SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3615

Upravljanje programskim okvirom za evolucijsko računanje

Domagoj Stanković

Voditelj: prof. dr. sc. Domagoj Jakobović

Zagreb, svibanj, 2014.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 12. ožujka 2014.

ZAVRŠNI ZADATAK br. 3615

Pristupnik: **Domagoj Stanković**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Upravljanje programskim okvirom za evolucijsko računanje**

Opis zadatka:

Opisati glavna svojstva stohastičkih optimizacijskih algoritama i njihovu primjenu u praksi. Ostvariti grafičko korisničko sučelje za olakšano rukovanje postojećim programskim okruženjem za evolucijsko računanje. Omogućiti ujednačeno rukovanje algoritmom optimizacije neovisno o optimizacijskom problemu. Osmisliti strukturu skladišta za pohranu rezultata stohastičkih optimizacijskih algoritama. Ostvariti programski pristup za pohranu i dohvat podataka iz skladišta na temelju zadanih svojstava algoritma i optimizacijskog problema. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 13. lipnja 2014.

Mentor:

Izv.prof.dr.sc. Domagoj Jakobović

Djelovođa:

Doc.dr.sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof.dr.sc. Siniša Srbljić

Zahvaljujem svom mentoru, prof. dr. sc. Domagoju Jakoboviću, na izuzetnom strpljenju, razumijevanju i podršci.

Također velika hvala mojoj obitelji na bezuvjetnom odricanju, potpori, utjesi i motivaciji. Hvala vam što ste uvijek bili uz mene kada sam vas trebao.

Sadržaj

1. Uvod

Unatoč tome što su računala danas nevjerojatno brza, u praksi često nailazimo na probleme koje nije moguće riješiti tehnikom grube sile (engl. *brute-force*) tj. pretraživanjem cjelokupnog prostora rješenja. Ti problemi poznati su kao NP-potpuni i NP-teški problemi. Takvi su problemi netraktabilni tj. za takve probleme ne postoje algoritmi čija je složenost zadovoljavajuća. Primjeri takvih problema su problem trgovačkog putnika, problem izrade rasporeda predavanja i raspoređivanja studenata u grupe, problem bojanja grafova, problem raspoređivanja medicinskih sestara u smjene itd.

Kako često nije potrebno pronaći optimalno već dovoljno dobro rješenje, postoje brojni algoritmi koji nam u tome pomažu, a imaju nisku računsku složenost. Takvi algoritmi nazivaju se heuristički algoritmi, ili jednostavnije, heuristike. U današnje doba posebno su nam zanimljive metaheuristike. Metaheuristika je skup algoritamskih koncepata koji koristimo za definiranje heurističkih metoda primjenjivih na širok skup problema. Možemo reći da je metaheuristika heuristika opće namjene čiji je zadatak usmjeravanje problemski specifičnih heuristika prema području prostora rješenja u kojem se nalaze dobra rješenja [11]. Primjeri su metaheuristika algoritam simuliranog kaljenja, genetski algoritmi, genetsko programiranje, algoritam roja čestica, algoritam diferencijske evolucije itd.

Uz priču o algoritmima pretraživanja važno je spomenuti *no-free-lunch* teorem koji kaže da su svi algoritmi pretraživanja prosječno dobri. Za jednu vrstu problema određeni algoritam ponašat će se bolje u odnosu na neki drugi algoritam dok će za drugi skup problema vrijediti obrnuto. Iz toga slijedi zaključak da je jako važno dobro odabrati algoritam kojim će se rješavati određeni problem jer bi u suprotnom i običan algoritam slijepe pretrage mogao davati bolje rezultate.

Ovaj završni rad sastoji se od dva dijela, korisničkog sučelja za rukovanje programskim okvirom za evolucijsko računanje *Evolutionary Computation Framework*-om (ECF) i korisničkog sučelja za rukovanje skladištem rezultata rada optimizacijskih algoritama.

2. Korisničko sučelje za ECF

ECF je postojeći programski okvir za evolucijsko računanje napisan u programskom jeziku C++. Razvijan je kroz niz godina te sadrži brojne funkcionalnosti potrebne za rješavanje mnogih optimizacijskih problema algoritmima evolucijskog računanja. ECF se dosad pokretao isključivo preko konzole uz ručno pisanje konfiguracijskih datoteka što je bilo poprilično naporno uz veliku mogućnost pogreške. Uz to, rezultati dobiveni radom algoritama bili su u tekstualnom obliku te nisu bili vizualizirani. Iz tih razloga javila se potreba za izradom korisničkog sučelja kojim će se na jednostavan način upravljati ECF-om. Zamisao je da sučelje upravlja programskim okvirom neovisno o problemu, algoritmu, genotipu i ostalim parametrima.

