SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Metode optimizacije i nadogradnje postojećeg sustava

Ivan Sović

Voditelj: *prof.dr.sc. Branko Jeren*

Zagreb, travanj, 2009.

Sadržaj

1. Uvod

Protein Docking Tool (PDT) je aplikacija namijenjena korištenju u bioinformatičke svrhe. Ona omogućuje predviđanje struktura proteinskih kompleksa na temelju poznatih struktura proteina koji ih tvore. U okviru ovog seminara biti će nastavljen rad na već postojećem projektu. Kako *PDT* nije dovršen, bit će potrebno ispraviti već postojeće greške te popraviti nedostatke cijelog sustava. Osim toga, potrebno je i redizajnirati veće dijelove koda, kako bi sustav bio stabilniji, te zadovoljio postavljena očekivanja. Nakon ispravljanja grešaka, PDT će se povezati sa programom za vizualizaciju molekula, kako bi se mogli promatrati rezultati dobiveni izvršavanjem procesa prianjanja proteina.

Naposlijetku (a također i u međukoracima izrade seminara), biti će potrebno provoditi razne testove, kako bi se moglo vidjeti da li sustav radi na odgovarajući način. Za testiranje se koriste namještene «umjetne» strukture proteina, izlazni rezultati procesa prianjanja, te vizualizacija. Bitniji postupci testiranja su objašnjenji u ovome radu.

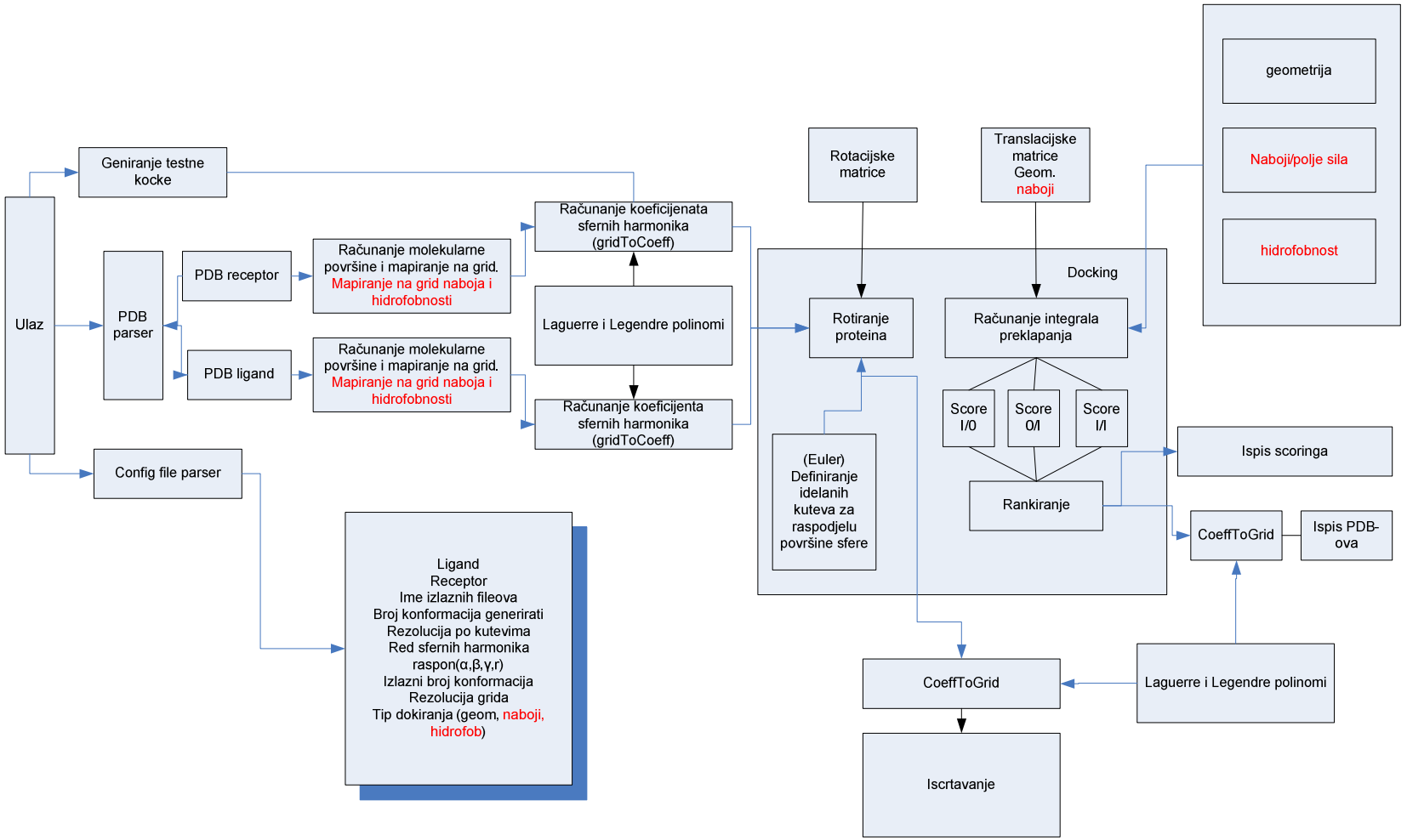
2. Opis postojećeg sustava i njegovog trenutnog stanja

