Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Diplomski rad br. 54

Primjena genetskog programiranja za rješavanje problema prijanjanja proteina

Ivan Kokan

Zagreb, lipanj 2010.

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Diplomski rad br. 54

Primjena genetskog programiranja za rješavanje problema prijanjanja proteina

Ivan Kokan

Zagreb, lipanj 2010.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 1. ožujka 2010.

DIPLOMSKI ZADATAK br. 54

Pristupnik: **Ivan Kokan**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Primjena genetskog programiranja za rješavanje problema prijanjanja proteina**

Opis zadatka:

Opisati postupak optimiranja uz pomoć genetskog programiranja. Proučiti načine zapisa proteinskih sljedova u bazi podataka. Definirati problem prijanjanja proteina sa svim potrebnim ograničenjima. Predložiti prikaz, način evaluacije tj. funkciju dobrote i genetske operatore koji su prikladni za rješavanje problema prijanjanja proteina. Ostvariti programsko rješenje u okviru radnog okruženja za evolucijsko računanje (engl. Evolutionary Computation Framework, ECF). Ispitati predloženi optimizacijski algoritam na nekoliko primjera problema i usporediti rezultate dobivene nekim slobodno dostupnim alatom iste namjene.

Zadatak uručen pristupniku: 5. ožujka 2010.
Rok za predaju rada: 18. lipnja 2010.

Mentor:
Doc.dr. sc. Marin Golub

Djelovođa:
Doc.dr.sc. Domagoj Jakobović

Predsjednik povjerenstva za diplomski ispit:
Prof.dr.sc. Siniša Srbljić

Zahvale

Zahvaljujem mentoru doc. dr. sc. Marinu Golubu na velikoj stručnoj pomoći i znanju koje mi je pružio prilikom izrade ovog rada. Također zahvaljujem doc. dr. sc. Domagoju Jakoboviću na pomoći u razumijevanju i razvoju radnog okruženja ECF, kao i na ostalim korisnim savjetima.

Veliku zahvalu zaslužuju i moji prijatelji — Saša, Inja i Marino — bez čije bi pomoći i savjeta u biološkom području izrada ovog rada bila znatno teža. Prijateljici Riti zahvaljujem na brojnim jezičnim savjetima i lektoriranju teksta.

Zahvaljujem i svim svojim učiteljima, nastavnicima i profesorima koji su me pratili tijekom dosadašnjeg školovanja.

Konačno, zahvaljujem majci Senki, ocu Luki, bratu Mati i ostalim najmilijima na pruženoj ljubavi i razumijevanju. Ovaj rad posvećujem upravo njima.

Sažetak

Primjena genetskog programiranja za rješavanje problema prijanjanja proteina

Genetsko programiranje jedna je od metoda evolucijskog računanja, koje se temelje na biološkim principima. Rezultat evolucijskog procesa genetskog programiranja je računalni program, što genetsko programiranje uvelike razlikuje od ostalih metoda. Iznesene su teorijske postavke genetskog programiranja, kao i osnove problema prijanjanja proteina. Ostvareno je programsko rješenje problema u okviru radnog okruženja ECF. Konačno, izneseni su rezultati ispitivanja i usporedba s programskim alatom Hex.

Ključne riječi

genetsko programiranje, ECF, algoritam, evolucija, prijanjanje proteina

Summary

Solving Protein Docking Problem with Genetic Programming

Genetic programming is one of the methods in evolutionary computation, based on biological principles. The result of evolutionary process of genetic programming is a computer program, which makes it distinct from other methods. The basic theoretical frames of genetic programming have been introduced, as well as the basis for protein docking problem. An application, based on ECF framework, has been developed in order to solve the protein docking problem. Finally, the testing results and the comparison with the Hex tool have been carried out.

Keywords

genetic programming, ECF, algorithm, evolution, protein docking

Sadržaj

Poglavlje 1

Uvod

Širina spektra inženjerskih problema smanjuje vjerojatnost da su njihova rješenja jednostavna, a kamoli egzaktna. Sukladno toj činjenici, svijet modernog računarstva sve više se okreće empirizmu, čak i u slučaju eksplicitno riješenih problema (najčešće zbog složenosti rješenja).

Temelj evolucijskog računanja (eng. *evolutionary computation*), čijoj obitelji pripada i **genetsko programiranje** (eng. *genetic programming*), su pojave iz biološke evolucije [24]. Preciznije gledano, genetsko programiranje spada u užu obitelj evolucijskih algoritama, koje odlikuju analogoni bioloških procesa poput reprodukcije, rekombinacije, mutacije, selekcije i sl. [23]. Iako na prvi pogled ne postoji jasno vidljiva poveznica između spomenutih domena, nakon kraćeg razmatranja ipak se dolazi do zaključka kako su principi **darvinizma** — "bolji opstaje, lošiji izumire" — prikladni za pronalaženje dovoljno dobrog rješenja zadanog problema (ako ne i u potpunosti točnog).

Iako se teorijski razvijala usporedno s ostalim evolucijskim metodama još od pedesetih godina prošlog stoljeća, metoda genetskog programiranja svoju pravu vrijednost dosegla je tek u devedesetima, kada su teorijski koncepti potpomognuti računalnom revolucijom zaživjeli u pravom smislu. Najveću ulogu u izdvajanju genetskog programiranja kao zasebne metode imao je **John Reed Koza**, doajen računarske znanosti sa Sveučilišta Stanford. Stalni napredak u razvoju i široka primjena prolongirali su genetsko programiranje kao jednu od vodećih sila **umjetne inteligencije** (eng. *artificial intelligence*) [22].

Ovaj rad dotiče se konkretne primjene genetskog programiranja u rješavanju problema prijanjanja proteina, a svojevrsno je finale seminara, projekata i završnog rada s prijašnjih kolegija [7, 12, 13, 14]. Ostvareno je programsko rješenje problema u okviru radnog okruženja ECF te su izneseni rezultati ispitivanja i usporedbe s alatom Hex [18].

Poglavlje 2

Genetsko programiranje kao metoda optimiranja

"How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?"

Arthur Lee Samuel[1]*, 1959.*

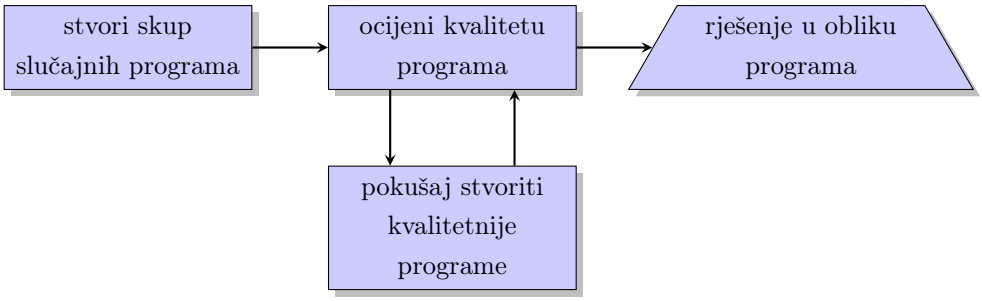
"Kako računala mogu naučiti rješavati probleme, a da ih se ne programira eksplicitno? Drugim riječima, kako računala mogu napraviti ono što trebaju napraviti, a da ih se u potpunosti ne uputi kako to napraviti?"

2.1 Uvod u genetsko programiranje

Genetsko programiranje (eng. *genetic programming*, GP) automatizirana je metoda optimiranja računalnih programa, čija je namjena rješavanje složenih problema iz područja računarstva, ali i problema iz drugih istraživačkih područja. Koncept je zasnovan na općim idejama proizašlima iz teorije evolucijskih algoritama (eng. *evolutionary algorithms*), kao i drugih evolucijskih metoda. Cilj genetskog programiranja je stvaranje **računalnih programa** koji rješavaju zadane probleme.

Pojednostavljeni shematski prikaz genetskog programiranja dan je na Slici 2.1.

[1] Arthur Lee Samuel (1901. – 1990.), američki računarski znanstvenik



Slika 2.1. Pojednostavljeni shematski prikaz genetskog programiranja.

2.2 Povijest genetskog programiranja

Počeci genetskog programiranja vezani su za evolucijske algoritme i primjenu u evolucijskim simulacijama (Nils Aall Barricelli[2], 1954.). Šezdesetih i sedamdesetih godina evolucijski algoritmi naglo se razvijaju i učvršćuju svoje mjesto u svijetu metoda optimiranja, a tomu svjedoče primjene u rješavanju raznih složenih inženjerskih problema poput optimiranja aerodinamike zrakoplovnih krila (Ingo Rechenberg[3], 1971.) i izgradnje automata s konačnim brojem stanja (Lawrence Jerome Fogel[4], 1964.), kao i klasifikatorskih sustava [25].

Prekretnica u razvoju genetskog programiranja dogodila se 1985., kada je Nichael Lynn Cramer u radu *A Representation for the Adaptive Generation of Simple Sequential Programs* [6] uveo koncept **stablaste jedinke** evolucijskog algoritma. Preuzevši taj koncept na prijelazu u posljednje desetljeće prošlog stoljeća, John Reed Koza[5] konačno je izdvojio genetsko programiranje kao zasebnu metodu optimiranja i tako postao pionirom GP-a. Njegova knjiga *Genetic Programming: On the Programming of Computers by Means of Natural Selection* [15] danas se smatra temeljnom literaturom genetskog programiranja.

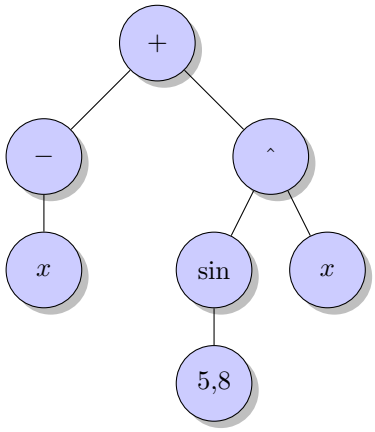
Unatoč ovisnosti o računalnoj moći tijekom prvih nekoliko godina "samostalnosti", koncept genetskog programiranja se održao te u posljednjih petnaestak godina ostvario zapažene rezultate u područjima kao što su: kvantno računarstvo, teorija igara, sortiranje, pretraživanje itd. Uspješnost dokazuju i brojni patenti temeljeni na GP-u [16].

[2]Nils Aall Barricelli (1912. – 1993.), norveško-talijanski matematičar
[3]Ingo Rechenberg (1934. –), njemački računarski znanstvenik
[4]Lawrence Jerome Fogel (1928. – 2007.), američki računarski znanstvenik
[5]John Reed Koza (1943. –), američki računarski znanstvenik

2.3 Formalni prikaz računalnog programa

Glavna razlika između genetskog programiranja i ostalih evolucijskih algoritama je u prikazu jedinke evolucijskog algoritma. Naime, umjesto linearne binarne jedinke, genetsko programiranje ima stablastu jedinku u kontekstu teorije grafova. Svaki računalni program može se prikazati kao stablo (eng. *tree*) ili šuma stabala (u širem smislu), gdje unutarnji čvorovi stabala (eng. *nodes*) imaju ulogu operatora (ili funkcije određenog broja varijabli), a završni čvorovi stabala (listovi) ulogu operanada. Pritom su skup operatora F (eng. *function set*) i skup operanada T (eng. *terminal set*) **unaprijed definirani** konačni skupovi.

