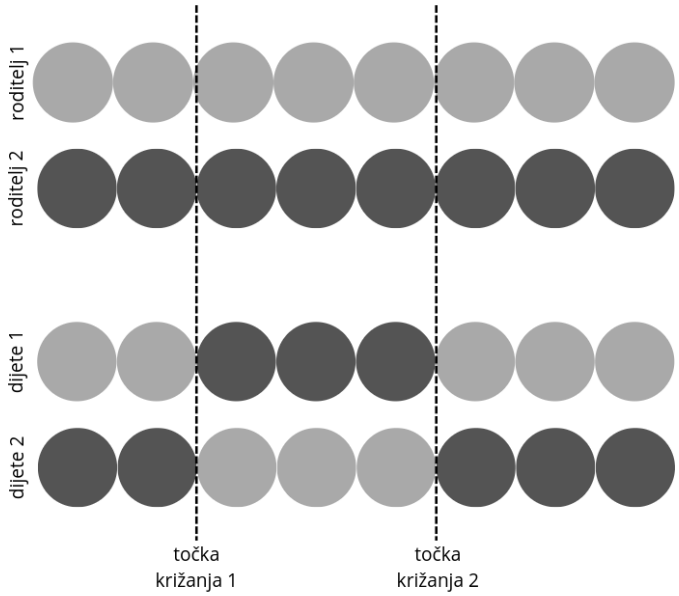
U istraživanju genetskih algoritama mutacija se načelno smatra sekundarnom operacijom, a križanje primarnom, što je razlog malim vrijednostima mutacije, odnosno većima za stope križanja. S druge strane, postoje pristupi evolucijskog računarstva pak uzimaju kao primarni faktor razvoja populacije u optimizacijskim problemima [19].



Slika 4.4 Mutacija
Uz vjerojatnost jednaku stopi mutacije r_m mijenja se vrijednost gena jedinke.

Križanje

Termin križanje (*crossover*) odnosi na kombiniranje roditeljskih kromosoma s ciljem formiranja djece. Postoje mnogi načini na koje se ta kombinacija može izvesti, a mogu se podijeliti u standardne, binarne i načine ovisne o primjeni [20].



Slika 4.5 Križanje
Nastanak potomstva križanjem na primjeru križanja na dvije točke.

Strategije zamjene jedinki

Kod svake generacije dolazi do odbacivanja dijela populacije kako bi njena veličina ostala fiksna. Pritom se razlikuju dva glavna moguća pristupa [17]:

- **Generacijska zamjena** - sve potomstvo nastalo u jednoj iteraciji zamjenjuje roditelje od kojih je nastalo, tj. oni se u potpunosti odbacuju.
  Budući da ova metoda pruža rizik da se najbolja jedinka populacije u jednoj iteraciji neće moći razmnožavati u sljedećoj jer je odbačena, taj se pristup često kombinira sa strategijama elitizma. To znači da se najbolja jedinka (ili nekoliko najboljih jedinki) populacije bezuvjetno kopira u populaciju sljedeće generacije.
- **Stacionarna reprodukcija** - prije prelaska u iduću generaciju odbacuje se dio jedinki trenutne, i to tako da se veličina populacije održi fiksnom. Najčešće odbačene jedinke budu one najlošijih vrijednosti dobrote.

4.3 Tok algoritma

Radi lakšeg razumijevanja, u nastavku najprije slijedi pseudokod svojstven svim implementacijama genetskih algoritama, a zatim i detaljniji uvid u isti.

```
POČETAK
Definiranje uvjeta terminacije
Definiranje inicijalne populacije
Izračun vrijednosti dobrote

RADI
    Odabir
    Križanje
    Mutacija
    Izračun vrijednosti dobrote
DOK uvjet terminacije nije zadovoljen
KRAJ
```

Genetski se algoritmi sastoje od triju faza: odabira, križanja i mutacije. Te se faze ciklički i iterativno odvijaju nad populacijom fiksne i konačne veličine sve dok se ne zadovolji definirani uvjet terminacije. Po završetku slijeda *odabir-križanje-mutacija* računaju se vrijednosti dobrote novodobivene populacije koje mogu (ali ne moraju) služiti za provjeru uvjeta terminacije ili kao varijable u idućoj generaciji.

Za uvjet terminacije najčešće se uzima stanje konvergencije, u kojem se potomci u novim generacijama ne razlikuju bitno od svojih predaka iz prethodnih generacija. Populacija posljednje iteracije smatra se setom rješenja zadanog problema. Osim tog kriterija, za terminaciju se može uzeti i sljedeće [14], [16]:

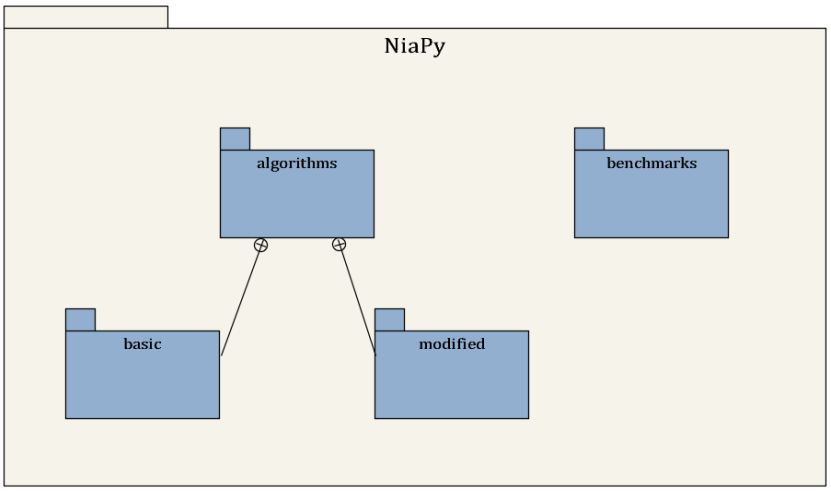
- proizveden je unaprijed zadani broj generacija
- pronalazak rješenja koje zadovoljava minimalne kriterije
- dosegnuti su zadani limiti resursa
- vrijednost dobrote jedinke s najvećom vrijednošću dobrote dosegla je teoretski vrhunac, nakon kojeg nema smisla tražiti bolje vrijednosti
- ručnim pregledom rezultata ocijenjeno je da su zadovoljavajući
- kombinacija gore navedenih pristupa

4.4 NiaPy

Programski okvir za implementaciju genetskog algoritma u ovom radu pruža NiaPy, Python mikro razvojni okvir (eng. *microframework*) za izgradnju algoritama inspiriranih prirodom [21]. U kod se uvozi kao knjižnica te pruža sve potrebne metode, strukture podataka i ostalu podršku za implementaciju, uspoređivanje i izvoz rezultata brojnih algoritama inspiriranih prirodom. Naravno, na svakom je programeru da kod dobiven iz knjižnice prilagodi svom problemu. NiaPy inženjeri iz organizacije NiaOrg kao glavnu motivaciju u razvoju ovog razvojnog okvira navode ograničenja postojećeg sličnog softvera u vidu nedovoljne dokumentacije, premalog broja algoritama, nekonzistentnih uvjeta terminacije te nemodularne ar-

hitekture [22].