Optimizacijski problem definiran je kroz izvršnu datoteku dobivenu prevođenjem programa koji koristi biblioteku ECF. Iz te je datoteke moguće, uz pokretanje s određenim parametrima, dobiti datoteku s popisom svih algoritama, genotipa i ostalih parametara koje nudi ECF. Na taj način grafičko sučelje zna kako prikazati i ponuditi korisniku sve opcije koje nudi ECF.

Algoritam [13] je konačan slijed dobro definiranih naredbi za ostvarenje zadatka, koji će za dano početno stanje završiti u definiranom konačnom stanju. Algoritam ustvari određuje način na koji će se zadani problem rješavati. Primjeri algoritama evolucijskog računanja su: algoritam diferencijske evolucije, algoritam kolonije mrava i genetski algoritam.

Genotip označava jedno rješenje optimizacijskog problema i posjeduje mjeru dobrote, tj. numeričku vrijednost koja označava u kolikoj mjeri rješenje zadovoljava određeni problem. Cilj je evolucijskog računanja pronaći rješenje s što je većom mogućom dobrotom, ali ne postoji garancija da će algoritam pronaći optimalno rješenje. Postoje različite vrste genotipa kao što su genotip u obliku stabla, genotip u obliku niza bitova i genotip u obliku polja decimalnih brojeva.

Parametri definiraju dodatne podatke o algoritmu odnosno problemu. Algoritmi i problemi predstavljaju općenite principe, dok s parametrima definiramo konkretni algoritam odnosno problem. Tako je pokus definiran kao skup algoritma i problema s pripadajućim parametrima. Primjeri parametara su: veličina populacije, najveći dopušteni broj iteracija i selekcijski pritisak.

Algoritmi, genotipi i ostali parametri koji će se koristiti u pokusu definirani su kroz konfiguracijsku datoteku u XML obliku. Ta se datoteka predaje ECF-u koji ju čita te oblikuje sve potrebno za izvođenje evolucijskog računanja. Konfiguracijska se datoteka sastoji od skupa algoritama, skupa genotipa te skupa dodatnih parametara. Za svaki algoritam i genotip naveden je niz pripadnih parametara.

Rezultat rada ECF-a odnosno rezultat pokretanja pokusa tekstualna je datoteka u kojoj su zapisane neke osnovne značajke stanja dobrote populacije po generacijama.

Korisničko sučelje za rukovanje ECF-om započeto je u okviru kolegija Projekt koji sam radio u suradnji s kolegama Vlahom Polutom i Svenom Vidakom. Kolege su bile zadužene za dio koji je obuhvaćao parsiranje potrebnih dokumenata koji nastaju tokom rada ECF-a te za komunikaciju korisničkog sučelja s procesom ECF-a. Nad tim je komponentama potom izgrađeno grafičko sučelje.

2.1 ECF Lab

ECF Lab je *desktop* aplikacija nastala kao rezultat prethodno iznesenih težnji. Aplikacija je napisana u programskom jeziku Java korištenjem alata Swing tako da je neovisna o platformi i operacijskom sustavu. Korištena je mogućnost *Look and Feel* kako bi aplikacija poprimila izgled autohtone aplikacije (engl. *native*).