Na Slici 2.2 prikazano je stablo ekvivalentno jednostavnom matematičkom izrazu, a može predstavljati računalni program.

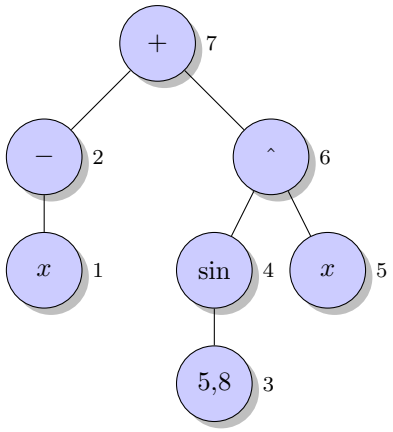


Slika 2.2. Stablasta predodžba izraza $-x + \sin^x 5{,}8$.

Evaluacija stabla obično se obavlja tzv. *postorder* obilaskom (eng. *tree traversal*) [5], kod kojeg se prvo evaluiraju podstabla (u pravilu slijeva nadesno), a zatim korijen (eng. *root*). Evaluacija je opisana u Pseudokodu 2.1, a na Slici 2.3 prikazan je redoslijed evaluacije čvorova stabla prikazanog na Slici 2.2.

Pseudokod 2.1. Evaluacija stabla *postorder* obilaskom.

```
PostorderEval(stablo)
početak
    ako stablo je prazno onda povratak
    za svako podstablo radi
        PostorderEval(podstablo)
    kraj
    evaluiraj
kraj
```



Slika 2.3. Redoslijed evaluacije čvorova.

U skladu s općom teorijom evolucijskih algoritama, ulogu **kromosoma** u evolucijskim procesima genetskog programiranja imaju **nelinearni** objekti — grafovi, odnosno stabla.

2.4 GP kao poseban oblik evolucijskog algoritma

Genetsko programiranje poseban je oblik evolucijskog algoritma, opisanog u Pseudokodu 2.2 [10].

Pseudokod 2.2. Opći oblik evolucijskog algoritma.

```
EvolucijskiAlgoritam
početak
    stvori početnu populaciju
    evaluiraj početnu populaciju
    dok uvjet završetka nije zadovoljen radi
        provedi selekciju
        primijeni genetske operatore
        evaluiraj populaciju
    kraj
kraj
```

Naime, genetsko programiranje, kao i ostali evolucijski algoritmi, ima sljedeće karakteristike:

1. stvaranje početne **populacije** rješenja (eng. *population*)
2. analizu svake **jedinke** (eng. *unit*) populacije i populacije općenito

3. proces selekcije uz odabir **genetskih operatora** koji obavljaju odgovarajuće akcije nad jedinkom ili jedinkama.

2.5 Populacija i jedinke

Biološka definicija opisuje populaciju skupom jedinki iste vrste koje žive na jednom staništu [26]. U genetskom programiranju jedinku predstavlja računalni program, a prostor egzistencije jedinki iste populacije je jedna iteracija optimizacijskog algoritma, odnosno jedna generacija evolucijskog procesa. Prema tome, često se govori o populaciji računalnih programa unutar jedne generacije evolucijskog procesa.

2.6 Dobrota jedinke i funkcija dobrote

Poznato je da se među biološkom vrstom na određenom staništu neke jedinke više ističu, imaju dulji životni vijek, veće mogućnosti za stvaranje potomstva i sl. Genetsko programiranje karakteriziraju ista svojstva. Naime, u cilju uspješnog rješavanja problema, vrlo je bitno da bolja rješenja ostaju u daljnjem razmatranju, a ona lošija otpadaju.

Kvalitetu rješenja vrednuje unaprijed definirana funkcija dobrote. Na temelju raznih parametara, ona određuje **dobrotu** (eng. *fitness*) pojedine jedinke, o kojoj kasnije uvelike ovisi sudbina iste. Dobrota je najčešće predstavljena realnim brojem, čiji se raspon vrijednosti normalizira na interval $[0, 1]$. Ovisno o problemu, moguća su preslikavanja dobrote i na intervale poput $[0, +\infty\rangle$ i $[-1, 1]$ ili se vrši njena diskretizacija.

Definiranje funkcije dobrote jedan je od ključnih problema genetskog programiranja. Potrebno je da bude što "bolja", a što "jednostavnija", jer je njeno evaluiranje prisutno u svakom koraku evolucijskog procesa. Stoga se odabir i definiranje funkcije dobrote **prilagođava** problemu i njegovim karakteristikama, a zadovoljenost navedenih oprečnih zahtjeva (kvaliteta i jednostavnost) pokušava se uravnotežiti.

2.7 Genetski operatori

Evolucijski aspekt genetskog programiranja očituje se u načinu optimiranja, gdje su u ulogama genetskih operatora prisutne metode **reprodukcije** (eng. *reproduction*), **križanja** (eng. *crossover*) i **mutacije** (eng. *mutation*).

U nekim slučajevima pojavljuju se i operatori permutacije (eng. *permutation*), zamjene (eng. *swap*), inverzije (eng. *inversion*) itd.

2.7.1 Reprodukcija

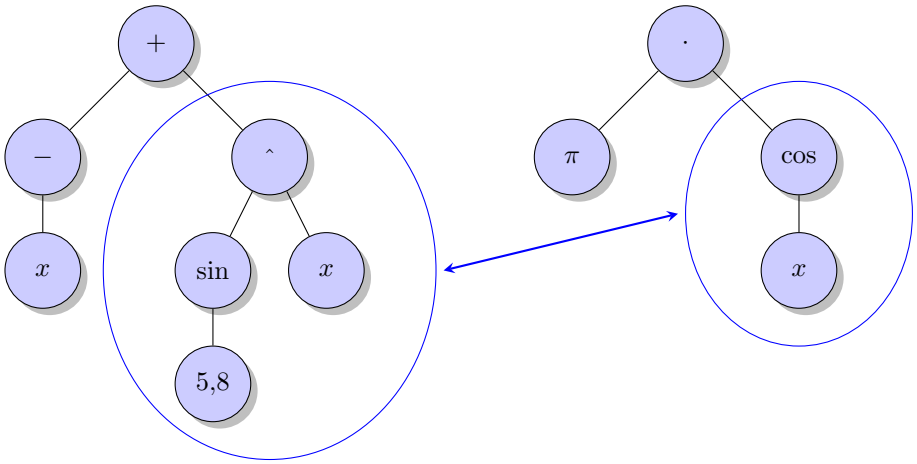
Reprodukcija je jednostavno kopiranje odabrane jedinke i umetanje iste u novu generaciju populacije. Semantika ovog genetskog operatora je **preživljavanje** odabrane jedinke, a očekivana posljedica je povećanje prosječne dobrote populacije.

Važni parametri koji utječu na reprodukciju su:

- vjerojatnost odabira reprodukcije kao genetskog operatora
- vjerojatnost odabira pojedine jedinke.

2.7.2 Križanje

Križanje je analogno biološkoj **spolnoj rekombinaciji** (Slika 2.4). U toj metodi dolazi do zamjene slučajno odabranih podstabala dviju odabranih jedinki, čime se svojstva stabala roditelja prenose na stabla djecu.



Slika 2.4. Križanje.

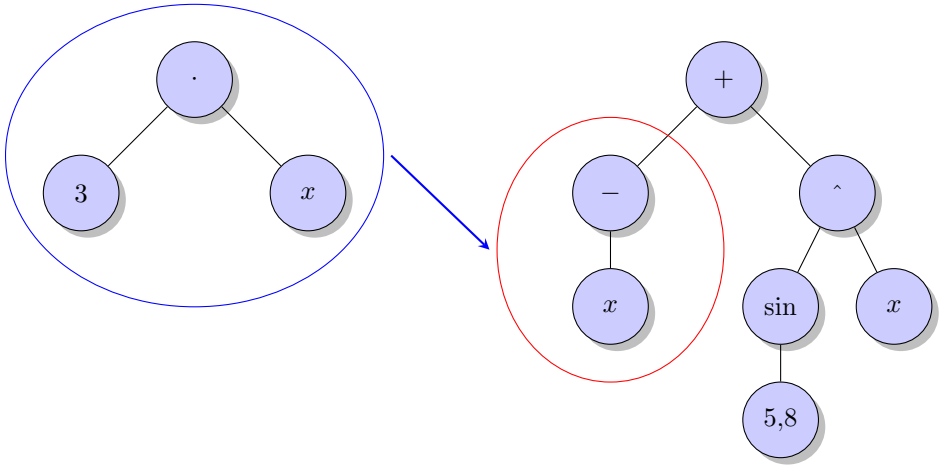
Važni parametri koji utječu na križanje su:

- vjerojatnost odabira križanja kao genetskog operatora
- vjerojatnost odabira pojedine jedinke
- vjerojatnost odabira pojedinih čvorova jedinki kao korijena podstabala.

2.7.3 Mutacija

Analogno biološkoj mutaciji, ovaj genetski operator pridonosi **poremećaju** "genskog materijala" najčešće slučajno odabrane ili drugim operatorima stvorene jedinke.

Konkretno, događa se promjena nekog dijela stabla jedinke (Slika 2.5).



Slika 2.5. Mutacija.

Uobičajene metode mutiranja [17] su:

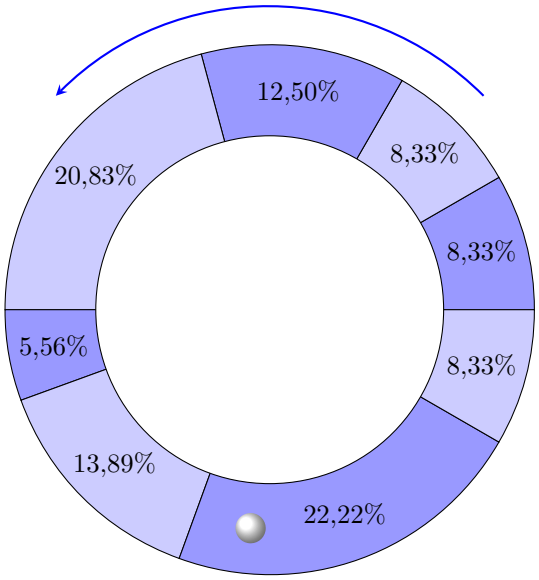
- zamjena slučajno odabranog podstabla slučajno generiranim stablom
- zamjena slučajno odabranog čvora čvorom istog tipa
- zamjena slučajno odabranog čvora komplementarnim čvorom
- zamjena slučajno odabranog podstabla završnim čvorom.