Razvojni se okvir sastoji od algoritama, referentnih funkcija (eng. *benchmarks*) i značajki podrške koje se odnose na pokretanje algoritama i izvoz dobivenih rezultata [22]. Samim time, od velikog je razvojnog značaja ideja minimalizma i lakoće korištenja NiaPya; nastoji se stvoriti okvir koji je lako razumljiv i za čije korištenje nije potrebno pisati mnogo novog koda, a eventualne preinake u većini se slučajeva jednostavno implementiraju. Ipak, za potrebe ovog rada pokazalo se kako je mnoge metode potrebno *overrideati* kako bi se prilagodile potrebama konkretnog problema. Zbog visoke uparenosti NiaPy klasa, takve se promjene katkad pokažu kompliciranima za implementirati i potrebne su detaljne provjere pouzdanosti implementiranih promjena.



Slika 4.6 Pojednostavljeni dijagram strukture NiaPy razvojnog okvira
Izvor: NiaPy Github repozitorij [22]

4.4.1 NiaPy algoritmi

Algoritmi implementirani u sklopu NiaPy dijele se u sljedeće kategorije:

- osnovni (*Basic*) - osnovni oblici danih algoritama
- modificirani (*Modified*) - na neki način izmijenjene i/ili prilagođene inačice

osnovnih algoritama

- ostali (*Other*) - nekoliko drugih algoritama koji nisu nužno inspirirani prirodom

Za korištenje algoritma, isti se kreira kao objekt čiji konstruktor prima vrijednosti odgovarajućeg niza parametara (ili ih nadomješta zadanim vrijednostima). Svaki je algoritam realiziran kroz zasebnu klasu, a klase svih algoritama nasljeđuju klasu `Algorithm`, koja ostvaruje karakteristike zajedničke svim algoritmima.

Prije kreiranja objekta algoritma i pokretanja istog, potrebno je postaviti problem (`Problem`) i njime definirati zadatak optimizacije (`Task`).
Klasu `Problem` nasljeđuje klasa problema koji odgovara problemu kojeg algoritam nastoji optimizirati, a koju za tu potrebu programer sam definira (ili uzima jednu od već implementiranih). U njoj se definira dimenzija problema te funkcija evaluacije jedinke. Tu je moguće odrediti i gornju i donju granicu dozvoljenih vrijednosti problema.

Definirani se problem zatim šalje u konstruktor klase `Task`, kojom se određuje tok izvođenja zadatka - maksimalni broj iteracija i evaluacija, *cutoff* vrijednosti (ako postoje), funkcije popravka te je li u pitanju zadatak maksimizacije ili minimizacije. Kako cijeli razvojni okvir podrazumijeva minimizaciju kao zadani pristup, potonje služi određivanju hoće li se vrijednosti evaluacije invertirati množenjem faktorom -1.

Zatim se definira sam algoritam, pri čemu su ključne metode inicijalizacije algoritma i definiranja populacije. Konačno, algoritam se pokreće pomoću `run(task)` metode, čiji je parametar ranije definirani `Task` objekt.

4.4.2 NiaPy genetski algoritam

Definicija genetskog algoritma kao cjeline direktno ili indirektno podrazumijeva definiciju svih parametara navedenih u Potpoglavlju 4.2. U sklopu knjižnice već

su definirane funkcije za neke metode odabira (turnirski, rulet), križanja (uniformno, na dvije točke, na više točaka, modificirano) te mutacije (uniformna, *creep*, modificirana). Glavna je funkcionalnost NiaPy genetskog algoritma sadržana u metodi `run_iteration`, u kojoj se pozivanjem odgovarajućih metoda odvija tok algoritma na način predstavljen u Potpoglavlju 4.3.

Poglavlje 5

Rezultati

Rezultati ovog rada odnose se na skup rezultata triju etapa njegova izvođenja opisanih u ranijim poglavljima - pretragu trijadi, implementaciju genetskih algoritama temeljenu na NiaPy knjižnici te rezultate izvedbe i performanse samih algoritama.

5.1 Metode pretrage

U Tablici 5.1 naveden je broj katalitičkih trijada (ukupno i po enzimima) za svaku od metoda opisanih u Poglavlju 3. Sukladno očekivanjima, stroži kriteriji donose manji broj pronađenih trojki - najmanje je katalitičkih trijadi u hibridnoj metodi, a metodom zasnovanom na aminokiselinama pronalazi se manje sličnih trokuta negoli originalnom. Isto vrijedi i obratno - metoda zasnovana na atomima, najmanje strogih kriterija, pronalazi najveći broj prihvaćenih katalitičkih trijadi.