2.1 Opis sustava

Protein Docking Tool sustav izvršava proces prianjanja proteina. Ulazni podatci su dvije *PDB* datoteke (eng. *Protein Data Bank*), te željene postavke procesa prianjanja. U zadanim PDB datotekama sadržane su strukture proteina. PDT prvo proslijedi ulazne proteine kroz PDB parser, koji izlučuje koordinate atoma, te njihove radijuse. Pomoću dobivenih podataka uzorkuje se površina svakog proteina, te se one prilagođuju prikazu u koordinatnoj mreži (eng. *grid*). Površine se zatim podvrgavaju transformaciji pomoću kuglinih funkcija, čime se bitno ubrzava i olakšava postupak rotiranja i translatiranja 3D oblika u prostoru. Završetkom svih ovih postupaka, započinje sam postupak prianjanja proteina. Translatiranjem i rotiranjem proteina, ispituje se komplementarnost njihovih površina za svaki od kuteva i radijusa (uz određenu zadanu razlučivost pomaka). Ispitivane se konformacije ocjenjuju, te na temelju tih ocjena rangiraju. Za svaki od radijusa uzima se konformacija sa najvećom ocjenom, te se od njih izgradi nova rang lista. Pomoću nove rang liste, određujemo koordinate atoma oba ulazna proteina, te se takvi zapisuju novu PDB datoteku. Izlazna datoteka sadrži sve konformacije sa najvećim ocjenama, kako bi se mogle pregledati u alatu za vizualizaciju.

U sklopu sustava, ali kao zasebna cjelina, realiziran je i alat za vizualizaciju molekula. Njegova je svrha uvid u strukturu molekula, i to prije, u procesu te nakon procesa prianjanja proteina.

Na slici 1 nalazi se shematski prikaz postojećeg sustava.



Slika 1. Inicijalni dizajn sustava.

2.2 Trenutno stanje sustava i njegovi nedostatci

Prilikom proučavanja i testiranja ovog sustava, primjećene su neke njegove mane i nedostatci. Sustav je postajao nestabilan prilikom nekih testova, te bi se neočekivano prekinulo izvođenje procesa prianjanja proteina. Sama komponenta za prilagođavanje ulaznih proteina nekoj od željenih konformacija te njezin upis u izlaznu PDB datoteku, nikada nije bila testirana, te su (očekivano) uočena odstupanja od pretpostavljenih rezultata. Alat za vizualizaciju je bio dostupan samo za *Windows* platformu, a osim toga je i postajao nestabilan prilikom učitavanja i prikazivanja većih PDB datoteka (kao što su upravo PDT-ove izlazne datoteke).

U ovom seminaru, glavnina pažnje će biti posvećena rješavanju sljedećih problema:

- sređivanje koda radi preglednosti,
- popravljanje stabilnosti sustava,
- testiranje pojedinih dijelova sustava, kako bi se uklonilo čim više grešaka,
- zamjena funkcija sa istovjetnim, ali dobavljenim iz gotovih programskih paketa (eng. *library*) sa ciljem rješavanja mogućih grešaka, te ubrzavanjem cijelog sustava,
- popravljanje stabilnosti vizualizatora molekula,
- prebacivanje (eng. *porting*) vizualizatora na *Linux* platformu,
- izrada jednostavnog programa za generiranje testnih proteina, te izrada proteina jasno vidljivih komplementarnih struktura,
- pronalaženje i ispravljanje grešaka u procesu prianjanja korištenjem testnih proteina,
- povezivanje sustava u cjelinu.