Važni parametri koji utječu na mutaciju su:

- vjerojatnost odabira mutacije kao genetskog operatora
- vjerojatnost odabira pojedine jedinke
- vjerojatnost odabira pojedinog čvora jedinke kao korijena podstabla.

2.8 Selekcija

U evolucijskom procesu odvija se **selekcija**, ovisna o vjerojatnosti odabira genetskih operatora i dobrotama jedinki. Uobičajena je **proporcionalna selekcija** (eng. *roulette wheel selection*), koja simulira igru ruleta u kojem se, zaustavljanjem kotača, kuglica smješta na jedno od polja. Polja odgovaraju jedinkama, a njihova veličina dobrotama jedinki (Slika 2.6), čime se postiže veća vjerojatnost odabira bolje jedinke.



Slika 2.6. Proporcionalna selekcija.

Vrlo su popularne i tzv. **turnirske** selekcije (eng. *tournament selections*), kod kojih se slučajno odabire k jedinki te se najgora, odnosno najgore jedinke zamjenjuju novim jedinkama nastalima primjenom genetskih operatora nad najboljima. Parametar k unaprijed je zadan (uobičajeno je $k = 3$ ili $k = 4$), stoga se često govori o k-turnirskoj selekciji. Turnirske selekcije imaju svojstvo **elitizma**, koje je karakterizirano činjenicom da najbolja jedinka sigurno preživljava selekciju u promatranoj iteraciji evolucijskog algoritma.

Kao jedna od boljih selekcija, u literaturi [15] navodi se i sljedeća:

1. populacija se podijeli na dvije podpopulacije
2. većina akcija (npr. 80%) određenog genetskog operatora provodi se nad jednom podpopulacijom.

2.9 Završetak evolucijskog procesa

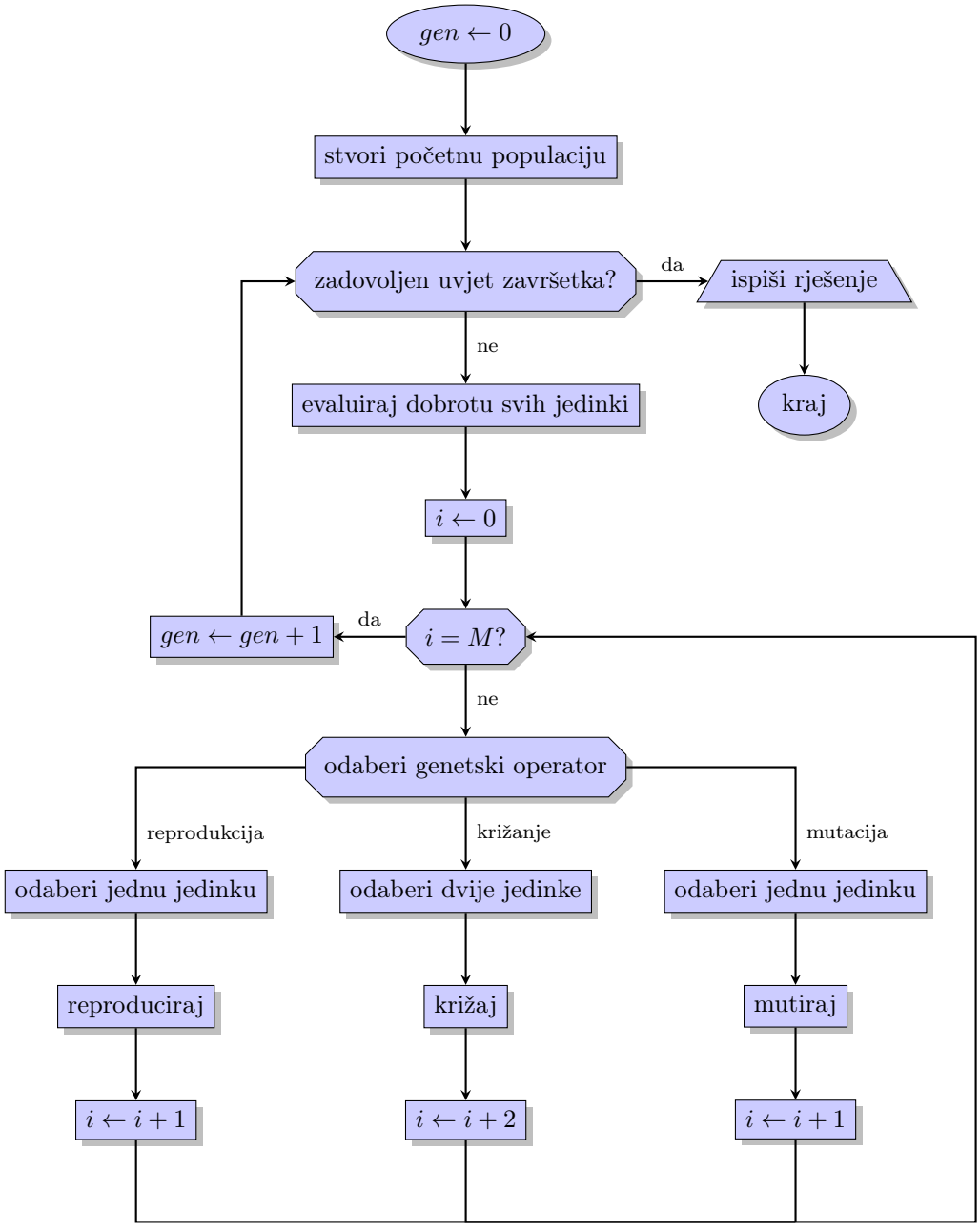
Evolucijski proces završava zadovoljavanjem određenog logičkog predikata ovisnog o dobroti. U obzir se dodatno uzimaju konstante G, M i $t_{\max}$ — broj generacija, broj jedinki i maksimalno vrijeme trajanja evolucijskog procesa.

Kada se jednom zadovolji uvjet završetka, iz populacije rješenja odabire se najkvalitetnije te se **semantički** opisuje, ako je ikako moguće. Poželjne su i daljnje analize, kao i ponavljanje cijelog postupka zbog nezadovoljstva dobivenim rješenjem. Bolje rješenje

uglavnom se pokušava pronaći redefiniranjem funkcije dobrote, redefiniranjem skupova operatora i operanada, povećanjem broja generacija itd.

2.10 Blok-dijagram genetskog programiranja

Uzevši u obzir sve navedeno, genetsko programiranje konačno se može prikazati blok-dijagramom na Slici 2.7.



Slika 2.7. Blok-dijagram genetskog programiranja.

Poglavlje 3

Problem prijanjanja proteina

"Everything should be made as simple as possible, but not simpler."

Albert Einstein[1]*, 1934.*

"Sve bi trebalo biti napravljeno što je moguće jednostavnije, ali ne i jednostavnije od toga."

3.1 Uvod

Ubrzanim razvojem istraživačke opreme, došlo je do povećanja učinkovitosti i preciznosti eksperimentalnih metoda u biologiji. Vrijeme izvođenja eksperimenata je smanjeno, što je rezultiralo naglim porastom broja novih spoznaja, ali i povećanjem njihove vjerodostojnosti. Usporednim razvojem računalnih sustava, prvenstveno računalnih mreža (Internet), stvoreni su uvjeti za **organizaciju** i **upravljanje** velikim brojem novih informacija.

Počeo je razvoj javno dostupnih baza podataka koje su obuhvaćale podatke vezane za pojedino područje istraživanja (npr. proteini, nukleinske kiseline). Prvotna ideja bila je sakupljanje podataka o što većem broju makromolekula, s naglaskom na njihov primarni slijed (npr. aminokiselina, nukleotida), budući da su njime jednoznačno određene. Osim same informacije o kompoziciji i slijedu određene makromolekule, dodatno su se unosile značajke kao što su: organizam iz kojeg potječe, funkcija, moguće interakcije i dr. Zbog heterogenosti tako unesenih podataka, bilo je nužno uspostaviti standarde unosa, a prvenstveno **format zapisa** makromolekulskih sljedova među kojima je i danas, zbog svoje jednostavnosti, najzastupljeniji FASTA. Na Slici 3.1 prikazan je zapis aminokiselinskog slijeda hemoglobina u FASTA formatu.

[1] Albert Einstein (1879. – 1955.), njemački teorijski fizičar i filozof

```
>swissprot|P68871|HBB_HUMAN Hemoglobin subunit beta;
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK
VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG
KEFTPPVQAAYQKVVAGVANALAHKYH
```

Slika 3.1. FASTA zapis hemoglobina.

Osim baza podataka koje su obuhvaćale unose primarnih sljedova makromolekula, razvijale su se baze koje su sadržavale zapise rezultata dobivenih različitim eksperimentalnim metodama (npr. strukturni prikaz proteina, spektar masa i dr.). Pojavila se potreba za njihovim međusobnim **povezivanjem** i **nadgradnjom**. Unakrsnim povezivanjem svih oblika informacija dostupnih u bazama podataka, stvoreni su uvjeti za njihovu lakšu manipulaciju. Pojavom novih i sve složenijih izazova (znanstvenih hipoteza), započeo je **razvoj alata** za lakšu obradu tako velike količine podataka.

Temelj takvih alata su algoritmi poput *Needleman–Wunsch* i *Smith–Waterman* algoritama za poravnanje, odnosno usporedbu sljedova. Oni olakšavaju izradu kompleksnih prikaza koji pružaju uvid u biološke sustave te omogućavaju bolje razumijevanje prirodnih zakona na kojima se osnivaju. Najzastupljeniji oblici takvih metoda su:

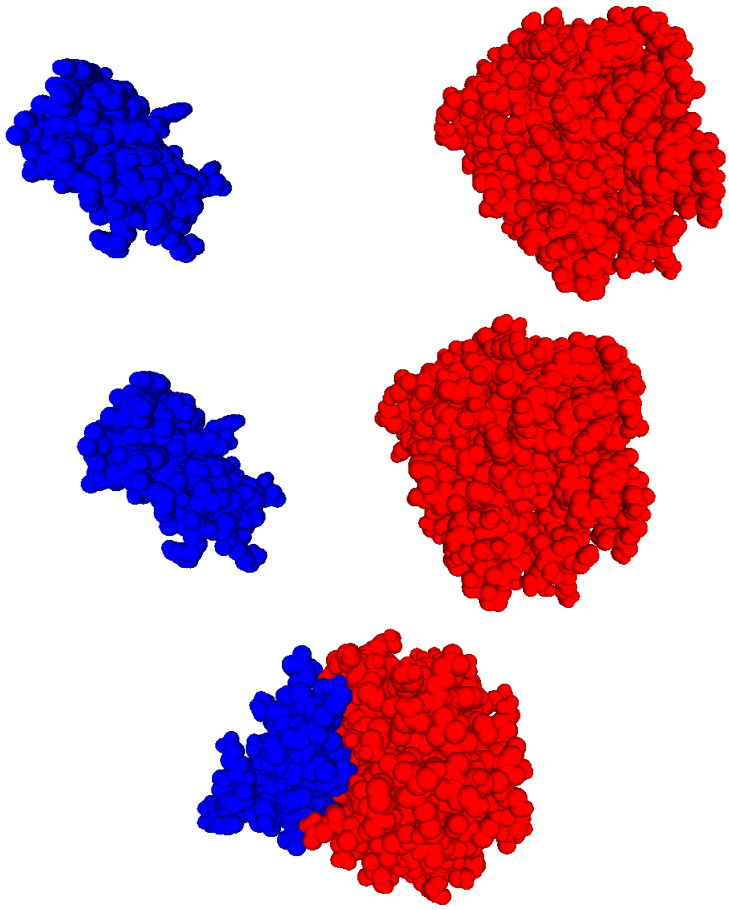
- izrada filogenetskih stabala
- uspostavljanje kontinuiranih shema metaboličkih putova
- grupiranje makromolekula s obzirom na slična svojstva
- simuliranje makromolekulskih interakcija i sl.