	Metoda pretrage					
	Originalna		***Atom-based***	***Residue-based***		**Hibridna**
Enzim	Trijade	Slični trokuti	Trijade	Trijade	Slični trokuti	Trijade
1agy	3	619	34	2	8	4
1aql	4	20648	64	4	1834	1
1auo	4	959	53	5	230	2
1bs9	4	1051	28	2	47	4
1din	3	606	38	1	163	3
1eh5	6	2709	48	4	498	3
1esc	1	4279	75	1	263	3
1hpl	2	25988	127	5	3329	1
1j00	2	428	21	2	45	3
1jkm	3	8461	123	4	1112	1
1ju3	2	42248	112	4	2963	2
1l7a	3	3449	61	3	364	4
1mah	1	14553	104	2	1544	3
1mpx	5	40561	140	7	3844	2
1pja	3	3612	61	3	663	3
1pp4	3	2496	47	3	363	3
1qe3	1	12396	57	3	1538	2
1r4z	3	517	22	2	77	2
1thg	1	22524	138	3	2513	2
1wab	3	944	37	2	312	3
1zoi	3	3803	86	4	652	3
2o7r	6	6294	59	4	816	1
ukupno	**66**	**219145**	**1535**	**70**	**23178**	**55**

Tablica 5.1 Broj katalitičkih trijada i sličnih trokuta u implementaciji originalne metode pretrage

5.2 Implementacija genetskog algoritma

Motivacija za primjenu genetskog algoritma u ovom radu leži u pitanju ima li kroz danu sekvencu (enzim) neki set kutova ima poseban značaj. Primjerice, postoji li među enzimima najčešći uzorak pronađene katalitičke trijade ili postoji li pak među trijadama uzorak onih kakve su zastupljene u najviše različitih enzima.
U tu je svrhu potrebno primijeniti algoritam koji omogućuje: (i) variranje promatranih parametara s ciljem pronalaženja optimalnih kombinacija, (ii) variranje područja ispitivanja s ciljem uočavanja mogućih razlika i zakonitosti u optimalnom skupu parametara, (iii) fleksibilnost na promjene u skupu podataka i/ili metodama pristupa.

Za potrebe ovog rada, NiaPy genetski algoritam nadograđen je i prilagođen osnovnim pitanjima od kojih rad polazi: (i) koji je uzorak gena kod trijada najčešći te (ii) koji je uzorak gena kod trijada zajednički svim enzimima.

5.2.1 Jedinka

Za modeliranje jedinki ovoga problema stvorena je `TriadIndividual` klasa. Svaka jedinka sadrži dvije varijeble: gene jednike i pripadajuću vrijednost dobrote, i to zato jer NiaPy metode zahtjevaju takvu definiciju. Nasljeđuje NiaPy klasu `Individual`, koja sadrži metode za rukovanje jedinkom i podacima o njoj.
Svaki gen neke jedinke odgovara reprezentaciji nekog njenog svojstva cijelim brojem, na način predstavljen Tablicom 5.2.

Gen	**Raspon vrijednosti**	**Objašnjenje**
Nuc	$0-1$	0 - OG Ser 1 - SG Cys
Acid	$0-1$	0 - OD1 1 - OD2
Base	$0-2$	0 - CG His 1 - CG Asp 2 - CG Glu
D1	$0-19$	kategorija u koju spada udaljenost *Acid-Base*
D2	$0-19$	kategorija u koju spada udaljenost *Base-Nuc*

Tablica 5.2 Geni jedinke u implementaciji algoritma

Kod gena čije vrijednosti nisu diskretne, tj. upadaju u raspon između neke minimalne i maksimalne vrijednosti dane varijable, koristi se pristup diskretizacije vrijednosti u zadane kategorije. Gen se izražava kao nenegativan cijeli broj kategorije k u koju spada vrijednost varijable koju opisuje, a računa se prema Jednadžbi 5.1, gdje je:

- max maksimalna vrijednost promatrane varijable u cijelom skupu podataka,
- v vrijednost promatrane varijable,
- r raspon mogućih vrijednosti promatrane varijable u cijelom skupu podataka,
- c željeni broj kategorija.

$$k = \texttt{floor}(1 - \frac{max - v}{r}) \times c \tag{5.1}$$

Pri određivanju kategorije koristi se 0-indeksiranje. Npr. ako neke vrijednosti upadaju u raspon od 0 do 99 Å, uz korištenje $c = 20$ kategorija, vrijednost $v = 31$ spadat će u kategoriju $k = 6$ (što je po redu ukupno 7. kategorija, ali njen je identitet „6”).

5.2.2 Problem

Ovisno o željenom rezultatu optimizacije, postavljaju se problemi koji se implementacijski razlikuju u pristupu računanja vrijednosti dobrote. U ovom se radu razlikuju dva pristupa, a koji traže najčešći, odnosno enzimima zajednički uzorak optimalnih jedinki.

Najčešći uzorak

Problem koji optimizira pretragu najčešćeg u skupu katalitičkih trijada i sličnih trokuta među svim enzimima modeliran je klasom `MostCommonPattern`. Za evaluaciju jedinke kao vrijednost funkcije dobrote uzima se broj jedinki istoga genoma u čitavom spomenutom skupu.

Zajednički uzorak

Problem koji optimizira pretragu uzoraka među katalitičkim trijadama i sličnim trokutima za koje postoji najveći broj enzima koji sadrže trijadu definiranu tim genima modeliran je klasom `EnzymeCommonPattern`. Za evaluaciju jedinke kao vrijednost funkcije dobrote uzima se broj enzima koji danu trijadu sadrže.

5.2.3 Populacija

Prethodno opisane komponente algoritma temelj za početak njegova izvođenja nalaze u formiranju populacije. Kao početna se populacija uzima skup jedinki za čiju je inicijalizaciju zadano da se 20% populacije nasumično odabire iz skupa trijada i sličnih trokuta. Preostalih se 80% nasumično uzima iz skupa svih mogućih permutacija promatranih parametra. Ukoliko je ukupan broj pronađenih trijadi manji od 20% veličine populacije, uvrštavaju se sve trijade te se ostatak popunjava jedinkama iz skupa permutacija.

Osim tog pristupa, na raspolaganju je i pristup u kojem se populacija u cijelosti odabire iz skupa permutacija. Ipak, prvi je pristup odabran jer realnije predstavlja

stvarno stanje.

Za programsku inicijalizaciju populacije algoritma potrebno je definirati veličinu populacije, koji se sve parametri (geni) koriste u definiciji te broj kategorija u enkodiranju udaljenosti i kutova.

5.2.4 Algoritam

Za implementaciju genetskog algoritma nad ovim problemom potrebno je definirati koji se tip odabira, križanja i mutacije događa te sa kojim stopama. Algoritam promatra $n = 5$ parametara te radi nad populacijom veličine 100, odabranom u skladu s veličinom seta permutacija svih mogućih jedinki, a koja iznosi $2 \times 2 \times 3 \times 20 \times 20 = 4800$.