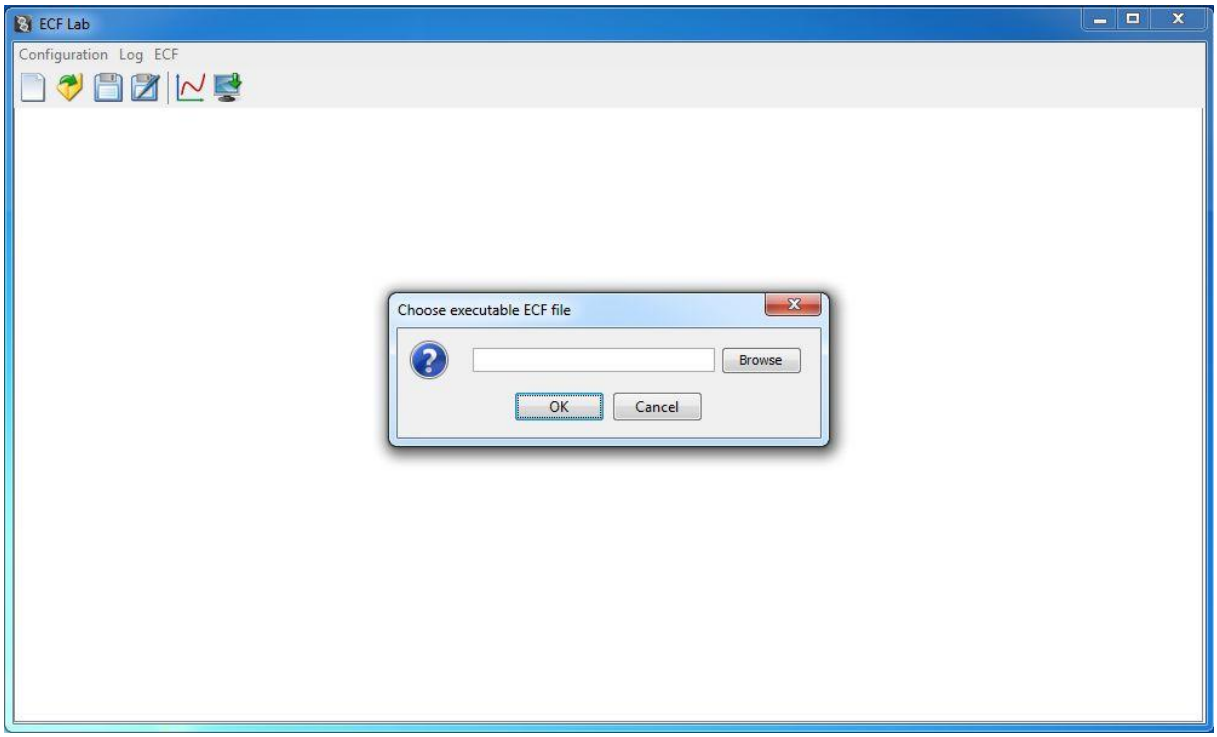
Aplikacija se sastoji od 3 dijela: dijela za komunikaciju s procesom ECF-a, dijela za parsiranje i generiranje svih potrebnih datoteka te grafičkog

korisničkog sučelja. Kako je ECF napisan u programskom jeziku C++, a grafičko sučelje u programskom jeziku Java, komunikaciju među tim komponentama ostvaruje se tako da grafičko sučelje pokreće proces ECF-a preko konzole. Proces ECF-a pokretat će se iz 2 razloga, za dobivanje popisa svih dostupnih opcija te za izvođenje pokusa. Za izvođenje se pri stvaranju procesa odredi put do konfiguracijske datoteke te mjesto gdje će se spremiti rezultat izvođenja. Nakon gašenja procesa jednostavno se pročitaju podaci koji se nalaze na dogovorenom mjestu.

2.1.1 Početni zaslon

Pri pokretanju aplikacije omogućen je odabir izvršne datoteke nad kojom će se izvršavati željene operacije. Nakon odabira, *ECF Lab* će zatražiti od izvršne datoteke ispis svih algoritama, genotipa i ostalih parametara kako bi ih znao prikazati korisniku.

Nakon toga korisniku se nude mogućnosti vezane za konfiguraciju, rezultate i izvršnu datoteku te dodatne informacije. Bilo koju od mogućnosti moguće je odabrati preko padajućeg izbornika, alatne trake ili tipkovničke prečice. Prelaskom pokazivača miša preko stavke u određenoj traci prikazuje se i opis određene akcije. Stvaranje i otvaranje konfiguracijske datoteke otvaraju novu karticu.