Jedan od glavnih izazova u istraživanju proteina je mogućnost određivanja struktura i svojstava poznavajući samo podatke o njihovom aminokiselinskom slijedu. Osim osnovnih ideja o razvoju općih zakona **smatanja proteina** (eng. *protein folding*), veliki cilj je i izrada simulacija koje što preciznije opisuju realne sustave. Time se može suziti izbor modela koji bi se uzeli u obzir za izvođenje dugotrajnih i složenih eksperimenata, ali i predvidjeti moguće promjene sustava u drukčijim uvjetima. Jedna od simulacijskih metoda koja pridonosi razumijevanju protein-proteinskih interakcija je **prijanjanje proteina**.

3.2 Prijanjanje proteina

Protein-proteinske interakcije predstavljaju jedan od osnovnih mehanizama koji se pojavljuje u gotovo svim biološkim procesima, stoga je unaprijeđenje računalnih metoda koje pridonose njihovom razumijevanju od velikog značaja.

Prijanjanje proteina (eng. *protein docking*) je računalno modeliranje složenih interakcija dvaju ili više proteina, a osnovni cilj mu je **predviđanje** kompleksnih trodimenzionalnih struktura koje se pritom formiraju. Na Slici 3.2 prikazano je jedno takvo formiranje.



Slika 3.2. Formiranje proteinskog kompleksa.

Prijanjanje proteina može se poistovjetiti s metodom optimiranja koja u nizu koraka (iteracija optimizacijskog algoritma) pokušava predvidjeti način formiranja proteinskog kompleksa. Pritom se u obzir uzimaju značajke pojedinog proteina, a kompleks se vrednuje poštivajući fizikalne i kemijske zakone. **Funkcije vrednovanja** (eng. *scoring functions*) mogu se razvrstati [28] u tri osnovne kategorije:

1. funkcije jakosti međumolekulskih i međuatomskih veza (eng. *force field functions*)
2. empirijske funkcije (eng. *empirical functions*)
3. funkcije temeljene na znanju (eng. *knowledge-based functions*).

3.3 Vrste prijanjanja

Osnovna podjela metoda prijanjanja uzima u obzir **čvrstoću** proteina te se mogu izdvojiti sljedeće dvije kategorije:

1. prijanjanje čvrstih proteina (eng. *rigid-body docking*)
2. prijanjanje savitljivih proteina (eng. *flexible-body docking*).

Prijanjanje čvrstih proteina pretpostavlja njihovu čvrstoću koja se očituje u nepromjenjivim geometrijskim značajkama veza između atoma (kutovi, duljine, rotacije i torzije). Takva pretpostavka zapravo je **restrikcija** problema prijanjanja te olakšava i ubrzava rješavanje problema, a pogotovo kada se primjenjuje na jednostavne sustave kod kojih fluktuacije struktura ne utječu na interakciju. S druge strane, restrikcija predstavlja ograničavajući čimbenik za točno predviđanje najpovoljnije interakcije ukoliko je fluktuacija struktura ključna za formiranje funkcionalnog kompleksa (npr. vezanje proteina u aktivno mjesto enzima).

3.4 Razine definiranja sustava

Ovisno o veličini, način definiranja sustava najvažniji je čimbenik u složenosti postavljanja problema prijanjanja, jer se detaljnijim opisom povećava broj mogućih interakcija između definiranih dijelova sustava.

Neke od osnovnih razina definiranja su:

- definiranje nizom aminokiselina
- definiranje nizom funkcionalnih skupina
- definiranje skupom atoma
- definiranje na kvantno-mehaničkoj razini.

Kako bi se povećala preciznost, dodatno se u obzir uzimaju fizikalna i kemijska svojstva.

Najčešći pristupi su definiranja **geometrijskim** i **elektrostatskim** značajkama, jer nude optimalan omjer složenosti prikaza i količine informacijskog sadržaja. Iako je elektrostatski pristup precizniji, još uvijek je dovoljno složen, stoga je geometrijski pristup prikladan u većini slučajeva.

3.5 Određivanje proteinske strukture

Za izradu simulacije interakcije dvaju proteina, nužno je poznavanje strukture pojedinog proteina, odnosno **prostornog rasporeda** atoma, a uobičajene metode određivanja su **rendgenska kristalografija** (eng. *X-ray crystallography*) i **NMR spektroskopija** (eng. *nuclear magnetic resonance spectroscopy*) [27].

Rendgenska kristalografija temelji se na ogibu snopa rendgenskih zraka o kristalnoj rešetki, a NMR spektroskopija koristi svojstvo rezoniranja jezgara atoma koji se nalaze unutar magnetskog polja.

3.6 Protein Data Bank

Protein Data Bank (skraćeno PDB) je repozitorij zapisa o trodimenzionalnim strukturama proteina. Podaci su eksperimentalnog karaktera i najčešće su rezultat metoda navedenih u Potpoglavlju 3.5.

S druge strane, kratica PDB je sinonim i za jedan od formata datotečnog zapisa trodimenzionalnih struktura proteina. Radi se o preglednom tekstualnom formatu, čije datoteke imaju ekstenziju `pdb`. Datoteke se raščlanjuju (parsiraju, eng. *parse*) prema vodećoj ključnoj riječi svakog retka, a sadrže informacije o funkciji proteina, taksonomsko obilježje, reference na radove, kao i ostale napomene.

Osnovni podaci zapisani u PDB datotekama su prostorni rasporedi atoma. Oni su navedeni u redovima označenima ključnim riječima ATOM i HETATM. Na Slici 3.3 prikazano je nekoliko takvih redova, a u njima su redom zapisani:

1. redni brojevi atoma
2. vrste i položaji atoma unutar aminokiselina
3. vrste aminokiselinskih ostataka
4. oznake odgovarajućih polipeptidnih lanaca
5. redni brojevi aminokiselinskih ostataka
6. x, y i z koordinate središta atoma (mjerna jedinica je Å — angstrom[2])
7. vjerojatnosti postojanosti
8. temperaturni koeficijenti promjenjivosti položaja atoma.

[2] $1\,\text{Å} = 0{,}1\,\text{nm}$

```
...
ATOM      1  N   ILE A  16      -8.155   9.648  20.365  1.00 10.68
ATOM      2  CA  ILE A  16      -8.150   8.766  19.179  1.00 10.68
ATOM      3  C   ILE A  16      -9.405   9.018  18.348  1.00 10.68
ATOM      4  O   ILE A  16     -10.533   8.888  18.870  1.00 10.68
ATOM      5  CB  ILE A  16      -8.091   7.261  19.602  1.00 10.68
ATOM      6  CG1 ILE A  16      -6.898   6.882  20.508  1.00  7.42
ATOM      7  CG2 ILE A  16      -8.178   6.281  18.408  1.00  7.42
ATOM      8  CD1 ILE A  16      -5.555   6.893  19.773  1.00  7.42
ATOM      9 1H   ILE A  16      -7.299   9.474  20.929  1.00 99.99
ATOM     10 2H   ILE A  16      -8.172  10.642  20.058  1.00 99.99
ATOM     11 3H   ILE A  16      -8.997   9.448  20.942  1.00 99.99
ATOM     12  N   VAL A  17      -9.224   9.305  17.090  1.00  9.63
ATOM     13  CA  VAL A  17     -10.351   9.448  16.157  1.00  9.63
ATOM     14  C   VAL A  17     -10.500   8.184  15.315  1.00  9.63
ATOM     15  O   VAL A  17      -9.496   7.688  14.748  1.00  9.63
...
```

Slika 3.3. Redovi PDB zapisa označeni ključnom rječju ATOM.

Iako standard zapisa propisuje i navođenje kemijskih simbola atoma i njihovih naboja, u velikom broju slučajeva oni nisu navedeni [29].

Često nedostaju i podaci o atomima vodika, s obzirom da rendgenska kristalografija ne uspijeva odrediti njihove položaje. Kako je njihov udio u građi proteina čak 50%, poželjne su **naknadne obrade** PDB zapisa temeljene na znanju [4].

Poglavlje 4

Programsko rješenje

"The question of whether machines can think... is about as relevant as the question of whether submarines can swim."

Edsger Wybe Dijkstra[1]*, 1984.*

"Raspravljanje o tome mogu li strojevi misliti, jednako je umjesno kao i raspravljanje o tome mogu li podmornice plivati."

4.1 Osnovne informacije o radnom okruženju ECF

ECF (eng. *Evolutionary Computation Framework*) je radno okruženje za rješavanje problema evolucijskog računanja koje se razvija na matičnom fakultetu pod vodstvom doc. dr. sc. Domagoja Jakobovića [11]. Ostvareno je u programskom jeziku C++, uz korištenje STL-a (eng. *Standard Template Library*) [20] i Boost C++ Libraryja [21].

Posljednja inačica pokriva metode genetskih algoritama i genetskog programiranja, a nudi i mogućnost paralelizacije korištenjem MPI-ja (eng. *Message Passing Interface*) [3]. Implementacija slijedi **objektno-orijentiranu** paradigmu i potpuno je parametrizirana.

Najvažniji moduli ECF-a u kontekstu genetskog programiranja su razredi `Tree`, `Algorithm` i `Registry`. Razredom `Tree`, koji nasljeđuje apstraktni razred `Genotype`, predstavljen je stablasti kromosom genetskog programiranja, razredom `Algorithm` predstavljeni su operatori i algoritmi selekcije, dok je razredom `Registry` obuhvaćena registracija i dohvat parametara evolucijskog procesa.

Parametri se zadaju u konfiguracijskoj datoteci XML formata (eng. *Extensible Markup Language*), a dohvaćaju se prilikom inicijalizacije ECF-a. Primjer konfiguracijske datoteke dan je na Slici 4.1.