3. Metode optimizacije i nadogradnje sustava PDT

3.1 Uređivanje koda

PDT je sustav napravljen timskim radom više autora. On se, kao takav, sastoji od dijelova koji su rađeni odvojeno, te naknadno spojeni u jednu cjelinu. Zbog toga se stilovi kodiranja svakog od autora razlikuju (ako se nije pridržavalo dogovorenih normi). Prije bilo kakvih radova na sustavu, bilo je potrebno proučiti kod, te uskladiti cijeli sustav radi lakšeg snalaženja i razumijevanja njegovih dijelova. Osim toga, neki dijelovi sustava bili su povezani na način neprilagođen ostatku PDT-a, što bi uzrokovalo nemogućnost kompajliranja (eng. *compiling*) u slučaju većih izmjena koda, ili dodavanja novih mogućnosti sustavu. Također, nakon samog sređivanja koda, bilo je potrebno omogućiti jednostavnu izgradnju sustava (eng. *build*), tj. njegovo kompajliranje. U tu svrhu je izrađeno nekoliko skripti koje rade upravo taj posao. Tek tada se moglo početi sa popravljanjem sustava, nadogradnjom, te njegovom optimizacijom.

3.2 Popravljanje stabilnosti sustava

Pokretanje sustava s određenim testnim podatcima uzrokovalo je prekid izvođenja procesa prianjanja proteina. Kako sustav ne smije biti nestabilan za bilo kakve ulazne podatke, problem je bilo potrebno riješiti. Ispravljanje greške otežavala je činjenica da je jedina poruka koju bi operativni sustav vratio korisniku bila «*Killed.*». Navedena poruka upućuje da je greška vjerojatno vezana uz raspolaganje dostupnom memorijom, ali ne znamo gdje i kako.

Ispravljanje programske greške (eng. *debugging*, u nastavku *debugiranje*) izvedeno je «binarnim» pretraživanjem. Drugim riječima, dijelovi koda bi se uključivali/isključivali iz sustava, te ispitivalo postojanje tražene greške. Cijeli proces debugiranja bio je dosta dugotrajan, i to iz razloga što bi se postojanje greške očitovalo tek nakon dužeg vremena izvođenja programa (približno dvadesetak minuta). Greška je konačno otkrivena u jednoj od funkcija, te ispravljena. Ispostavilo se da je problem u funkciji koja računa faktorijele višestrukom preciznošću, a problem je riješen tako da je kod prilagođen u oblik koji koristi samo dvostruku preciznost (eng. *double*). O zamjeni višestruke preciznosti u double biti će još riječi u nastavku rada.

Ispravljanjem ove greške nije uočena niti jedna dodatna nestabilnost sustava, koja bi uzrokovala prekidanje izvođenja programa

3.3 Upotreba programskih biblioteka

U izvornom projektu PDT, za računanje raznih koeficijenata rotacijskih matrica i polinoma, korištena je višestruka preciznost brojeva. Kako C++ sam po sebi podržava najviše dvostruku preciznost (*double*), u projekt je uključena i biblioteka koja omogućuje korisniku baratanje praktički neograničeno velikim brojevima (jedino ograničenje je količina memorije) – *GNU Multiple Precision Library* (*GMP*). U sklopu ovog projekta odlučeno je izmijeniti PDT tako da više ne koristi *GMP*, već samo dvostruku preciznost. Razlog tome je već spomenuti problem sa memorijom koji se događao (koji je možda bio uzrokovan i načinom kodiranja, a ne samom bibiliotekom), a osim toga i sama brzina izvođenja koda. Dodatni razlog je neispravnost biblioteke ako se pri kompiliranju koristi krivi kompajler (eng. *compiler*), jer, prema uputama na web stranicama GMP-a, ova bibilioteka nije kompatibilna sa svim kompilerima. [5] Samo korištenje višestruke preciznosti pretpostavlja korištenje funkcija koje implementiraju potrebne operacije, a čim su veći podatci koje obrađuju, time i samo izvođenje funkcija traje duže. S druge strane, tipovi varijabli definirani u *C++*-u. Problem koji se može javiti je da pokušamo u varijablu tipa *double* zapisati jako velik broj. Zbog samog načina interpretacije zapisa realnih brojeva (mantisa i eksponent), i u *double* varijabli možemo koristiti vrlo velike brojeve, ali uz gubitke informacija. Broj će biti zaokružen na svega 52 decimale (normirano), što možda neće biti dovoljno da informacija ostane u potpunosti očuvana. Zbog toga se rezultati procesa prianjanja proteina (tzv. *score*-ovi) razlikuju od onih dobivenih originalnim sustavom. Bitno je još napomenuti da su konformacije, koje su u originalnom slučaju davale najveće rezultate, također i sada na istim položajima u rang listi rezultata.