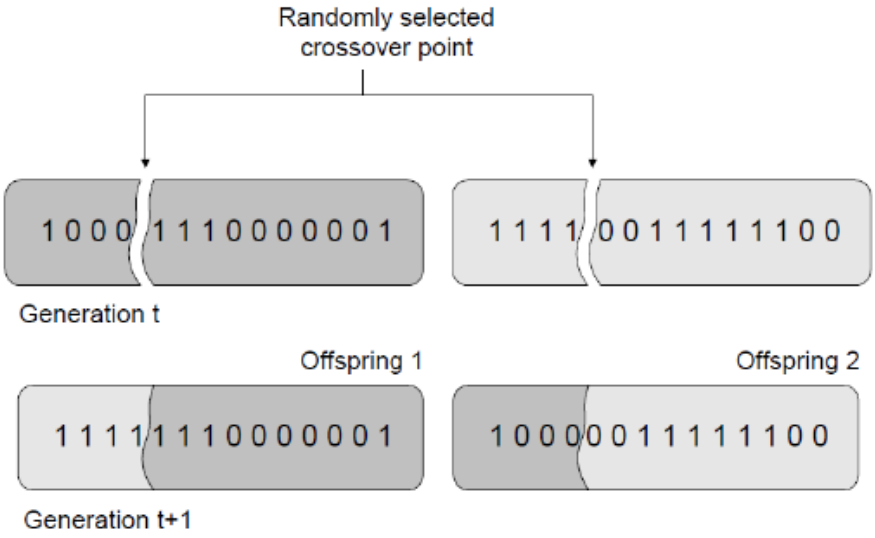
Odabir

Algoritam u izvođenju prolazi kroz cijelu populaciju te sa svakom jedinkom ponavlja proces odabira, križanja i mutacije. Proces se može ponavljati u više iteracija - idejno, do konvergencije.

Koristi turnirski tip odabira, veličine turnira 2. To znači da se svaka jedinka uspoređuje s nekom drugom, nasumično odabranom iz populacije. Ona koja ima bolju vrijednost dobrote koristit će se u daljnjim koracima. Obzirom na relativno malo veličinu populacije, ta je veličina turnira odabrana kako bi se i slabijim jedinkama pružila realna mogućnost za reprodukciju.

Križanje

Križanje se odvija na jednoj točki, tj. nasumično odabranom genu. Na jednoj se točki dva roditelja prelamaju te zamjenom nastalih dijelova nastaju potomci (Slika 5.1). Dakle, svaki potomak u genomu ima dio jednog roditelja do točke križanja i dio drugog roditelja nakon točke križanja [20].



Slika 5.1 Križanje jedinki na jednoj točki

Izvor: Neural Networks Optimization through Genetic Algorithm Searches: A Review [23]

Vjerojatnost križanja ovisi o stopi križanja, koja u ovom slučaju iznosi 0.9. To znači da za svaku jedinku koja ulazi u funkciju križanja i njoj različitu, nasumično iz populacije odabranu, drugu jedinku vjerojatnost da će doći do križanja iznosi 90%. Implementira se stacinarna reprodukcija: ukoliko do križanja dođe, nasljednici (eng. *offspring*) ne zamjenjuju roditelje, već se privremeno zapisuju u listu novonastale populacije. Po završetku iteracije cijelog algoritma iz skupa stare i novonastale populacije bira se broj najboljih jedinki koji odgovara zadanoj veličini populacije.

Mutacija

Mutacija se manifestira kao izmjena jednog, nasumično odabranog gena dane jedinke. On se mijenja novom vrijednošću iz zadanog raspona vrijednosti za taj gen, pri čemu je nova vrijednost nužno različita od prethodne. Mutacija se odvija uz stopu mutacije 0.01, odnosno 1% vjerojatnosti da će se dogoditi.

Terminacija

Za uvjet terminacije uzima se zadani NiaPy pristup, definiran u `Task` klasi. On nalaže da algoritam prestaje s izvođenjem kad se dosegne zadani broj zadanih generacija, evaluacija, ili se zadovolje zadane referentne vrijednosti ukoliko iste postoje. Svaki se algoritam (najčešći i zajednički uzorak) izvodi u 15 iteracija izvedbe („životnih vjekova" algoritma). Maksimalni dozvoljeni broj iteracija (generacija) u jednoj iteraciji izvedbe algoritma za najčešći uzorak iznosi 15, a za zajednički 10. Taj je broj određen eksperimentalno jer se pokazalo da će do tog broja generacija vrijednosti s gotovo potpunom sigurnošću konvergirati konačnom rezultatu. Broj dozvoljenih iteracija manji je za algoritam za zajednički uzorak jer zbog malenog diverziteta vrijednosti dobrote algoritam brzo konvergira, nakon čega iteracije postaju spore jer se odvijaju nad populacijama niskog diverziteta jedinki.

Kroz iteracije (generacije) se prate vrijednosti minimalne, prosječne i maksimalne dobrote unutar populacije. Po završetku svake iteracije izvedbe, ti se podaci zapisuju u `csv` datoteku te skiciraju na grafocima. Jedinka prepoznata kao optimalna se zajedno sa svojom vrijednošću dobrote zapisuje u algoritmu pripadajuću `csv` datoteku.

Završetkom svih iteracija izvedbe oba algoritma radi se dodatna analiza netom dobivenih podataka: prebrojavanje dobivenih trijadi i sličnih trokutova, kretanje vrijednosti funkcije dobrote najuspješnije jedinke po iteracijama izvedbe, analiza učestalosti najuspješnijih jedinki te izračun njihove međusobne sličnosti. Sve je ovo detaljnije predstavljeno u narednim potpoglavljima.

5.3 Najčešći uzorak

Petnaest je iteracija izvedbe algoritma za najčešći uzorak kao rezultat pretrage dalo pet različitih rezultata, od kojih se dva pojavljuju sedam, odnosno pet puta. Očekivano, to su ujedno jedinke najvećih dobrota u skupu rezultata prikazanom

Tablicom 5.3.