[1] Edsger Wybe Dijkstra (1930. – 2002.), nizozemski računarski znanstvenik

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

  <Genotype>
    <Tree>
      <Entry key="treetype">action</Entry>
      <Entry key="maxdepth">5</Entry>
      <Entry key="mindepth">0</Entry>
      <Entry key="functionset">+ - sin cos * / act_ser_4 act_ser_3 act_ser_2 transl2_x
          rot1_x rot1_y rot1_z rot2_x rot2_yy rot2_zz</Entry>
      <Entry key="terminalset">DOUBLE [-10 10] no_act</Entry>
      <Entry key="crx.simple">1.00</Entry>
      <Entry key="crx.simple.functionprob">0.9</Entry>
      <Entry key="mut.hoist">0.25</Entry>
      <Entry key="mut.gauss">0.25</Entry>
      <Entry key="mut.nodecomplement">0.25</Entry>
      <Entry key="mut.nodereplace">0.25</Entry>
    </Tree>
  </Genotype>

  <Registry>
    <Entry key="randomizer.seed">0</Entry>
    <Entry key="population.size">80</Entry>
    <Entry key="population.demes">1</Entry>
    <Entry key="mutation.indprob">0.3</Entry>
    <Entry key="term.stagnation">20</Entry>
    <Entry key="term.maxgen">30</Entry>
    <Entry key="log.level">3</Entry>
    <Entry key="log.filename">log.txt</Entry>
    <Entry key="milestone.filename">out.txt</Entry>
    <Entry key="milestone.interval">0</Entry>
  </Registry>
</ECF>
```

Slika 4.1. Primjer konfiguracijske datoteke.

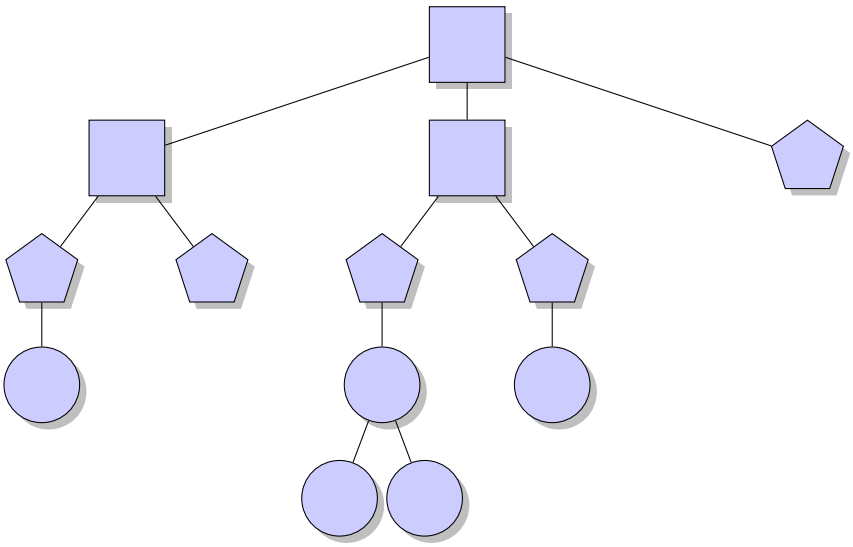
4.2 Pretpostavke programskog rješenja

Programsko rješenje rješava problem prijanjanja dvaju čvrstih proteina, koji su predstavljeni **skupovima idealiziranih atoma** u obliku kugli s Van der Waalsovim polumjerima. Proteinski kompleksi vrednuju se s obzirom na geometrijske značajke, tako da manje prostorno zauzeće ima veću dobrotu.

Na početku evolucijskog procesa proteini su udaljeni za aritmetičku sredinu polumjera R_1 i R_2 njihovih omeđujućih sfera, tako da je prvi protein postavljen u ishodište, a drugi protein na x-os trodimenzionalnog koordinatnog sustava, te se nizom geometrijskih transformacija pokušava formirati proteinski kompleks.

4.3 Prikaz jedinke

Jedinka predstavlja kompoziciju geometrijskih transformacija obaju proteina, a stablasti prikaz slijedi prototip prikazan na Slici 4.2.



Slika 4.2. Prototip jedinke.

Na početnim razinama stabla nalaze se čvorovi (označeni kvadratima) koji predstavljaju kompozicije geometrijskih transformacija — njihove djece (označene peterokutima). Podstabla koja slijede nakon transformacijskih čvorova predstavljaju matematičke izraze (označene krugovima) čijom se evaluacijom određuju argumenti transformacija (kutovi i pomaci).

Kompozicije imaju ulogu razgranavanja stabla, a implementirane su:

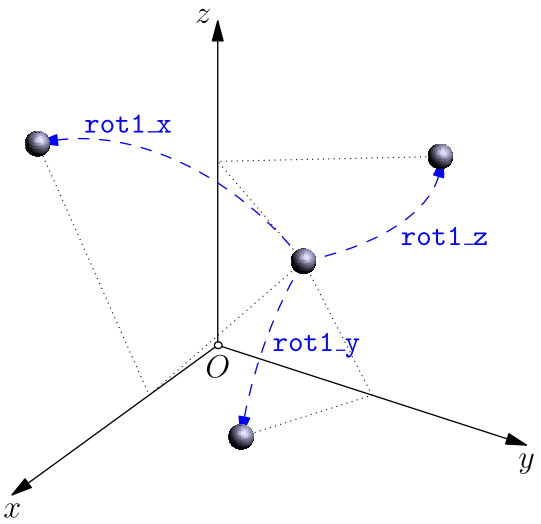
- kompozicija dviju transformacija — `act_ser_2`
- kompozicija triju transformacija — `act_ser_3`
- kompozicija četiriju transformacija — `act_ser_4`.

Implementirane geometrijske transformacije su:

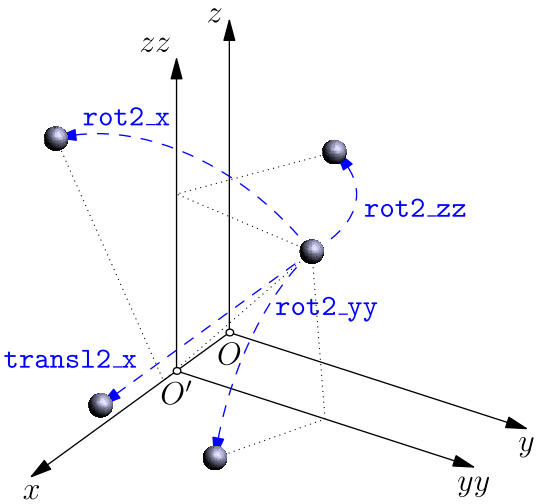
- rotacija prvog proteina oko x-osi — `rot1_x`
- rotacija prvog proteina oko y-osi — `rot1_y`
- rotacija prvog proteina oko z-osi — `rot1_z`
- rotacija drugog proteina oko x-osi — `rot2_x`

- rotacija drugog proteina oko yy-osi (y-osi lokalnog koordinatnog sustava drugog proteina) — `rot2_yy`
- rotacija drugog proteina oko zz-osi (z-osi lokalnog koordinatnog sustava drugog proteina) — `rot2_zz`
- translacija drugog proteina uzduž x-osi — `transl2_x`
- transformacija identiteta — `no_act`

a prikazane su na Slikama 4.3 i 4.4. Njima su pokriveni svi stupnjevi slobode proteina s ciljem pretraživanja skupa svih mogućih proteinskih kompleksa.



Slika 4.3. Transformacije atoma prvog proteina.



Slika 4.4. Transformacije atoma drugog proteina.

Matematičke izraze formiraju ranije implementirani operatori ECF-a: `+`, `-`, `*`, `/`, `sin`, `cos`, `pos`, `neg`, `max` i `min`.

4.4 Nadgradnja ECF-a

Implementacijom navedenih kompozicijskih i transformacijskih čvorova nadgrađeno je radno okruženje ECF. Svakom čvoru pridružena su svojstva `subtreeType_` i `argumentTypes_` s odgovarajućim vrijednostima — `Action` ili `Value`. Ostvareni su i kontrolni mehanizmi u izgradnji stabala, kako bi se postigla struktura prikazana na Slici 4.2. Registriran je novi parametar `treetype` kojim se određuje tip stabla.

Konačno, nadgrađen je genetski operator križanja `TreeCrxSimple` te genetski operatori mutacije: `TreeMutNodeReplace`, `TreeMutNodeComplement`, `TreeMutHoist` i `TreeMutGauss`, kako bi se njihovim utjecajem zadržala opisana stablasta struktura.

4.5 Vrednovanje proteinskih kompleksa

Kako je već navedeno, proteinski kompleksi se vrednuju s obzirom na geometrijske značajke tako da **manje prostorno zauzeće ima veću dobrotu**. Dodatno se moraju **kazniti preklapanja** atoma, jer su takve pojave fizički nemoguće. Najmanji dozvoljeni razmak površina atoma je 1,4 Å, što odgovara prostornom zauzeću vode, u kojoj se proteinske interakcije najčešće odvijaju.

U formiranim kompleksima može se uspostaviti sljedeća podjela atoma u disjunktne skupove:

1. S_1 — skup atoma prvog proteina koji se ne preklapaju s atomima drugog proteina
2. $\overline{S_1}$ — skup atoma prvog proteina koji se preklapaju s atomima drugog proteina
3. S_2 — skup atoma drugog proteina koji se ne preklapaju s atomima prvog proteina
4. $\overline{S_2}$ — skup atoma drugog proteina koji se preklapaju s atomima prvog proteina.

Oni su temelj definiranja funkcija dobrote opisanih u Potpoglavlju 4.6.

4.6 Funkcije dobrote

Implementirane su tri funkcije dobrote, koje nasljeđuju razred `EvaluateOp`:

1. `EvalOpCenter`
2. `EvalOpMinDistance`
3. `EvalOpPairwise`.

Funkcija `EvalOpCenter` vrednuje proteinski kompleks na temelju prosječne udaljenosti atoma drugog proteina od središta prvog proteina. Njen pseudokod dan je u Pseudokodu 4.1.

Pseudokod 4.1. Funkcija dobrote `EvalOpCenter`.

```
EvalOpCenter
početak
    dobrota ← 0
    za svaki atom a drugog proteina radi
        ako atom a se preklapa s nekim atomom prvog proteina
        onda
            dobrota ← dobrota + R_1 + R_2
        kraj
        inače
            dobrota ← dobrota + udaljenost atoma a od središta
            prvog proteina
        kraj
    kraj
    dobrota ← dobrota / (|S_2| + |S_2 (overline)|)
kraj
```

Funkcija `EvalOpMinDistance` vrednuje proteinski kompleks na temelju najmanje prosječne udaljenosti atoma prvog proteina od svih atoma drugog. Njen pseudokod dan je u Pseudokodu 4.2.

Pseudokod 4.2. Funkcija dobrote `EvalOpMinDistance`.