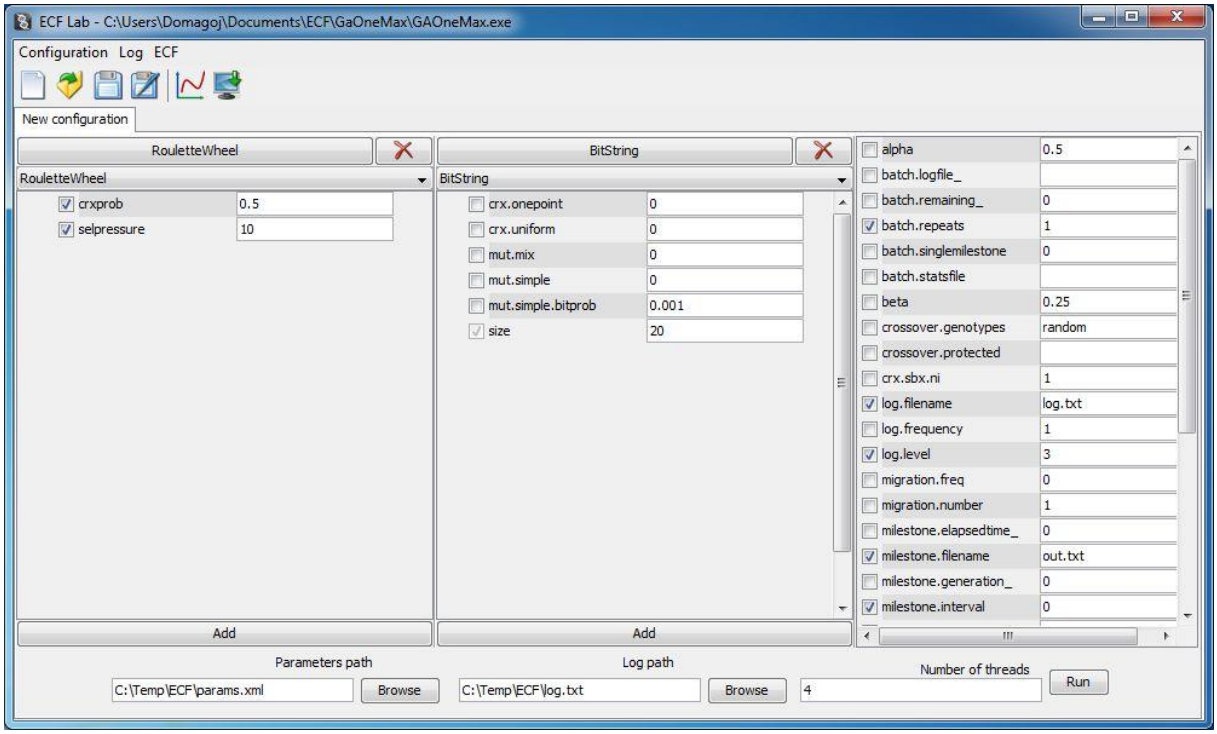


Slika 2.1 Početni zaslon

2.1.2 Stvaranje nove konfiguracije

Kako bi izvođenje pokusa moglo početi, potrebno je navesti algoritme i njihove parametre, genotipe i njihove parametre te dodatne opcije. To se ostvaruje tako da se stvori nova konfiguracijska datoteka odabirom željenih algoritama i genotipa te unosom njihovih parametara. Parametri se unose označavanjem odgovarajuće kućice te upisom vrijednosti parametra. Obvezni su parametri već označeni i ne mogu biti odznačeni. Kako je omogućen unos više algoritama i genotipa, potrebno je pritisnuti gumb *Add* kako bi algoritam odnosno genotip bio upisan u konfiguracijsku datoteku. Nakon tog koraka moguće je unijeti idući algoritam odnosno genotip. U slučaju da želimo promijeniti neki od navedenih algoritama ili genotipa, potrebno je kliknuti na taj unos nakon čega će se otvoriti novi prozor u kojem će biti moguće mijenjati pojedine parametre. Za slučaj da želimo ukloniti određeni algoritam ili genotip, potrebno je pritisnuti gumb s crvenim znakom

„X“ pokraj unosa. Konfiguracijska datoteka bit će spremljena tek kada se ili pokrene rad ECF-a ili odabere opcija za spremanje konfiguracije.



Slika 2.2 Stvaranje nove konfiguracijske datoteke

2.1.3 Otvaranje postojeće konfiguracijske datoteke

Ako je dostupna već postojeća konfiguracijska datoteka, ta se datoteka može učitati pomoću aplikacije kako bi se izmijenila ili iskoristila za pokretanje novog pokusa.