Nuc	**Acid**	**Base**	**D1** [Å]	**D2** [Å]	**Dobrota**	**Broj**
OG Ser	OD1	CG Asp	$28.5958 \leq D1 < 32.0179$	$24.3917 \leq D2 < 27.9395$	1736.0	7
OG Ser	OD2	CG Asp	$32.0179 \leq D1 < 35.4399$	$27.9395 \leq D2 < 31.4872$	1559.0	5
OG Ser	OD1	CG Asp	$28.5958 \leq D1 < 32.0179$	$20.8439 \leq D2 < 24.3917$	1292.0	1
OG Ser	OD1	CG Asp	$35.4399 \leq D1 < 38.8620$	$27.9395 \leq D2 < 31.4872$	1341.0	1
OG Ser	OD2	CG Asp	$28.5958 \leq D1 < 32.0179$	$24.3917 \leq D2 < 27.9395$	1576.0	1

Tablica 5.3 Geni, dobrota i učestalosti najuspješnijih jedinki u 15 iteracija algoritma za najčešći uzorak

5.4 Zajednički uzorak

Različitih rezultata petnaest iteracija izvedbe algoritma za zajednički uzorak ima više negoli kod algoritma za najčešći uzorak - njih deset. Nema većih odskakanja u broju pojavljivanja kao rezultat pretrage, to jest, kako sve pronađene jedinke imaju istu vrijednost dobrote, češće pojavljivanje neke od njih može se naprosto pripisati slučaju.

Nuc	**Acid**	**Base**	**D1** [Å]	**D2** [Å]	**Dobrota**	**Broj**
OG Ser	OD1	CG Asp	$25.1752 \leq D1 < 28.5958$	$20.8439 \leq D2 < 24.3917$	22.0	3
OG Ser	OD2	CG Asp	$18.3306 \leq D1 < 21.7529$	$17.2961 \leq D2 < 20.8439$	22.0	2
OG Ser	OD2	CG Asp	$21.7529 \leq D1 < 25.1752$	$17.2961 \leq D2 < 20.8439$	22.0	2
OG Ser	OD2	CG Asp	$25.1752 \leq D1 < 28.5958$	$20.8439 \leq D2 < 24.3917$	22.0	2
OG Ser	OD1	CG Asp	$21.7529 \leq D1 < 25.1752$	$17.2961 \leq D2 < 20.8439$	22.0	1
OG Ser	OD2	CG Asp	$21.7529 \leq D1 < 25.1752$	$20.8439 \leq D2 < 24.3917$	22.0	1
OG Ser	OD2	CG Asp	$25.1752 \leq D1 < 28.5958$	$17.2961 \leq D2 < 20.8439$	22.0	1
OG Ser	OD2	CG Asp	$25.1752 \leq D1 < 28.5958$	$24.3917 \leq D2 < 27.9395$	22.0	1
OG Ser	OD2	CG Glu	$14.9083 \leq D1 < 18.3306$	$13.7483 \leq D2 < 17.2961$	22.0	1
OG Ser	OD2	CG Glu	$25.1752 \leq D1 < 28.5958$	$24.3917 \leq D2 < 27.9395$	22.0	1

Tablica 5.4 Geni, dobrota i učestalosti najuspješnijih jedinki u 15 iteracija algoritma za zajednički uzorak

5.4.1 Sličnost rezultata

Sličnost skupa rezultata promatra se preko Levenshteinove udaljenosti (eng. *edit distance*). U računarstvu i teoriji informacija, Levenshteinova se udaljenost dvaju

nizova određuje kao minimalan broj operacija potreban da se jedan u njih pretvori u onaj drugi. Dozvoljene su operacije brisanja, ubacivanja i zamjene jedinica niza [24].

Levenshteinova udaljenost $\mathrm{lev}_{a,b}(|a|,|b|)$ nizova a i b duljina $|a|$ i $|b|$ izračunava se prema Jednadžbi 6.2. Izračunava se rekurzivnim ispitivanjem Levenshteinove udaljenosti prvih i elemenata niza a i j elemenata niza b. Karakteristične operacije izračuna Levenshteinove udaljenosti - brisanje, ubacivanje i zamjena - tim su redom predstavljene slučajevima *min* bloka [25]. Iako po matematičkoj definiciji rekurzivna, programski se Levenshteinova udaljenost može izračunavati i iterativno, kako je predstavljeno u nastavku.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \min(i,j) = 0, \\ 1 + \min \begin{cases} \mathrm{lev}(i-1,j) + 1 \\ \mathrm{lev}(i,j-1) + 1 \\ \mathrm{lev}(i-1,j-1)) + 1_{a_i \neq b_j} \end{cases} & \text{inače} \end{cases} \tag{5.2}$$

Kako se najčešće primjenjuje u uspoređivanju znakovnih nizova (u pravilu riječi), jedinice u pitanju obično su slova ili alfanumerički znakovi. U ovom slučaju gleda se sličnost jedinki čiji je zapis numeričko polje, pa su jedinice u pitanju cijeli brojevi. Pritom se Levenshteinova udaljenost izračunava koristeći isječak koda predstavljen u Listingu 1.

Sličnost liste trijada izračunava se tako da se izračuna na dvije decimale zaokružena aritmetička sredina Levenshteinovih udaljenosti svih kombinacija dane liste. Manje vrijednosti upućuju na veću sličnost, i obrnuto. Najmanja moguća vrijednost sličnosti iznosi 0.0, što znači da su svi članovi liste identični - dakle, broj potrebnih zamjena njihovih elemenata je 0. Najveća je moguća sličnost u ovom sličaju 5.0, u situaciji gdje dvije jedinke nemaju nijedan zajednički gen (sve ih je potrebno zaijeniti).

```
def levenshtein(a, b):
    n, m = len(a), len(b)
    if n > m:
        # Make sure n <= m, to use O(min(n,m)) space
        a,b = b,a
        n,m = m,n

    current = range(n+1)
    for i in range(1, m+1):
        previous, current = current, [i]+[0]*n
        for j in range(1, n+1):
            add, delete = previous[j]+1, current[j-1]+1
            change = previous[j-1]
            if a[j-1] != b[i-1]:
                change = change + 1
            current[j] = min(add, delete, change)

    return current[n]
```

Listing 1 : Funkcija za izračunavanje Levenshteinove udaljenosti dvaju lista [26]

Iako je za potrebe ovog rada Levenshteinova udaljenost jednaka broju različitih gena dvaju jedinki, računanje sličnosti dvaju jedinki nije implementirano na taj, programski nezahtjevniji način. Kako se u daljnjem radu na ovom projektu želi ispitati provedbu genetskih algoritama nad jedinkama koje čini po definiciji drugačiji skup gena (širi ili uži), ovaj pristup pruža veću modularnost u usporedbi dvaju ili više skupova izlaznih podataka pretrage.