Druga stvar vezana uz biblioteke koju je trebalo ostvariti, je implementacija matematičkih funkcija. PDT je već sadržavao svoju verziju matematičkih metoda, ali one su zamijenjene već postojećima u *GNU Scientific Library* (*GSL*) biblioteci. Razlog tome je mogućnost postojanja grešaka u PDT kodu (a koje još nisu bile uočene), te brzina izvođenja programa. Osim toga, čak ako GSL inačica matematičkih funkcija i sadrži grešku, ona će prije ili kasnije biti ispravljena, a sve što korisnik ovog sustava treba napraviti je osvježiti biblioteku na svom računalu.

3.4 Ubrzavanje rada sustava

Daljnjim testiranjem sustava uočeno je da se vrijeme izvođenja programa znatno produljuje korištenjem nešto većeg reda polinoma u funkcijama za aproksimaciju površine rastavom u red sfernih harmonika, što znači da je potrebno optimizirati kod. Kako se sam proces prianjanja proteina odvija odvojeno i naknadno od procesa koji priprema podatke za obradu (tj. postavljanje proteina na *grid*), ta dva dijela PDT-a možemo proučavati odvojeno.

Pogledajmo što se zapravo događa u dijelu sustava za učitavanje i obradu podataka. Analizom koda utvrđeno je da je složenost funkcije za izračunavanje koeficijenata sfernih harmonika i funkcije za postavljanje tih koeficijenata na *grid* čak $O(n^3)$ (zapravo je nešto manja, ali uzimamo $O(n^3)$ radi jednostavnosti analize).

Unutar tih funkcija poziva se još jedna funkcija koja ima istu toliku složenost! Petlje koje se izvode u njima kreću se u ovisnosti u redu korištenih sfernih harmonika, te je upravo to bio razlog znatnog usporenja izvođenja programa povećanjem reda. Nažalost, zbog korištenih algoritama nije moguće smanjiti složenost koda, ali moguće je ubrzati njegovo izvođenje. Unutarnje tri petlje služe za izračunavanje vrijednosti grida na pojedinoj koordinati, određenoj sa vanjske tri petlje. Umjesto da računamo vrijednost neke koordinate svaki put ispočetka, možemo zamijeniti redoslijede izvođenja petlji (3 unutarnje pretvorimo u 3 vanjske), te tada jednostavno kumulativno zbrajamo vrijednosti koeficijenata. Time smo smanjili broj poziva drugih funkcija (što također igra znatnu ulogu u kodu ovakve složenosti), te potrebu za ponovnim računanjem vrijednosti grida. Na ovaj način je složenost algoritma ostati ista, ali će se izvođenje jedne naredbe ubrzati, što bi se trebalo odraziti na cijelu učinkovitost koda.

Nadalje, vrijednost grida se u svakoj iteraciji pohranjivala u memoriju. Pohranjivanje je bilo izvedeno pomoću *std::map*-a. To je standardni *C++* spremnik podataka, koji omogućuje pohranjivanje i dohvaćanje podataka prema nekom određenom ključu. Iako praktičan u nekim primjenama, on ipak ima preveliku složenost za ovakvu konkretnu primjenu. Prilikom pristupanja pojedinom elementu, ili pak ubacivanja novog elementa u map, složenost algoritma koji se mora izvršiti je *O(log n)*. [6] Kako su vrijednosti koji sačinjavaju ključ po kojem pretražujemo elemente u map-u brojevi u pravilnim rasponima, to ih možemo koristiti za indeksiranje elemenata u običnom polju. Memoriju za polje koje bi sadržavalo podatke bismo morali ručno dinamički alocirati, te bismo smanjili apstrakciju koda (koja je pri optimizaciji zapravo poželjna), ali vrijeme pristupanja nekom elementu bi bilo *O(1)* (neovisno da li pišemo ili brišemo nešto u polje). To je značajno ubrzanje, a ovi konkretni rezultati imaju za posljedicu potpuno izbacivanje korištenja *std::map* spremnika u PDT-u.

Dio sustava koji izvršava proces prianjanja proteina ne nudi previše mogućnosti za poboljšanje, jer je kod već dosta optimiran. Jedina promjena koja bi mogla ubrzati ovaj dio je PDT-a je zamjena *map*-a sa poljima, kao i u dijelu sustava za obradu podataka.

3.5 Prebacivanje vizualizatora na *Linux* platformu

Vizualizator makromolekula (*VMol*) razvijen je za potrebe *PDT*-a, kako bi se mogao dobiti uvid u stanja pojedinih koraka procesa prianjanja proteina. On omogućuje učitavanje i prikaz proteinskih struktura iz *PDB* datoteka (eng. *Protein Data Bank*), a osim toga može i rekonstruirati površinu molekule koja je postavljena na grid.