```
EvalOpMinDistance
početak
    dobrota ← 0
    za svaki atom a prvog proteina radi
        ako atom a se preklapa s nekim atomom drugog proteina
        onda
            dobrota ← dobrota + R_1 + R_2
        kraj
        inače
            dobrota ← dobrota + najmanja udaljenost atoma a od
            nekog atoma drugog proteina
        kraj
    kraj
    dobrota ← dobrota / (|S_1| + |S_1 (overline)|)
kraj
```

Funkcija `EvalOpPairwise` vrednuje proteinski kompleks na temelju prosječne udaljenosti atoma prvog i drugog proteina. Njen pseudokod dan je u Pseudokodu 4.3.

Pseudokod 4.3. Funkcija dobrote `EvalOpPairwise`.

```
EvalOpPairwise
početak
    dobrota ← 0
    za svaki atom a prvog proteina radi
        ako atom a se preklapa s nekim atomom drugog proteina
        onda
            dobrota ← dobrota + (|S2| + |S2'|) · (R1 + R2)
        kraj
        inače
            dobrota ← dobrota + zbroj svih udaljenosti atoma a od
            atoma drugog proteina
        kraj
    kraj
    dobrota ← dobrota / [(|S1| + |S1'|) · (|S2| + |S2'|)]
kraj
```

Opisane funkcije dobrote poprimaju realne vrijednosti u intervalu $[0, +\infty\rangle$, a manja vrijednost dobrote odgovara boljoj jedinki.

4.7 Ostali moduli programskog rješenja

4.7.1 Razred `Point`

Razredom `Point` predstavljena je točka trodimenzionalnog euklidskog prostora. Koordinate točke odgovaraju svojstvima `x`, `y` i `z` tipa `double`.

4.7.2 Razred `Vector`

Razredom `Vector` predstavljen je vektor trodimenzionalnog euklidskog prostora. Komponente vektora odgovaraju svojstvima `dx`, `dy` i `dz` tipa `double`.

4.7.3 Razred `Quaternion`

Razredom `Quaternion` predstavljen je kvaternion (eng. *quaternion*) — matematički objekt koji se može definirati uređenim parom skalara $s \in \mathbb{R}$ i trodimenzionalnog

vektora $\vec{v} \in \mathbb{R}^3$:

$$q := (s, \vec{v}), \tag{4.1}$$

a koristi se u implementaciji rotacije opisanoj u Dodatku A.

Skalarni član s odgovara svojstvu `s` tipa `double`, a vektorski član $\vec{v}$ svojstvu `v`, koje je instanca razreda `Vector`.

4.7.4 Razred `Atom`

Razredom `Atom` predstavljen je atom idealiziran kuglom. Središte atoma odgovara svojstvu `center`, koje je instanca razreda `Point`, a polumjer odgovara svojstvu `radius` tipa `double`. Simbol atoma predstavljen je svojstvom `symbol`, koje je instanca razreda `std::string`.

4.7.5 Razred `Protein`

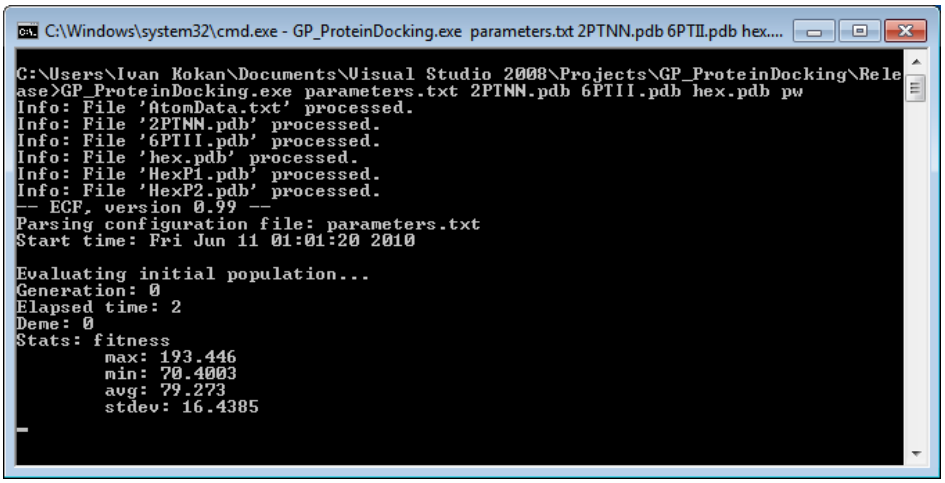
Razredom `Protein` predstavljen je protein kao skup atoma, kojem odgovara svojstvo `atoms` — instanca razreda `std::vector` nad razredom `Atom`. Svojstva `boundingSphereCenter` (instanca razreda `Point`) i `boundingSphereRadius` (tip `double`) odgovaraju središtu i polumjeru omeđujuće sfere proteina.

4.8 Pokretanje programskog rješenja

Programsko rješenje `GP_ProteinDocking` ostvareno je u razvojnom okruženju Microsoft Visual Studio 2008 u operacijskom sustavu Windows 7. Pokretanje se vrši iz konzole. Izvršnoj datoteci `GP_ProteinDocking.exe` redom se navode sljedeći argumenti:

1. konfiguracijska datoteka
2. PDB datoteka prvog proteina
3. PDB datoteka drugog proteina
4. PDB datoteka proteinskog kompleksa koja je rezultat alata Hex
5. izbor funkcije dobrote:
   a) `c` — `EvalOpCenter`
   b) `md` — `EvalOpMinDistance`
   c) `pw` — `EvalOpPairwise`.

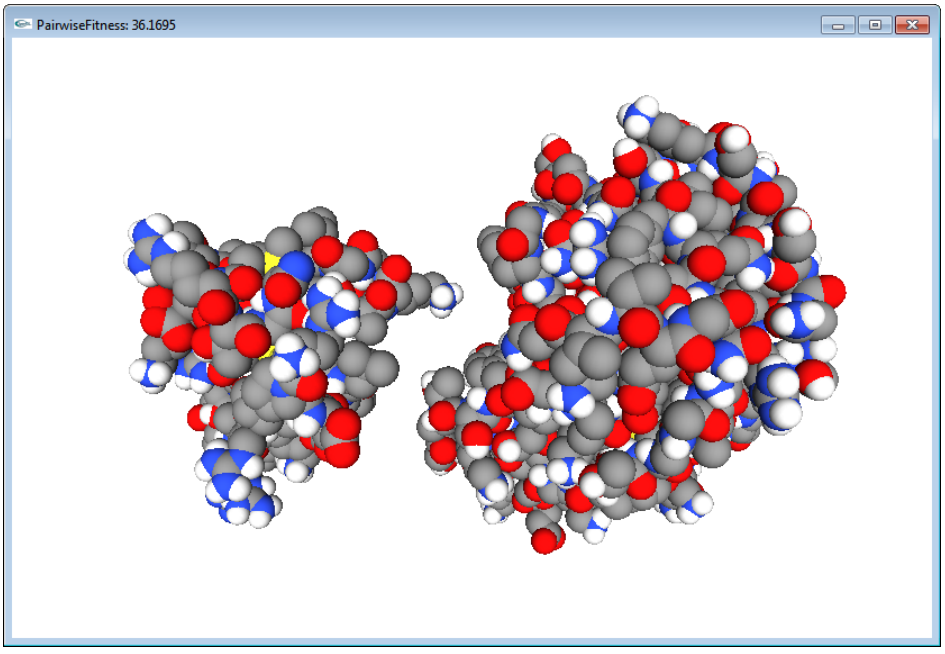
Ukoliko su putanje nekih datoteka krivo unesene ili je krivo odabrana funkcija dobrote, aplikacija ispisuje odgovarajuće poruke i završava s izvođenjem. U suprotnom, započinje evolucijski proces. Primjer pokretanja dan je na Slici 4.5.



Slika 4.5. Primjer pokretanja programskog rješenja.

4.9 Vizualizacija

Kako bi se omogućio uvid u rezultate prijanjanja proteina, implementiran je vizualizator koji koristi OpenGL (eng. *Open Graphics Library*) tehnologiju posredstvom GLUT-a (eng. *OpenGL Utility Toolkit*). Vizualizator nakon završetka evolucijskog procesa automatski prikazuje prvi pronađeni proteinski kompleks, a korisnik daljnjim manipulacijama može pregledati i ostale. Primjer vizualizacije dan je na Slici 4.6.



Slika 4.6. Vizualizacija proteinskog kompleksa.

Pretpostavljeno bojanje atoma slijedi prošireni CPK-model bojanja (u skladu s alatom Jmol [1]), a RGB-kodovi boja preuzeti su s [2] i nalaze se u tekstualnoj datoteci `AtomData.txt`. Ona se raščlanjuje prilikom pokretanja programskog rješenja. U istoj datoteci navedeni su i Van der Waalsovi polumjeri, preuzeti s [9].

Popis manipulacija vizualizatora dan je u Tablici 4.1.

Tablica 4.1. Manipulacije vizualizatorom.

tipka	**manipulacija**
`F1`	promjena načina bojanja atoma proteina u prošireni CPK-model
`F2`	promjena načina bojanja atoma proteina u plavo-crveni model
`F3`	promjena boje scenske pozadine u bijelu
`F4`	promjena boje scenske pozadine u crnu
`H`	promjena prikaza kompleksa (`GP_ProteinDocking` ili `Hex`)
`Enter`	prikaz boljeg kompleksa
`Backspace`	prikaz lošijeg kompleksa
+	povećanje detaljnosti prikaza
–	smanjenje detaljnosti prikaza
↑, ↓, ←, →	pomicanje kamere u sceni
lijeva tipka miša	usmjeravanje kamere u sceni
scroll tipka miša	približavanje i udaljavanje kamere u sceni
`Esc`	završetak izvođenja programa

Poglavlje 5

Ispitivanja i rezultati

"If we fail, we fall. If we succeed — then we will face the next task."

John Ronald Reuel Tolkien[1]*, iz trilogije "Gospodar prstenova"*

"Ako ne uspijemo, propadamo. Ako uspijemo — suočit ćemo se sa sljedećim zadatkom."

5.1 Uvod u ispitivanja

Kvaliteta ostvarenog programskog rješenja ispitana je kroz niz različitih slučajeva, sukladno činjenici kojom evolucijski proces genetskog programiranja ovisi o brojnim parametrima. Obavljena je i usporedba s rezultatima alata Hex, koji vrednuje **komplementarnost površina** proteina koji formiraju proteinski kompleks, a temelji se na rastavu površine proteina u red baznih funkcija (radijalne funkcije i sferni harmonici).

Osnovna ispitivanja obavljena su na paru proteina navedenom u Potpoglavlju B.1 te su neki rezultati izneseni i komentirani u Potpoglavljima 5.2, 5.3 i 5.4. Ispitana su i ostala dva para proteina navedena u Potpoglavljima B.2 i B.3 te su u Potpoglavlju 5.5 prikazani bolji formirani kompleksi.