Sličnost najboljih jedinki

Analizirana je sličnost triju skupova najuspješnijih jedinki:

- rezultati (top 1 najbolje jedinke) svih iteracija izvedbe algoritama - Tablica 5.5

- top 10 najboljih jedinki zadnje generacije svih iteracija izvedbe algoritama - Tablica 5.6
- top 10 naboljih jedinki u zadnjoj generaciji jedne iteracije izvedbe algorita - Tablica 5.7

Budući da u svim promatranim slučajevima postoji postoji visok stupanj sličnosti genoma najboljih jedinki (Levenshteinova udaljenost manja od 2), može se zaključiti kako postoje kategorije gena koje u pravilu donose bolje rezultate pretrage. Relativno mala razlika sličnosti najboljih jedinki (Tablica 5.5) i najboljih 10 najboljih jedinki poslijednjih generacija (Tablica 5.6) ukazuje na konzistentnost rezultata pretrage i visoku razinu konvergencije čitave populacije. Potonjem svjedoče i vrijednosti sličnosti najboljih 10 jedinki poslijednje generacije jedne iteracije (Tablica 5.7).

Kao mogući uzrok veće sličnosti među konačnim rezultatima potrage za najčešćim uzorkom u odnosu na sličnost među najboljim jedinkama posljednje generacije prepostavlja se širi raspon mogućih vrijednosti u funkciji dobrote, uslijed kojeg algoritam širi područje pretrage. Obratno vrijedi za algoritam za zajednički uzorak, kod kojeg postoje tek 23 moguće vrijednosti funkcije dobrote - dakle, mnoge jedinke dijele istu vrijednost dobrote i opstanak jedne od više takvih jedinki u populaciji nerijetko biva prepušten slučajnosti, tj. ne ovisi o rezultatu vrednovanja.
Zanimljivo je primijetiti kako u Tablici 5.7 kod agoritma za zajednički uzorak postoji samo jedna iteracija u kojoj najboljih 10 jedinki nije potpuno jednako, a sve to zahvaljujući upravo opisanoj pojavi.

Algoritam	**Sličnost**
Najčešći uzorak	1.92
Zajednički uzorak	1.58

Tablica 5.5 Sličnost najboljih jedinki izražena prosjekom Levehensteinovih udaljenosti

Algoritam	**Sličnost**
Najčešći uzorak	2.03
Zajednički uzorak	1.67

Tablica 5.6 Sličnost 10 najboljih jedinki svih poslijednjih generacija algoritma izražena prosjekom Levehensteinovih udaljenosti

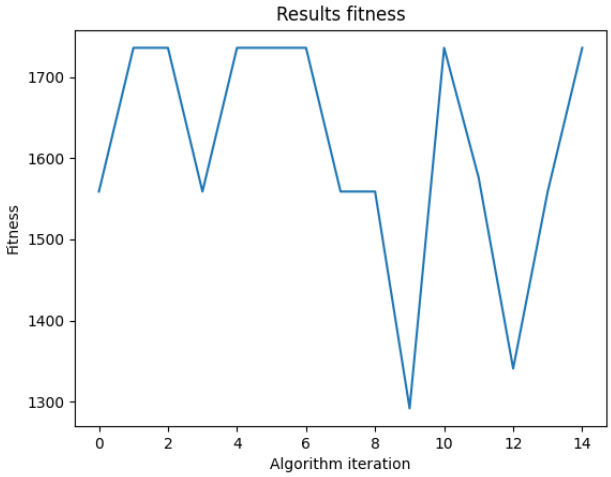
Iteracija	**Sličnost (najčešći)**	**Sličnost (zajednički)**
0	1.09	0.0
1	1.11	0.0
2	1.4	0.0
3	1.73	0.0
4	1.67	0.0
5	1.0	0.0
6	1.49	0.0
7	0.56	0.0
8	1.42	0.0
9	1.31	0.0
10.	1.62	0.0
11	2.07	0.67
12	1.56	0.0
13	1.36	0.0
14	1.18	0.0

Tablica 5.7 Sličnost 10 najboljih jedinki u poslijednjim generacijama algoritma za najčešći i zajednički uzorak izražena prosjekom Levenshteinovih udaljenosti

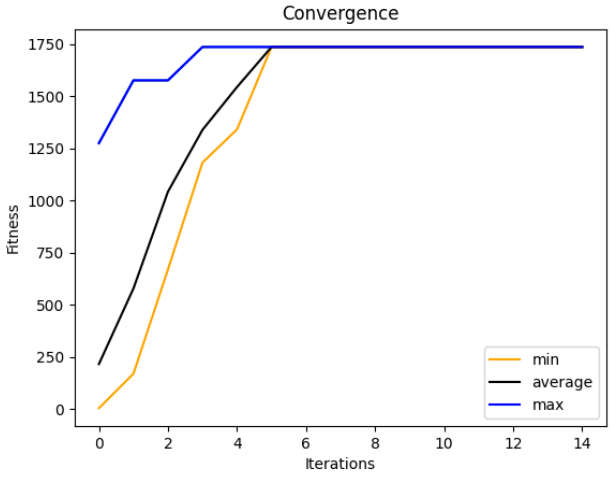
5.5 Vrijednosti funkcije dobrote

Za vrijeme izvođenja algoritma, u svakoj se iteraciji (generaciji) prikupljaju podaci o minimalnoj, prosječnoj i maksimalnoj vrijednosti dobrote te iteracije. Sve one vremenom konvergiraju ka iznosu vrijednosti iznosa dobrote jedinke prepoznate kao konačan rezultat pretrage. Po završetku izvedbi algoritama, stvara se grafički prikaz navedene konvergencije.

Također, u sklopu analize dobivenih podataka koja se vrši na kraju programa, stvaraju se i grafički prikazi kretanja vrijednosti dobrote najbolje jedinke sviju iteracija izvedbe algoritma. Oba navedena prikaza vidljiva su na Slikama 5.2 i 5.3.