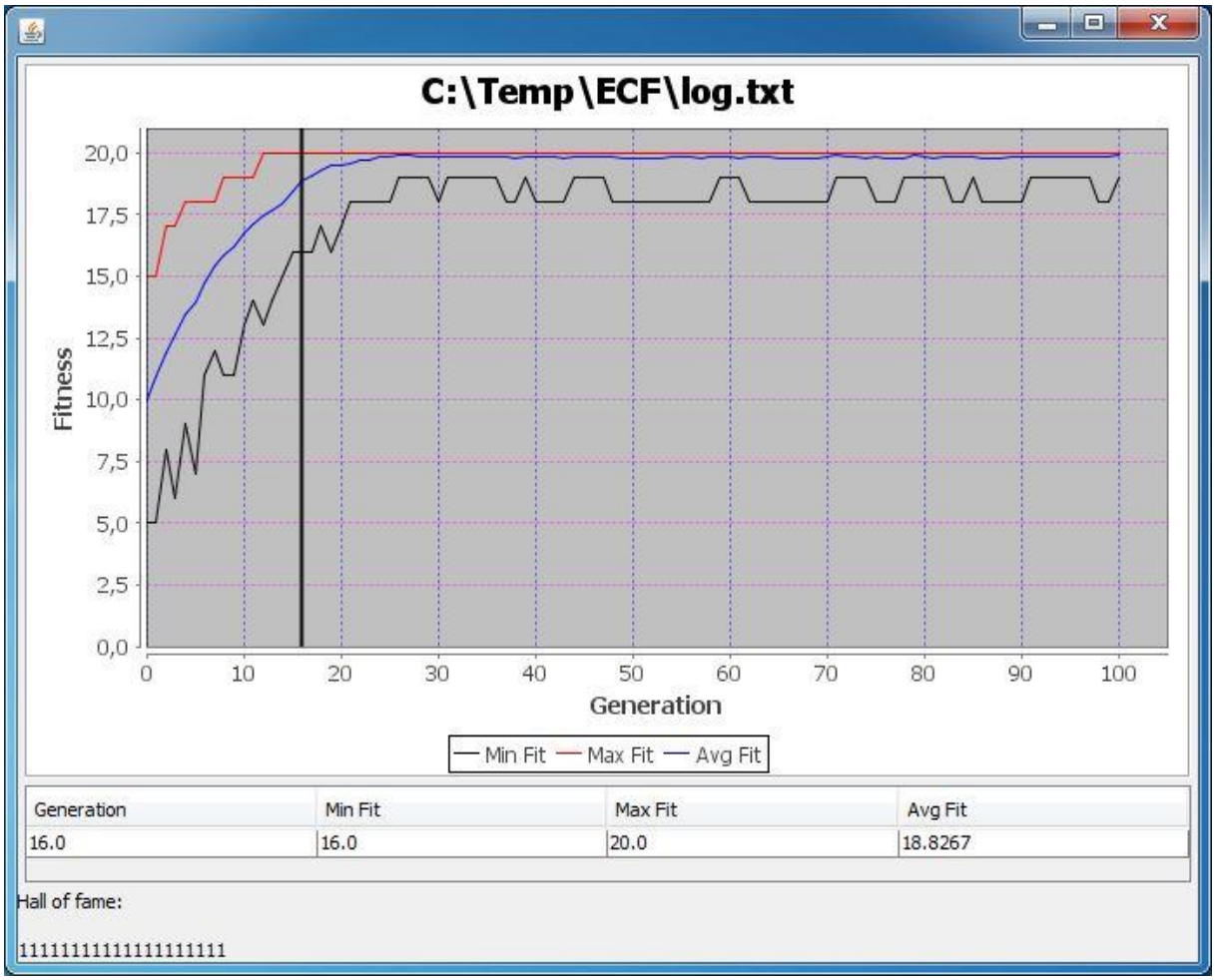
2.1.4 Spremanje trenutne konfiguracije

Trenutno odabrana konfiguracija može se spremiti u konfiguracijsku datoteku. Ako se odabere *Configuration -> Save,* konfiguracija će se spremiti pod imenom navedenim ispod u rubrici *Parameters path.* Taj put može se

promijeniti pritiskom na gumb *Browse.* Nakon spremanja konfiguracije, naziv označene kartice pretvorit će se u put pod kojim je ta konfiguracija spremljena. Ako se odabere *Configuration -> Save As,* iskočit će izbornik za odabir mjesta za spremanje konfiguracijske datoteke. Spremanje na taj način neće utjecati na naziv kartice.

2.1.5 Otvaranje rezultata

U okviru aplikacije *ECF Lab* moguće je otvoriti datoteke u kojima su zapisani rezultati izvođenja ECF-a. To se ostvaruje odabirom *Log -> Open* nakon čega se odabire put do željene datoteke. Nakon što se odabrana datoteka isparsira, pojavljuje se novi prozor koji prikazuje graf dobrote populacije kroz generacije. Vodoravna os predstavlja broj generacije, a okomita dobrotu. Uz graf se nalazi i legenda koja govori koja linija prikazuje maksimalnu, koja minimalnu, a koja srednju dobrotu. Lijevim klikom na graf pojavljuje se okomita crna linija koja označava generaciju i siječe linije koje predstavljaju dobrotu. Kako je na taj način teško očitati koje su vrijednosti dobrote za odabranu generaciju, ispod grafa nalazi se i tablica u kojoj pišu odgovarajuće vrijednosti za odabranu generaciju. Ispod tablice nalazi se i prikaz najboljeg rješenja (engl. *Hall of fame*). Desnim klikom na graf moguće je odabrati još neke dodatne opcije kao što su zumiranje, mijenjanje svojstava grafa, kopiranje slike grafa te spremanje ili ispis iste. Zumiranje je također moguće ostvariti označavanjem dijela grafa koji se želi zumirati.



Slika 2.3 Vizualizacija rezultata pokusa

2.1.6 Spremanje rezultata

Odabirom opcije *Log -> Save* moguće je spremiti rezultat izvođenja pokusa za određenu karticu. Otvorit će se izbornik u kojem se treba odabrati put gdje će se spremiti odgovarajuća datoteka s rezultatima.