Sva ispitivanja obavljena su na osobnom računalu s Intelovim Core2 Duo procesorom i 4 GB radne memorije. Ispitivana je 3-turnirska selekcija jedinki s najvećom dubinom stabla jednakom 5. Korištena su sva četiri operatora mutacije navedena u Potpoglavlju 4.4 s uniformnom raspodjelom vjerojatnosti odabira. Završni čvorovi stabala jedinki su slučajno odabrani realni brojevi iz intervala $[-10, 10]$. Za dobivanje rezultata prijanjanja alata Hex korištena je *web*-inačica istog [19] s pretpostavljenim parametrima optimiranja.

[1] John Ronald Reuel Tolkien (1892. – 1973.), engleski književnik

5.2 Ovisnost dobrote o broju generacija

Iz rezultata ispitivanja navedenih u Tablici 5.1 vidljivo je kako se povećanjem broja generacija povećava dobrota jedinki (dobrota formiranih kompleksa), a i smanjuje standardna devijacija dobrote, što svjedoči o konvergenciji rješenja. Može se zaključiti kako korištene funkcije dobrote nisu ekvivalentne kriteriju optimiranja kojeg koristi alat Hex.

Tablica 5.1. Ovisnost dobrote o broju generacija.

proteini	2ST1		3SSI	
broj atoma	2369		931	
veličina populacije	80			
vjerojatnost križanja	1,00			
vjerojatnost mutacije	0,15			
broj generacija	5	10	20	40
funkcija dobrote	EvalOpCenter			
prosječna dobrota	36,25923	34,85415	33,34309	33,30706
standardna devijacija dobrote	3,81565	1,82090	1,47343	0,89166
prosječno vrijeme trajanja jedne iteracije	6,12 s	5,46 s	5,06 s	4,99 s
dobrota rješenja alata Hex	44,36030			
funkcija dobrote	EvalOpMinDistance			
prosječna dobrota	24,02667	17,97817	14,78662	14,77138
standardna devijacija dobrote	13,47013	3,96154	1,10547	1,11757
prosječno vrijeme trajanja jedne iteracije	6,40 s	6,07 s	5,57 s	5,67 s
dobrota rješenja alata Hex	19,95391			
funkcija dobrote	EvalOpPairwise			
prosječna dobrota	32,70136	31,83497	30,49004	30,85346
standardna devijacija dobrote	1,24472	1,11796	1,24992	1,39853
prosječno vrijeme trajanja jedne iteracije	6,10 s	5,70 s	5,19 s	5,22 s
dobrota rješenja alata Hex	42,74599			

5.3 Ovisnost dobrote o veličini populacije

Iz rezultata ispitivanja navedenih u Tablici 5.2 vidljivo je kako veći broj jedinki populacije utječe na povećanje dobrote, a i smanjenje njene standardne devijacije, što svjedoči o konvergenciji rješenja.

Tablica 5.2. Ovisnost dobrote o veličini populacije.

proteini	2ST1		3SSI	
broj atoma	2369		931	
broj generacija	20			
vjerojatnost križanja	1,00			
vjerojatnost mutacije	0,15			
veličina populacije	10	20	40	80
funkcija dobrote	EvalOpCenter			
prosječna dobrota	55,72639	36,31310	34,02196	32,95507
standardna devijacija dobrote	24,69586	5,80128	1,68923	0,34865
prosječno vrijeme trajanja jedne iteracije	0,66 s	1,30 s	2,53 s	5,44 s
funkcija dobrote	EvalOpMinDistance			
prosječna dobrota	35,71174	15,56699	15,69584	15,58102
standardna devijacija dobrote	21,19042	1,39960	0,62237	1,21960
prosječno vrijeme trajanja jedne iteracije	0,72 s	1,41 s	3,00 s	5,52 s
funkcija dobrote	EvalOpPairwise			
prosječna dobrota	39,25254	32,78042	32,58466	31,11478
standardna devijacija dobrote	17,40219	3,31033	1,36653	1,29167
prosječno vrijeme trajanja jedne iteracije	0,68 s	1,34 s	2,52 s	5,44 s

5.4 Ovisnost dobrote o mutaciji

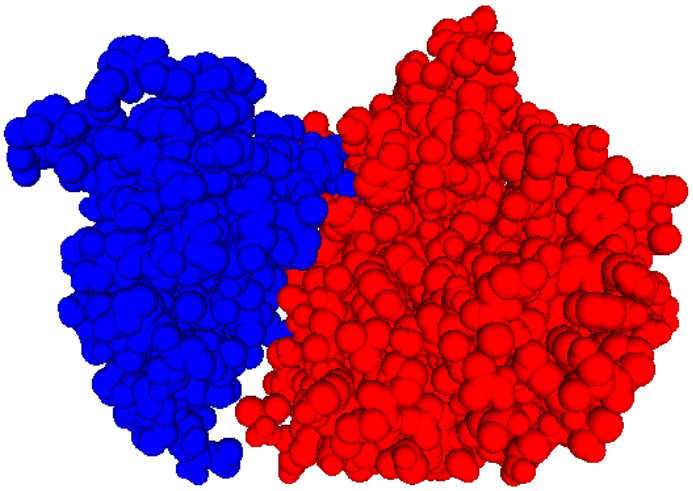
Iz rezultata ispitivanja navedenih u Tablici 5.3 vidljivo je kako ispitivane vjerojatnosti odabira genetskog operatora mutacije nemaju značajan utjecaj na dobrotu jedinki.

Tablica 5.3. Ovisnost dobrote o mutaciji.

proteini	2ST1		3SSI	
broj atoma	2369		931	
broj generacija	20			
veličina populacije	40			
vjerojatnost križanja	1,00			
vjerojatnost mutacije	0,00	0,01	0,05	0,10
funkcija dobrote	EvalOpCenter			
prosječna dobrota	32,29578	32,50487	32,88942	32,44546
standardna devijacija dobrote	1,68139	1,35982	1,59729	1,30832
prosječno vrijeme trajanja jedne iteracije	2,62 s	2,64 s	2,62 s	2,54 s
funkcija dobrote	EvalOpMinDistance			
prosječna dobrota	15,11215	15,13438	15,22122	15,11197
standardna devijacija dobrote	0,91855	0,84209	0,96182	0,67964
prosječno vrijeme trajanja jedne iteracije	2,78 s	2,84 s	2,86 s	2,87 s
funkcija dobrote	EvalOpPairwise			
prosječna dobrota	32,14819	32,42943	32,00234	32,31954
standardna devijacija dobrote	1,42914	0,97921	1,23056	0,71124
prosječno vrijeme trajanja jedne iteracije	2,59 s	2,63 s	2,62 s	2,58 s

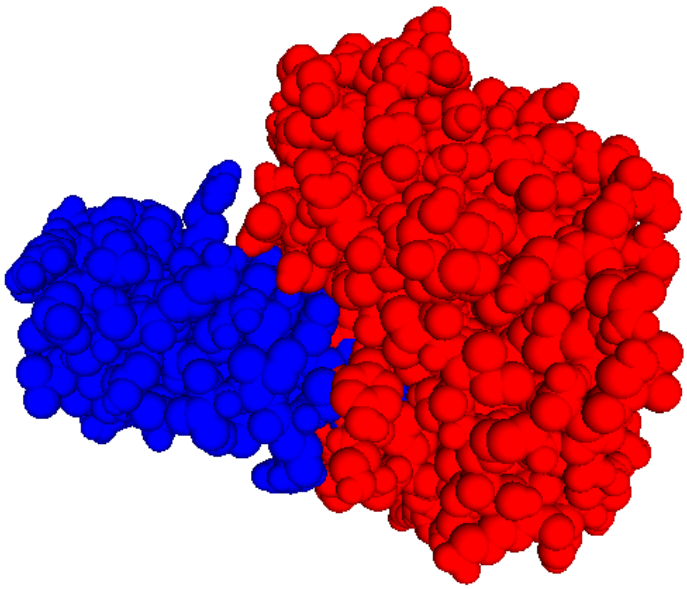
5.5 Bolji formirani kompleksi ispitnih parova

Na Slici 5.1 prikazan je jedan od boljih formiranih kompleksa subtilizina i *streptomyces* inhibitora.



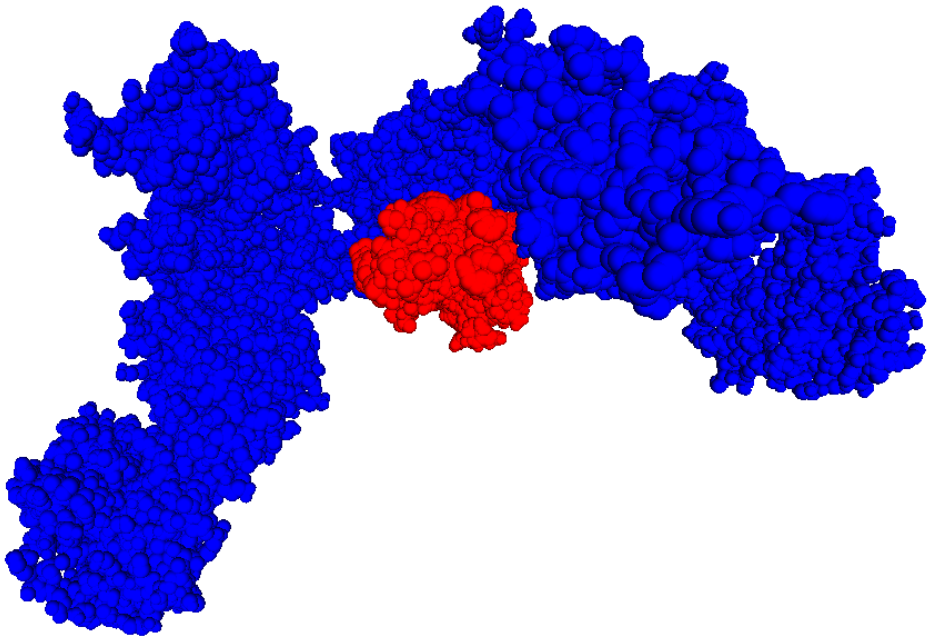
Slika 5.1. Kompleks subtilizina i *streptomyces* inhibitora.

Na Slici 5.2 prikazan je jedan od boljih formiranih kompleksa tripsina i njegovog inhibitora.



Slika 5.2. Kompleks tripsina i njegovog inhibitora.

Na Slici 5.3 prikazan je jedan od boljih formiranih kompleksa inzulina i inzulinskog receptora.



Slika 5.3. Kompleks inzulina i inzulinskog receptora.

Poglavlje 6

Zaključak

Genetsko programiranje jedna je od metoda evolucijskog računanja, a njena posebnost je optimiranje računalnih programa. Iako je pravu vrijednost dosegla tek krajem prošlog stoljeća (računalnom revolucijom), u vrlo kratkom vremenskom periodu je ostvarila zapažene rezultate u rješavanju različitih inženjerskih problema.

Problem prijanjanja proteina je simulacijska metoda koja pridonosi razumijevanju protein-proteinskih interakcija, a može se poistovjetiti s metodom optimiranja u formiranju proteinskih kompleksa u ovisnosti o brojnim parametrima. Sukladno toj činjenici ostvareno je programsko rješenje `GP_ProteinDocking`.

Rezultati obavljenih ispitivanja pokazali su kako se evolucijski proces genetskog programiranja vrlo brzo prilagođava problemu (kroz nekoliko desetaka generacija). Vizualizacijom rješenja omogućen je uvid u formiranje proteinskih kompleksa, koji su u skladu s kriterijima vrednovanja triju implementiranih funkcija dobrote.

Ivan Kokan

Dodatak A

Rotacija korištenjem kvaterniona

A.1 Osnovna razmatranja

Rotacija čvrstog tijela oko zadanog pravca u trodimenzionalnom euklidskom prostoru svodi se na rotaciju svih njegovih točaka. Pravac l jednoznačno je određen točkom $T(x_T, y_T, z_T)$ i vektorom smjera $\vec{v} = (a, b, c)$, a njegova jednadžba u kanonskom obliku glasi

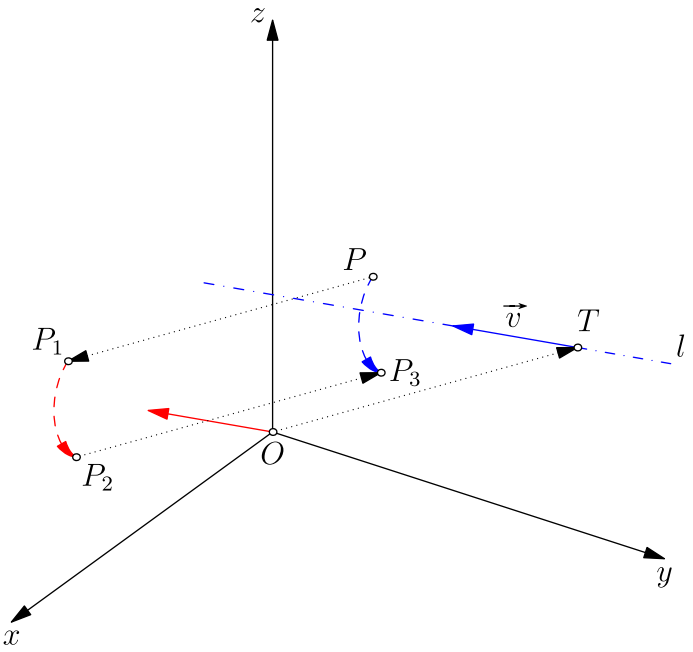
$$l \ldots \frac{x - x_T}{a} = \frac{y - y_T}{b} = \frac{z - z_T}{c}. \tag{A.1}$$