VMol je uglavnom pisan u ANSI C++-u, što je omogućilo jednostavno prebacivanje (eng. *porting*) na Linux operativni sustav. Za one dijelove VMol-a koji se nisu mogli napisati kao ANSI, korištene su bibilioteke dostupne na velikom broju platformi, uključujući i Linux. To su: *OpenGL* (eng. *Open Graphics Library*) korištena za 3D prikaz molekula, i *GLUT* (eng. *OpenGL Utility Library*), koja omogućuje

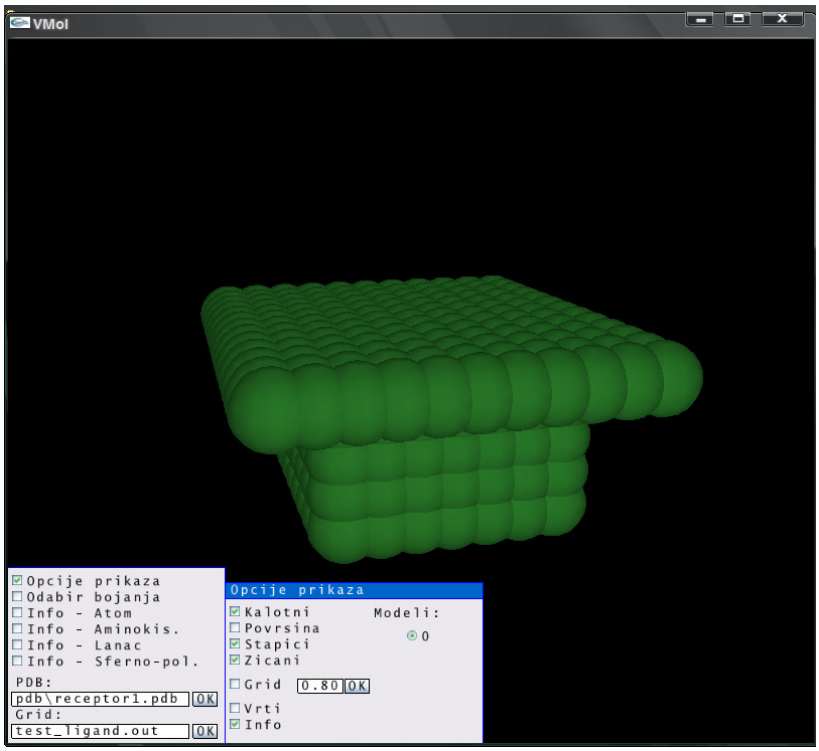
stvaranje prozora i postavljanje OpenGL konteksta. Za uspješno pokretanje VMol-a, potrebno je postaviti navedene bibilioteke na računalo.

3.6 Generiranje testnih proteina

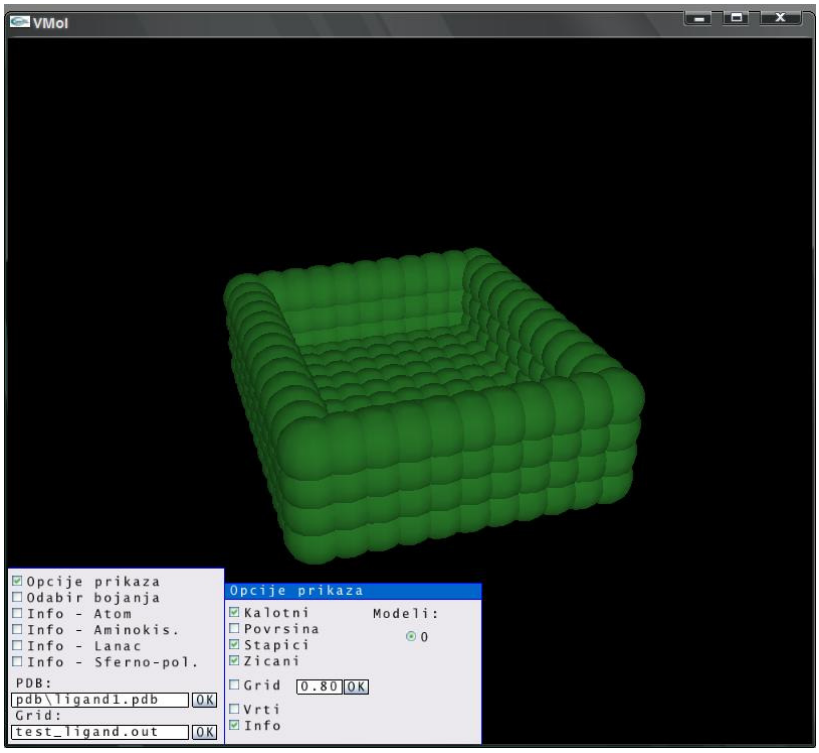
Kada bismo pokrenuli sustav sa neka dva realna proteina, zbog njihovog nepravilnog oblika bilo bi teško, ili skoro nemoguće odrediti da li je proces prianjanja dobro odradio svoj posao. Stoga, potrebno je izvršiti prianjanje nad ulaznim podatcima za koje znamo kako bi (barem otprilike) trebao izgledati rezultat.