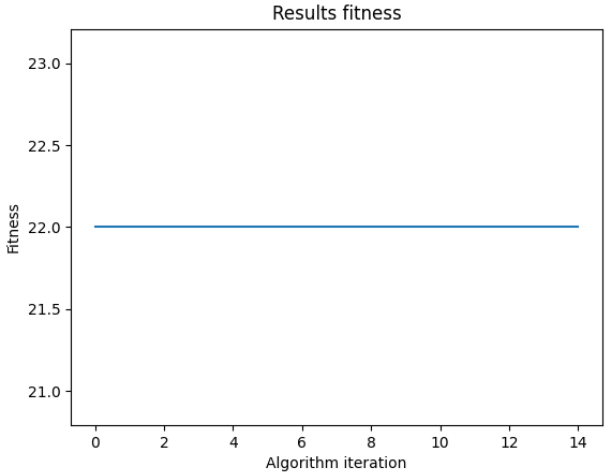


(a) Vrijednost dobrote konačnog rezultata po izvedbama algoritma

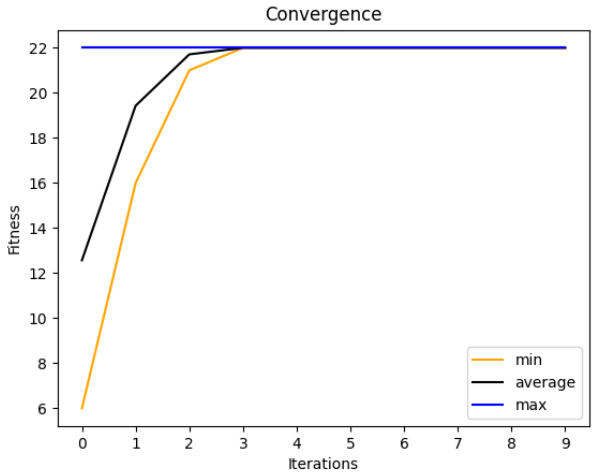


(b) Konvergencija minimalne, prosječne i maksimalne dobrote u populaciji kroz generacije na primjeru jedne izvedbe

Slika 5.2 Grafički prikazi vrijednosti dobrote kod algoritma za najčešći uzorak



(a) Vrijednost dobrote konačnog rezultata po izvedbama algoritma



(b) Konvergencija minimalne, prosječne i maksimalne dobrote u populaciji kroz generacije na primjeru jedne izvedbe

Slika 5.3 Grafički prikazi vrijednosti dobrote kod algoritma za zajednički uzorak

Poglavlje 6

Zaključak

Kemija i biotehnologija grane su znanosti koje sve veću podršku svojim istraživanjima nalaze u primjeni suvremenih računarskih metoda i tehnologija. U tom je duhu i ovaj projekt za cilj imao automatizirati, ubrzati i olakšati pretragu strukturnih obrazaca enzima, odnosno traženje katalitičkih trijadi i njima sličnih trokutova. Značaj dobivenih rezultata konkretiziran je obradom istih korištenjem dvaju genetskih algoritama - algoritma za zajednički uzorak i algoritma za najčešći uzorak.

Tehnički je oblikovan i implementiran program za pripremu `pdb` zapisa enzima, pretragu katalitičkih trijadi i sličnih trokutova u obrađenim podacima, provedbu genetskih algoritama nad istima te analizu rezultata algoritama. U radu je opisan, a rezultati svih njegovih etapa analizirani, uz teoretsku podršku predstavljanjem pojmova katalitičkih trijadi i genetskih algoritama. Dobiveni je program učinjen dovoljno fleksibilnim na nadolazeće promjene uslijed pitanja i novih pristupa koji se tokom istraživanja nameću.

Četiri su metode pretrage u svim enzimima pronašle katalitičke trijade i slične trokutove, kod metoda koje i njih pretražuju, a broj dobivenih rezultata razlikuje se ovisno o strogosti kriterija pretrage. Najviše je katalitičkih trijadi pronađeno metodom zasnovanom na atomima (1535), a najmanje hibridnom metodom (55). Genetski su algoritmi, za najčešći i za zajednički uzorak, implementirani osla-

njanjem na razvojni okvir NiaPy te provedeni u petnaest ponovljenih mjerenja. Algoritam za najčešći uzorak u tih je petnaest iteracija kao najbolju prepoznao pet različitih jedinki, od kojih se najčešće pojavljivala jedinka *OG Ser-OD1-CG Asp-8-16* ukupnog broja pojavljivanja na 1736 mjesta unutar 22 enzima - čak sedam od petnaest puta. Algoritmom za zajednički uzorak pronalazi deset različitih trokuta koji se pronalaze u svih 22 enzima, od kojih je najčešća *OG Ser-OD1-CG Asp-7-5* sa tri pojavljivanja. Pri tome valja napomenuti kako postoji veliki broj jedinki koje imaju maksimalnu vrijednost dobrote te bi u budućem radu moglo razmotriti opciju uvođenja dodatnog kriterija pretrage, kako bi se moglo gradirati jednako dobra rješenja. U dobivenim rezultatima postoji značajna sličnost na razini svih najboljih jedinki jednog algoritma te na razini deset najboljih jedinki posljednjih populacija svih izvedbi algoritma. Rana konvergencija algoritama rezultira izrazito visokom sličnošću najboljih deset jedinki jedne iteracije algoritma, posebice kod algoritma za zajednički uzorak gdje su navedene jedinke u gotovo svim iteracijama potpuno jednake.

Buduće istraživanje može biti usmjereno na izmjenu prostora pretraživanja koje je u ovoj studiji slučaja bila ograničena na temelju prethodne ručne analize i znanja iz domene problema. Također, od interesa može biti istražiti razloge nepoželjne pojave prerane konvergencije i suboptimalnog ponašanja genskog pomaka (eng. *genetic drift*), pojave da se učestalost nekog gena u populaciji mijenja pukom slučajnošću. Genski je pomak nevođen vanjskim faktorima i kao takav svojstven nedeterminističkim algoritmima. Može se reći da je genski pomak od jednake važnosti kao i odabir, a u svakom je slučaju faktor koji treba dodatno uzeti u obzir iz perspektive u ovom radu predloženih problema [27].

Rezultati dobiveni u ovom radu potvrđuju ispravnost metode pretrage katalitičke trijade, koja je zajednička u svih 22 enzima karaktestične evolucijske građe. Doprinos razvijene metodologije za pretraživanje najbrojnijih uzoraka i onih koji su zajednički čim većem broju proteina je utoliko veća jer je spremna za primjenu na mnogo većim skupovima podataka. Primjerice, uvrštavanjem svih homologa izvorno odabranih enzima može povećati skup podataka na tisuće zapisa. Brzi

algoritam pretraživanja bi u tom slučaju bio od neizmjerne važnosti.

Bibliografija

[1] G. Dodson i A. Wlodawer, „Catalytic triads and their relatives," *Trends in biochemical sciences*, sv. 23, br. 9, str. 347–352, 1998.