2.1.7 Mijenjanje izvršne datoteke

Tijekom rada aplikacije *ECF Lab* moguće je i promijeniti izvršnu datoteku nad kojom će se izvoditi pokusi. To se obavlja odabirom opcije *ECF -> Change ECF* nakon čega se otvara prozor u kojem je potrebno odabrati

novu izvršnu datoteku. Nakon toga će *ECF Lab* pozvati ispis svih algoritama, genotipa i ostalih opcija nad novom izvršnom datotekom. Ti će podaci biti vidljivi tek pri otvaranju nove kartice. Na vrhu prozora, iznad izborničke trake, uz ime aplikacije naveden je i put do izvršne datoteke koja se trenutno koristi.

2.1.8 *Web* stranica ECF projekta

Više informacija o ECF-u možete potražiti na službenoj *web* stranici projekta [1] ili iz *ECF Lab*-a jednostavnim pritiskom na opciju *ECF home page* iz padajućeg izbornika *ECF.* Odabirom te opcije, otvorit će se *web* preglednik s odgovarajućom stranicom.

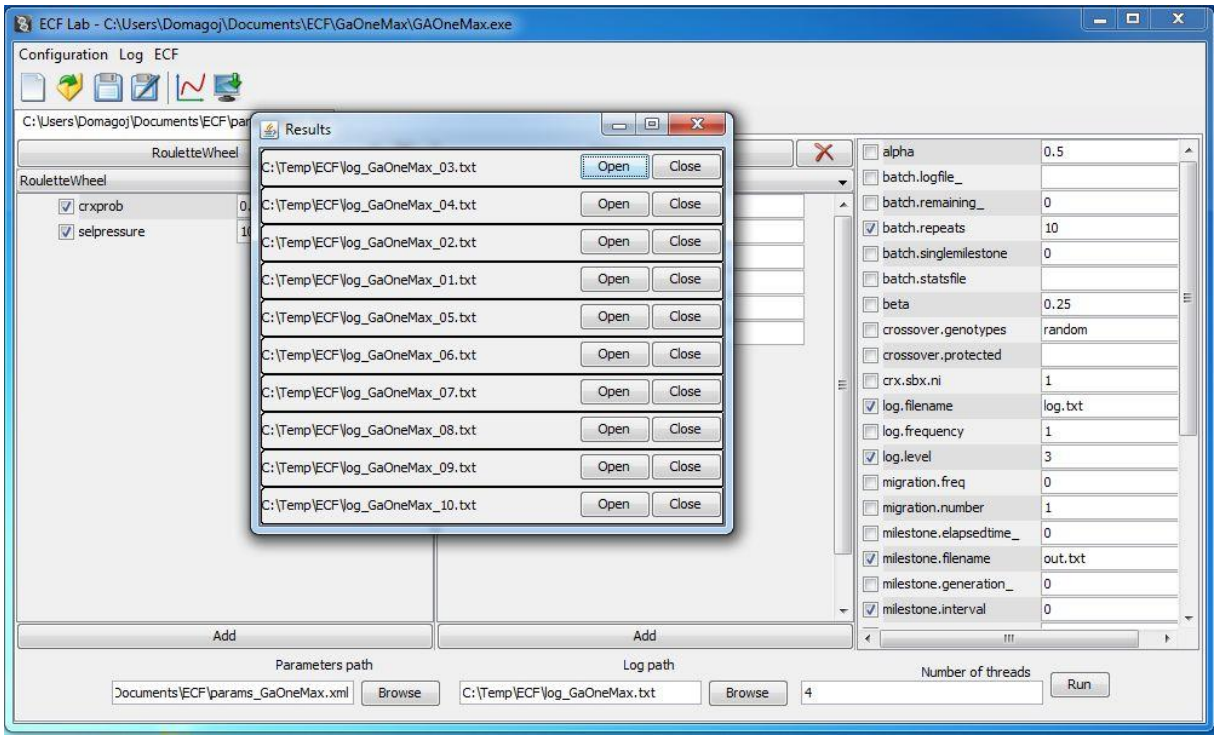
2.1.9 Pokretanje pokusa

Nakon što se odaberu svi željeni algoritmi i genotipi te se definiraju svi potrebni parametri, moguće je pokrenuti izvođenje pokusa. Lijevo od gumba *Run* nalaze se 3 rubrike: rubrika za odabir mjesta gdje će se spremiti konfiguracijska datoteka, rubrika za odabir mjesta gdje će se spremiti rezultat izvođenja pokusa te rubrika za odabir broja dretvi koje će izvoditi zadani pokus. U dodatnim parametrima postoji opcija pod imenom „batch.repeats“ koja govori koliko puta zaredom će se pokus izvršiti. Kada je „batch.repeats“ označen i vrijednost mu je veća od 1, ECF će automatski dodavati broj ponavljanja u ime datoteke ispred ekstenzije, npr. log.txt postat će log_XX.txt gdje XX označava broj ponavljanja. *ECF Lab* će dočekati sve datoteke nastale na taj način te omogućiti prikazivanje istih. U slučaju da je označeno više od jedne dretve, *ECF Lab* će promijeniti konfiguraciju kako bi se omogućilo paralelno izvođenje pokusa. „batch.repeats“ će se postaviti na 1 te će se stvoriti onoliko poslova koliko je ponavljanja bilo označeno. Poslovi će se staviti u red izvođenja koje će izvoditi bazen dretvi (engl. *thread pool*) sačinjen od onoliko dretvi koliko je bilo označeno u odgovarajućoj rubrici. U