Testni proteini su umjetne proteinske strukture pravilnog oblika, dizajnirane kao ulazni podatci pri testiranju sustava prianjanja proteina. PDT kao ulaz koristi dva proteina: ligand i receptor. Najlakše bi bilo utvrditi ispravnost prianjanja ako su ligand i receptor komplementarne strukture. Tada postoji samo jedno najbolje rješenje, dok su sva ostala nešto lošija (možda se površine proteina na još nekim mjestima dobro poklapaju, pa se zato dobije dobar rezultat i za neku drugu konformaciju).

Postavljanje proteina u prostor kako bi tvorili određeni oblik nije nimalo lak posao ako se pokušava napraviti na pamet. Stoga je u sklopu ovog rada izrađen jednostavan program koji omogućuje vizualno stvaranje i uređivanje proteinskih struktura. On je korišten za generiranje nekoliko testnih proteina. Slika 2. i slika 3. prikazuju komplementarne strukture liganda i receptora koje su stvorene sa programom za generiranje proteinskih struktura, a prikazane u VMol-u.



Slika 2. Primjer testnog proteina, struktura liganda prikazana u VMol-u.



Slika 3. Primjer testnog proteina, struktura receptora prikazana u VMol-u.

3.7 Pronalaženje i ispravljanje grešaka u procesu prianjanja korištenjem testnih proteina

Upotrebom ranije stvorenih testinh proteina, konačno smo mogli testirati ispravnost cijelog sustava. Pri prvom testiranju (kao što je bilo i očekivano) sustav nije našao točno rješenje problema prianjanja proteina. Nakon ispravljanja pronađenih grešaka (vezanih uz rotacijske matrice pri zapisivanju rezultata, te neke vezane uz sam proces prianjanja), PDT je pokazivao znatno poboljšanje u dobivenim rezultatima.

3.8 Povezivanje sustava u cjelinu

Svi dijelovi PDT sustava opisani do sada radili su odvojeno: učitavanje i obrada podataka, proces prianjanja proteina i vizualizacija rezultata. Kako bi se cijeli sustav mogao pokrenuti odjednom, napravljena je skripta koja pojedinačno poziva svaki od ovih dijelova.

4. Dodatne metode za buduću optimizaciju sustava

Svime opisanom u ovom radu optimizacija sustava PDT, naravno, još nije gotova. U budućem nastavku rada na ovom projektu, koji izlazi iz okvira ovog seminara, bilo bi ubrzavanje rada sustava na nižoj razini. Neke od metoda optimizacije *C++* koda su [7]:

1. **Podešavanje veličina struktura na potenciju broja 2** – kada kompajler izvodi indeksiranje polja struktura, on množi veličinu polja sa brojem indeksa željenog elementa u polju. Ako je veličina potencija broja 2, zahtjevna operacija množenja će biti zamijenjena jednostavnom operacijom pomaka.
2. **Minimiziranje broja lokalnih varijabli** – ako program sadrži mali broj lokalnih varijabli, kompajler će ih pokušati staviti u registre, umjesto na stog u memoriju. To može poboljšati performanse sustava na dva načina: nema potrebe pristupanja memoriji i, ako nema potrebe spremanja lokalnih varijabli na stog, kompajler neće morati postavljati i resetirati *frame pointer*.
3. **Smanjiti broj parametara funkcijama** – korištenje velikog broja parametara uzrokuje puno stavljanja i skidanja podataka na stog. To osim za varijable vrijedi i za strukture, te bilo kakve blokove podataka, a umjesto njih, bolje je slati samo pointere na te strukture.
4. **Korištenje referenci za predaju parametara funkcijama i za povratnu vrijednost funkcije (za tipove podataka veće od 4 bajta)** – ako se ne koriste reference, svi podatci će biti kopirani na stog, što može biti vrlo zahtjevna operacija za veću količinu podataka.
5. **Definiranje malih konstruktora klasa** – kako su oni pozvani prilikom stvaranja svakog objekta, tako se pozivaju i prilikom stvaranja samo privremenih objekata, ili velikog polja objekata tog tipa, što može usporiti izvođenje programa ako konstruktor izvodi neke zahtjevne operacije.