Bez smanjenja općenitosti može se pretpostaviti da je $\vec{v}$ jedinični vektor. Ako je točka O ishodište sustava, rotacija proizvoljne točke $P(x_P, y_P, z_P)$ oko pravca l za kut φ može se ostvariti kompozicijom sljedećih transformacija:

1. translacija za vektor $\overrightarrow{TO}$ $(P \mapsto P_1)$
2. rotacija za kut φ oko vektora $\vec{v}$ učvršćenog u ishodištu $(P_1 \mapsto P_2)$
3. translacija za vektor $\overrightarrow{OT}$ $(P_2 \mapsto P_3)$.

Smjer rotacije ovisi o orijentaciji vektora $\vec{v}$, sukladno pravilu desne ruke.

Kompozicija je prikazana na Slici A.1, a rotacija pod 2 opisana je u Potpoglavlju A.2.



Slika A.1. Rotacija točke P oko pravca l ostvarena kompozicijom transformacija.

A.2 Rotacija oko vektora učvršćenog u ishodištu

Radijus-vektor točke P_2 odgovara vektorskom članu kvaterniona

$$p_2 = \left(\cos\frac{\varphi}{2}, \sin\frac{\varphi}{2} \cdot \vec{v}\right) \times (0, \vec{p_1}) \times \left(\cos\frac{\varphi}{2}, -\sin\frac{\varphi}{2} \cdot \vec{v}\right), \tag{A.2}$$

pri čemu je $\vec{p_1}$ radijus-vektor točke P_1 [8].

Operacija kvaternionskog množenja kvaterniona $q_1 = (s_1, \vec{v_1})$ i $q_2 = (s_2, \vec{v_2})$ definirana je na sljedeći način:

$$q_1 \times q_2 := (s_1 s_2 - \vec{v_1} \cdot \vec{v_2}, s_1\vec{v_2} + s_2\vec{v_1} + \vec{v_1} \times \vec{v_2}), \tag{A.3}$$

gdje $\vec{v_1} \cdot \vec{v_2}$ predstavlja skalarni, a $\vec{v_1} \times \vec{v_2}$ vektorski umnožak vektora $\vec{v_1}$ i $\vec{v_2}$.

Dodatak B

Ispitni parovi proteina

B.1 Subtilizin i streptomyces inhibitor

Subtilizin (2ST1) je enzim iz skupine serinskih proteaza i djeluje nespecifično (na sve proteine) kidajući polipeptidne lance na manje fragmente, a *streptomyces* subtilizin inhibitor (3SSI) sprječava njegovu aktivnost.

B.2 Tripsin i BPTI

Tripsin (2PTN) je hidrolaza (enzim koji vrši digestiju proteina u reakciji s prisustvom vode) i spada u skupinu serinskih proteaza, a 6PTI je njegov inhibitor dobiven iz gušterače goveda (BPTI).

B.3 Inzulin i inzulinski receptor

Inzulin (2KJU) je hormon koji kontrolira metabolizam glukoze, a inzulinski receptor (3LOH, struktura ektodomene) je molekula posrednik za prijenos signala u stanicu. Nakon vezanja na receptor, u stanici se pokreće niz procesa kojima se kontrolira razina glukoze.

Literatura

[1] *Jmol: an open-source Java viewer for chemical structures in 3D.* Dostupno na Internet adresi http://www.jmol.org/.

[2] *Jmol Colors.* Dostupno na Internet adresi http://jmol.sourceforge.net/jscolors/.

[3] *The Message Passing Interface (MPI) standard.* Dostupno na Internet adresi http://www.mcs.anl.gov/research/projects/mpi/.

[4] John Badger: *Hydrogens in PDB files.* Dostupno na Internet adresi http://www.umass.edu/microbio/chime/pe2.79/pe/protexpl/help_hyd.htm.

[5] Paul E. Black: *Postorder traversal.* Dictionary of Algorithms and Data Structures. Dostupno na Internet adresi http://www.itl.nist.gov/div897/sqg/dads/HTML/postorderTraversal.html, 2008.

[6] Nichael Lynn Cramer: *A Representation for the Adaptive Generation of Simple Sequential Programs.* Dostupno na Internet adresi http://homepages.sover.net/~nichael/nlc-publications/icga85/index.html, srpanj 1985.

[7] Luka Donđivić i Ivan Kokan: *Evolucijski algoritmi — Demonstracija rada evolucijskih algoritama: Genetsko programiranje.* Projekt, siječanj 2008.

[8] Laura Downs: *Using Quaternions to Represent Rotation.* Dostupno na Internet adresi http://www.genesis3d.com/~kdtop/Quaternions-UsingToRepresentRotation.htm#RepresentingRotations, 2002.

[9] Theodore Gray: *Van der Waals Radius for all the elements in the Periodic Table.* Dostupno na Internet adresi http://www.periodictable.com/Properties/A/VanDerWaalsRadius.an.html, 2010.

[10] Bill Hart: *Evolutionary Algorithms.* Dostupno na Internet adresi http://www.cs.sandia.gov/opt/survey/ea.html, 1997.

[11] Domagoj Jakobović: *ECF — Evolutionary Computation Framework.* Dostupno na Internet adresi http://gp.zemris.fer.hr/ecf/, 2010.

[12] Ivan Kokan: *Primjena genetskog programiranja u postupku simboličkog integriranja.* Završni rad br. 317, lipanj 2008.

[13] Ivan Kokan: *Implementacija genetskog programiranja u okviru ECF-a.* Seminar, svibanj 2009.

[14] Ivan Kokan: *Operatori mutacije ECF-a u kontekstu genetskog programiranja.* Projekt, siječanj 2010.

[15] John Reed Koza: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* A Bradford Book — The MIT Press, Cambridge, Massachusetts, 1992.

[16] John Reed Koza: *Patents Related to Genetic Programming.* Dostupno na Internet adresi `http://www.genetic-programming.com/patents.html`, 2004.

[17] Riccardo Poli, William B. Langdon i Nicholas F. McPhee: *A Field Guide To Genetic Programming*. Ožujak 2008. Dostupno na Internet adresi `http://www.gp-field-guide.org.uk/`.

[18] Dave Ritchie: *Hex Protein Docking.* Dostupno na Internet adresi `http://www.loria.fr/~ritchied/hex/`, 2010.

[19] Dave Ritchie: *Hex Server.* Dostupno na Internet adresi `http://www.loria.fr/~ritchied/hex_server/`, 2010.

[20] SGI: *Standard Template Library Programmer's Guide*. Dostupno na Internet adresi `http://www.sgi.com/tech/stl/`, 2010.

[21] Boost C++ Libraries Development Team: *Boost C++ Libraries.* Dostupno na Internet adresi `http://www.boost.org/`, 2010.

[22] Wikipedia: *Artificial intelligence.* Dostupno na Internet adresi `http://en.wikipedia.org/wiki/Artificial_intelligence`, 2010.

[23] Wikipedia: *Evolutionary algorithm.* Dostupno na Internet adresi `http://en.wikipedia.org/wiki/Evolutionary_algorithm`, 2010.

[24] Wikipedia: *Evolutionary computation.* Dostupno na Internet adresi `http://en.wikipedia.org/wiki/Evolutionary_computation`, 2010.

[25] Wikipedia: *Genetic programming.* Dostupno na Internet adresi `http://en.wikipedia.org/wiki/Genetic_programming`, 2010.

[26] Wikipedia: *Population.* Dostupno na Internet adresi `http://en.wikipedia.org/wiki/Population`, 2010.

[27] Wikipedia: *Protein Data Bank*. Dostupno na Internet adresi http://en.wikipedia.org/wiki/Protein_Data_Bank#Contents, 2010.

[28] Wikipedia: *Scoring functions for docking*. Dostupno na Internet adresi http://en.wikipedia.org/wiki/Scoring_Functions_for_Docking#Classes, 2010.

[29] wwPDB: *Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description*. Dostupno na Internet adresi ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v32_letter.pdf, 2008.