[2] W. Torng i R. B. Altman, „3D deep convolutional neural networks for amino acid environment similarity analysis," *BMC bioinformatics*, sv. 18, br. 1, str. 1–23, 2017.

[3] A. R. Buller i C. A. Townsend, „Intrinsic evolutionary constraints on protease structure, enzyme acylation, and the identity of the catalytic triad," *Proceedings of the National Academy of Sciences*, sv. 110, br. 8, E653–E661, 2013.

[4] S. Osuna, G. Jimenez-Oses, E. L. Noey i K. Houk, „Molecular dynamics explorations of active site structure in designed and evolved enzymes," *Accounts of chemical research*, sv. 48, br. 4, str. 1080–1089, 2015.

[5] M. Babić, „Evaluacija aktivnih mjesta enzima i usporedba sa kataliticki aktivnim peptidima koji kataliziraju hidrolizu estera," disertacija, University of Rijeka. Department of Biotechnology, 2021.

[6] L. Polgár, „The catalytic triad of serine peptidases," *Cellular and molecular life sciences CMLS*, sv. 62, br. 19, str. 2161–2172, 2005.

[7] A. Rauwerdink i R. Kazlauskas, „How the same core catalytic machinery catalyzes seventeen different reactions: the Ser-His-Asp catalytic triad of α/β-hydrolase fold enzymes," *ACS Catal*, sv. 5, br. 10, str. 6153–6176, 2015.

[8] S. C. Kamerlin i A. Warshel, „At the dawn of the 21st century: Is dynamics the missing link for understanding enzyme catalysis?" *Proteins: Structure, Function, and Bioinformatics*, sv. 78, br. 6, str. 1339–1375, 2010.

[9] T. Shafee, „Evolvability of a viral protease: experimental evolution of catalysis, robustness and specificity," disertacija, University of Cambridge, 2014.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.

[11] E.-G. Talbi, „A taxonomy of hybrid metaheuristics," *Journal of heuristics*, sv. 8, br. 5, str. 541–564, 2002.

[12] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz i A. Cosar, „A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, sv. 137, str. 106 040, 2019.

[13] A. Ghaheri, S. Shoar, M. Naderan i S. S. Hoseini, „The applications of genetic algorithms in medicine," *Oman medical journal*, sv. 30, br. 6, str. 406, 2015.

[14] M. J. Kochenderfer i T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[15] V. Mallawaarachchi, *Introduction to genetic algorithms - including example code*, ožujak 2020. adresa: `https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3`.

[16] *Genetic algorithm*, lipanj 2022. adresa: `https://en.wikipedia.org/wiki/Genetic_algorithm` (pogledano 28. 6. 2022.).

[17] K.-S. Tang, K.-F. Man, S. Kwong i Q. He, „Genetic algorithms and their applications," *IEEE signal processing magazine*, sv. 13, br. 6, str. 22–37, 1996.

[18] D. E. Goldberg i K. Deb, „A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, sv. 1, Elsevier, 1991., str. 69–93.

[19] I. De Falco, A. Della Cioppa i E. Tarantino, „Mutation-based genetic algorithm: performance evaluation," *Applied Soft Computing*, sv. 1, br. 4, str. 285–299, 2002.

[20] A. J. Umbarkar i P. D. Sheth, „Crossover operators in genetic algorithms: a review.," *ICTACT journal on soft computing*, sv. 6, br. 1, 2015.

[21] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister i I. Fister Jr., „NiaPy: Python microframework for building nature-inspired algorithms," *Journal of Open Source Software*, sv. 3, 23 2018., ISSN: 2475-9066. DOI: 10.21105/joss.00613. adresa: https://doi.org/10.21105/joss.00613.

[22] ——, *NiaPy: Python microframework for building nature-inspired algorithms*, verzija 1.0.2, listopad 2018. DOI: 10.21105/joss.00613. adresa: https://github.com/NiaOrg/NiaPy.

[23] H. Chiroma, S. Abdulkareem, A. Abubakar i T. Herawan, „Neural networks optimization through genetic algorithm searches: a review," *Appl. Math. Inf. Sci*, sv. 11, br. 6, str. 1543–1564, 2017.

[24] L. Yujian i L. Bo, „A normalized Levenshtein distance metric," *IEEE transactions on pattern analysis and machine intelligence*, sv. 29, br. 6, str. 1091–1095, 2007.

[25] S. Grashchenko, *Levenshtein distance computation*, studeni 2021. adresa: https://www.baeldung.com/cs/levenshtein-distance-computation (pogledano 18. 8. 2022.).

[26] *NTNU: Norwegian University of Science and Technology — ntnu.no*, 2019. adresa: https://folk.idi.ntnu.no/mlh/hetland_org/ (pogledano 18. 8. 2022.).

[27] G. Dick i P. Whigham, „The behaviour of genetic drift in a spatially-structured evolutionary algorithm," *2005 IEEE Congress on Evolutionary Computation*, IEEE, sv. 2, 2005., str. 1855–1860.

Sažetak

U ovom su radu opisana svojstva i značaj katalitičkih trijadi za hidrolizu estera, koji počiva na prostornoj raspoređenosti njihovih atoma. Iz geometrijskih karatkeristika katalitičkih trijadi i njima sličnih trokutova proizlaze četiri metode pretrage istih u enzimima. Rezultati pretrage originalnom metodom potvrđeni su genetskim algoritmom za pronalaženje najčešćeg i zajedničkog uzorka. Svi dobiveni rezultati dodatno su programski obrađeni i analizirani. Za potrebe rada sastavljen je program koji obavlja čitav opisan proces, a koji je spreman za nadogradnju u vidu daljnjeg rada na projektu.

Ključne riječi **— katalitičke trijade, enzimi, genetski algoritmi**

Abstract

This paper describes the properties and significance of catalytic triads for ester hydrolysis, which is based on the spatial distribution of their atoms. From the geometric characteristics of catalytic triads and triangles similar to them, four methods of searching for them in enzymes arise. The search results of one of the methods (the old method) were processed with genetic algorithms to find the most common and enzyme-common pattern. All obtained results were additionally processed and analyzed by software. For the needs of the work, a program that performs the entire described process was made, and is ready to be upgraded in the form of further work on the project.

Keywords **— catalytic triads, enzymes, genetic algorithms**