5. Testiranje

Tablica 1. Podatci o testiranju sustava.

Test 1	
Opis testa	Testiranje stabilnosti sustava izvođenjem procesa prianjanja proteina na dva realna proteina.
Cilj testa	Određivanje razloga nestabilnosti sustava i njegovo uklanjanje.
Testni podatci	Dva ulazna proteina (konkretno: *1crn.pdb* i *2rlx.pdb*), te konfiguracijska datoteka sa postavkama sustava.
Očekivani rezultat testa	PDT treba neprekinuto izvršiti proces prianjanja, te ispisati dobivenu rang listu, zapisati je u datoteku, te generirati izlaznu PDB datoteku.
Ispravljene greške u sustavu	Proces prianjanja bi se prekinuo prije završetka izvođenja – problem je bio u funkciji za računanje faktorijela sa visokom preciznošću. Funkcija je popravljena, te se proces izvodi bez prekida.

Tablica 2. Podatci o testiranju sustava.

Test 2	
Opis testa	Testiranje sustava nakon zamjene dijelova koda sa visokom preciznošću u *double*, te korištenjem matematičkih funkcija iz programskog paketa *GSL*.
Cilj testa	Provjeriti brzinu izvođenja i stabilnost sustava.
Testni podatci	Dva ulazna proteina (konkretno: *1crn.pdb* i *2rlx.pdb*), te konfiguracijska datoteka sa postavkama sustava.
Očekivani rezultat testa	Sustav bi trebao biti brži, stabilniji (na temelju prethodnog testa), ali uz veće (iako malo) odstupanje od točnog rješenja.
Ispravljene greške u sustavu	Na temelju dobivenih rezultata, biti će zadržana ovakva struktura sustava.

Tablica 3. Podatci o testiranju sustava.

Test 3	
Opis testa	Vizualizacija dobivene izlazne PDB datoteke.
Cilj testa	Testiranje ispravnosti zapisivanja rezultantnih konformacija u izlaznu datoteku, točnost rotiranja i translatiranja, te konačno sama točnost procesa prianjanja proteina.
Testni podatci	Izlazna PDB datoteka procesa prianjanja proteina.
Očekivani rezultat testa	Trebalo bi biti vidljivo kako se dvije proteinske strukture dodiruju svojim površinama, pod raznim kutevima i udaljenostima. Rezultat sa najvećim bodom (eng. *score*) bi trebao prikazivati dva proteina koji najbolje prianjanju površinom jedan uz drugi.
Ispravljene greške u sustavu	Zapisivanje u PDB datoteku je radilo ispravno. Greške su se javljale prilikom rotiranja proteina kako bi se prilagodili željenoj konformaciji (rotacijske matrice su bile pogrešno definirane, te su proteini bili plosnati). Radi grešaka dobivenih u ovome testu, sam proces dokiranja će biti testiran naknadno u novome testu.

Tablica 4. Podatci o testiranju sustava.

Test 4	
Opis testa	Isprobavanje alata za generiranje testnih proteina, te njihova vizualizacija.
Cilj testa	Provjeriti ispravnost zapisa proteina dobivenih ovim alatom.
Testni podatci	Sam alat nema testnih podataka, dok se za ulaz u vizualizator koriste proteini dobiveni ovim alatom.
Očekivani rezultat testa	Proteini prikazani u vizualizatoru bi trebali imati istu strukturu kao i u prikazu alata za njihovo generiranje.
Ispravljene greške u sustavu	Prilikom testiranja nije bilo odstupanjanja između prikaza stvorenih molekula u alatu za generiranje proteina, te istih prikazanih u vizualizatoru molekula.

Tablica 5. Podatci o testiranju sustava.

Test 5	
Opis testa	Vizualizacija dobivene izlazne PDB datoteke uz korištenje testnih proteinskih struktura kao ulaze u PDT. „Umjetne" testne proteinske strukture će nam omogućiti jednostavan uvid u ispravnost procesa prianjanja.
Cilj testa	Svrha ponovnog testa s vizualizacijom je testiranje samog procesa prianjanja, i to jedom kada već znamo da sitniji međukoraci ispravno funkcioniraju.
Testni podatci	Izlazna PDB datoteka procesa prianjanja proteina.
Očekivani rezultat testa	Trebalo bi biti vidljivo kako se dvije proteinske strukture dodiruju svojim površinama, pod raznim kutevima i udaljenostima. Rezultat sa najvećim bodom (eng. *score*) bi trebao prikazivati dva proteina koji najbolje prianjanju površinom jedan uz drugi.
Ispravljene greške u sustavu	Dobivene konformacije su se preklapale, te nisu zadovoljavale očekivane rezultate. Ispravljene su greške u rotacijskim matricama pri zapisivanju rezultata u datoteku, te greške u samom procesu prianjanja proteina.