svim će se ostalim slučajevima broj dretvi automatski postaviti na 1 jer nema potrebe za paralelizacijom.

Pokus se pokreće pritiskom na gumb „*Run“.* Kad ECF završi s radom odnosno kad završi pojedini pokus, pojavljuje se novi prozor u kojem se nalaze svi rezultati nastali radom ECF-a. Rezultat pojedinog pokusa može se otvoriti pritiskom na gumb „*Open*“ uz odgovarajući unos nakon čega će se otvoriti novi prozor u kojem će taj rezultat biti vizualiziran. Unos se također može i ukloniti s tog popisa pritiskom na gumb „*Close*“. Prozor s popisom svih rezultata također se može otvoriti i odabirom opcije *Log -> Results frame*.

Napomena: Na mjestu gdje su spremljeni rezultati pokusa, uz pojedini rezultat nalazi se i datoteka s identičnim imenom uz nadodanu ekstenziju .err. Tu datoteku automatski stvara ECF na početku rada te ju koristi za zapis pogrešaka ako do njih eventualno dođe.



Slika 2.4 Pokretanje pokusa i prikaz dobivenih datoteka rezultata

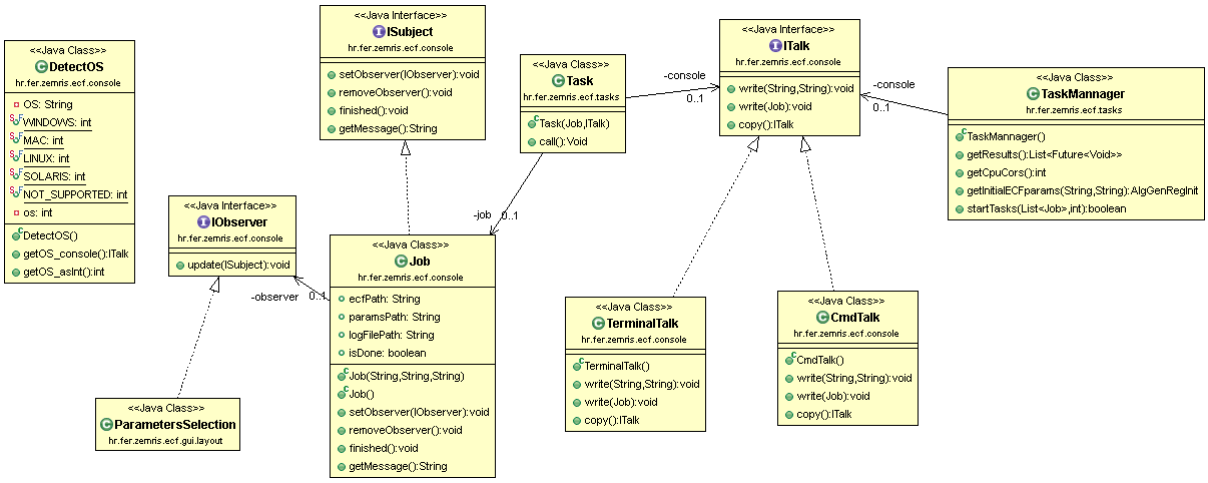
2.2 Struktura aplikacije

U ovom poglavlju bit će opisana struktura aplikacije kroz tri glavna dijela. Uz svaki dio bit će priloženi i dijagrami razreda.

2.2.1 Komunikacija s procesom ECF-a

Komunikacija s procesom ECF-a odvija se tako da grafičko sučelje preko konzole pokrene proces ECF-a uz sve potrebne parametre. To se obavlja preko sučelja *ITalk* koje je zaduženo za pisanje u konzolu operacijskog sustava na kojem se izvodi aplikacija. Razred *DetectOS* zadužen je za prepoznavanje operacijskog sustava i instanciranje odgovarajuće implementacije sučelja *ITalk.* Za operacijski sustav MS Windows predviđen je razred *CmdTalk* dok je za *Unix* predviđen razred *TerminalTalk.* Razred *Job* predstavlja posao koji je potrebno obaviti, a

implementira sučelje *ISubject* koje je zaduženo za obavještavanje promatrača da je posao obavljen. Trenutačno je jedini promatrač razred *ParametersSelection* koji je zadužen za prikaz informacija o pojedinom pokusu. Razred *TaskManager* zadužen je za paralelno izvođenje poslova pri čemu će za svaki posao stvoriti novi proces ECF-a uz pomoć razreda *Task*.