Tablica 6. Podatci o testiranju sustava.

Test 6	
Opis testa	Testiranje vizualizacije na *Linux* platformi.
Cilj testa	Utvrđivanje ispravnosti rada vizualizacije nakon prebacivanja na *Linux* platformu.
Testni podatci	Bilo koja PDB datoteka.
Očekivani rezultat testa	Vizualizator bi trebao raditi stabilno, te se ponašati jednako kao i inačica za *Windows*-e.
Ispravljene greške u sustavu	Prilikom testiranja VMol-a na Linux platformi, nisu uočena odstupanja od očekivanih rezultata testiranja.

6. Zaključak

PDT sustav nudi mogućnost predviđanja proteinskih struktura koje nastaju interakcijom dva proteina. Ova funkcionalnost je od velikog značenja primjerice u farmaciji prilikom otkrivanja novih lijekova. No, kako PDT nikada nije bio u potpunosti dovršen, nikada se nije mogao konkretno niti upotrijebiti. Testiranjem ovog sustava, utvrđene su neke njegove mane i nedostatci, kao što su nestabilnost za određene ulazne podatke, te sporost obrade ulaznih podataka. Predložene su i implementirane metode optimizacije i poboljšanja sustava. Rezultati dobiveni izmjenama PDT-a pokazuju kako je sustav nakon procesa optimizacije uistinu postao stabilniji i brži. Time su opravdani svi postupci opisani u ovome radu.

U daljnjem radu na ovom projektu, cijeli bi se sustav mogao još i više optimirati. Dodatne metode ubrzanja sustava trebale bi biti implementrane na nižoj razini apstrakcije programa, te bliže vezane uz korišteni kompajler.

7. Literatura

[1] Šikić, M., Dokmanić, I., Ćurić, J., Petrović, J., Antulov-Fantulin, N., Piškorec, M., Prebeg, I., Peretin, G., Čolić, D., Sović, I., *Dizajn dokument PDT-1*, FER, 2007.

[2] Šikić, M., Dokmanić, I., Ćurić, J., Petrović, J., Antulov-Fantulin, N., Piškorec, M., Prebeg, I., Peretin, G., Čolić, D., Sović, I., *Funkcionalna specifikacija PDT*, FER 2007.

[3] Majić, I., *Predviđanje proteinskih interakcija rastavom proteinskih kompleksa*, diplomski rad, FER, 2006.

[4] Dokmanić, I., *Algoritam za predviđanje strukture proteinskih kompleksa korištenjem razvoja površine u redove funkcija*, diplomski rad, 2007.

[5] *The GNU Bignum Library*, http://gmplib.org/, datum pristupa dokumentu: 17.05.2009.

[6] *Map (C++ container)*, http://en.wikipedia.org/wiki/Map_(C%2B%2B_container), datum pristupa dokumentu: 17.05.2009.

[7] *Optimizing C and C++ Code*, http://www.eventhelix.com/RealtimeMantra/Basics/OptimizingCAndCPPCode.htm, datum pristupa dokumentu: 17.05.2009.

8. Sažetak

Protein Docking Tool (PDT) je aplikacija namijenjena korištenju u bioinformatičke svrhe. Ona omogućuje predviđanje struktura proteinskih kompleksa na temelju poznatih struktura proteina koji ih tvore. U okviru ovog seminara nastavljen je rad na već postojećem projektu. Prikazan je opći opis sustava i njegovo trenutno stanje. U radu su iznesene neke ključne mane i nedostatci PDT-a. Proučene su i implementirane razne metode optimizacije i nadogradnje sustava. Najvažnija poglavlja u ovom radu su: Uređivanje koda, Popravljanje stabilnosti sustava, Upotreba programskih biblioteka, Ubrzavanje rada sustava, Prebacivanje vizualizatora na Linux platformu, Generiranje testnih proteina, Pronalaženje i ispravljanje grešaka u procesu prianjanja korištenjem testnih proteina, Povezivanje sustava u cjelinu i Testiranje.

Osim navedenih, dane su i dodatne metode optimizacije koje bi bilo prigodno također primijeniti, ali nisu obuhvaćene u ovome radu.