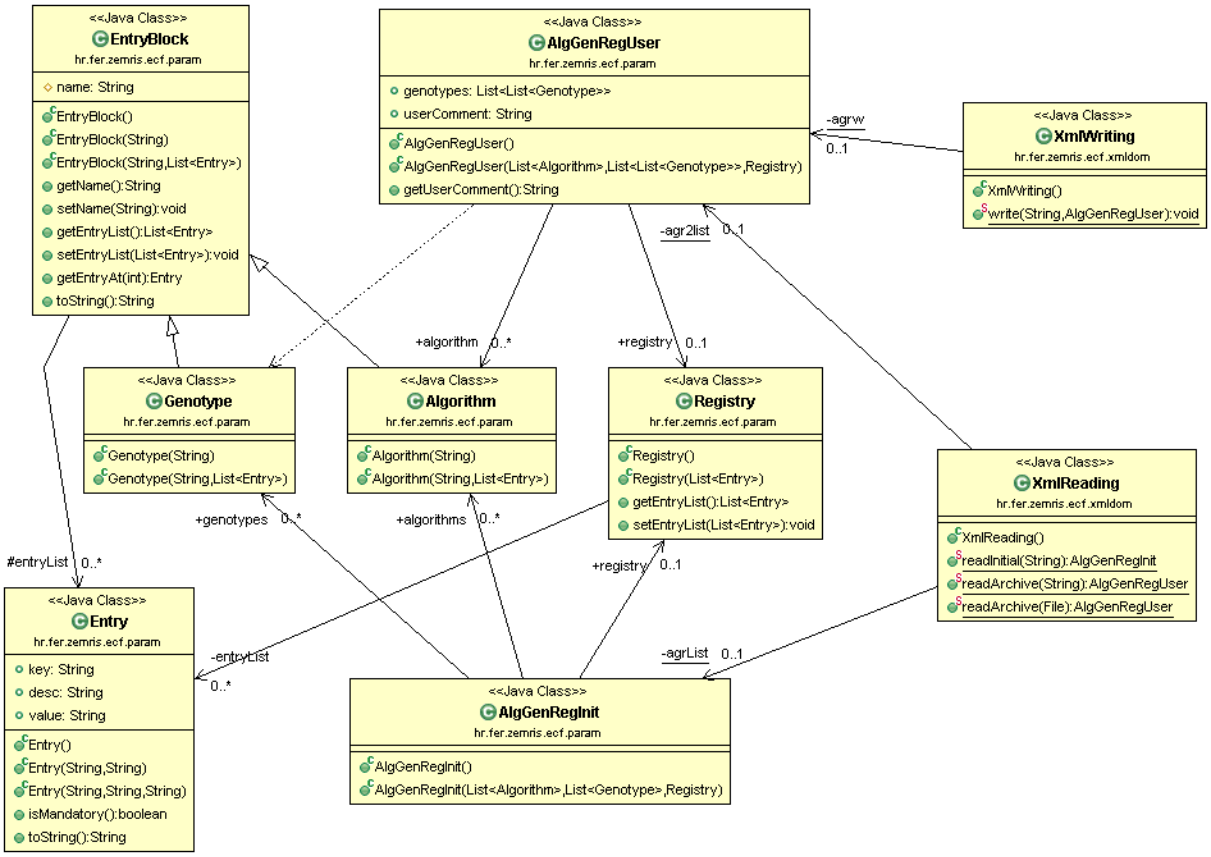


Slika 2.5 Dijagram razreda za dio aplikacije zadužen za komunikaciju s procesom ECF-a

2.2.2 Parsiranje i generiranje potrebnih datoteka

Kako se konfiguracija algoritama i genotipa predaje ECF-u u XML obliku, potrebno je imati razrede koji će se time baviti. Temeljni razred ove komponente je razred *Entry* koji predstavlja jedan unos u XML datoteci i sadrži ključ, vrijednost te opis. Skup unosa oblikovan je razredom *EntryBlock* kojeg naslijeđuju razredi *Algorithm* i *Genotype.* Ti razredi predstavljaju jedan algoritam odnosno jedan genotip zajedno s pripadajućim parametrima. Razred *Registry* predstavlja dodatne parametre o problemu te se također sastoji od niza unosa. Razredi *AlgGenRegInit* i *AlgGenRegUser* služe kao skup algoritama, genotipa i dodatnih parametara o problemu s ciljem

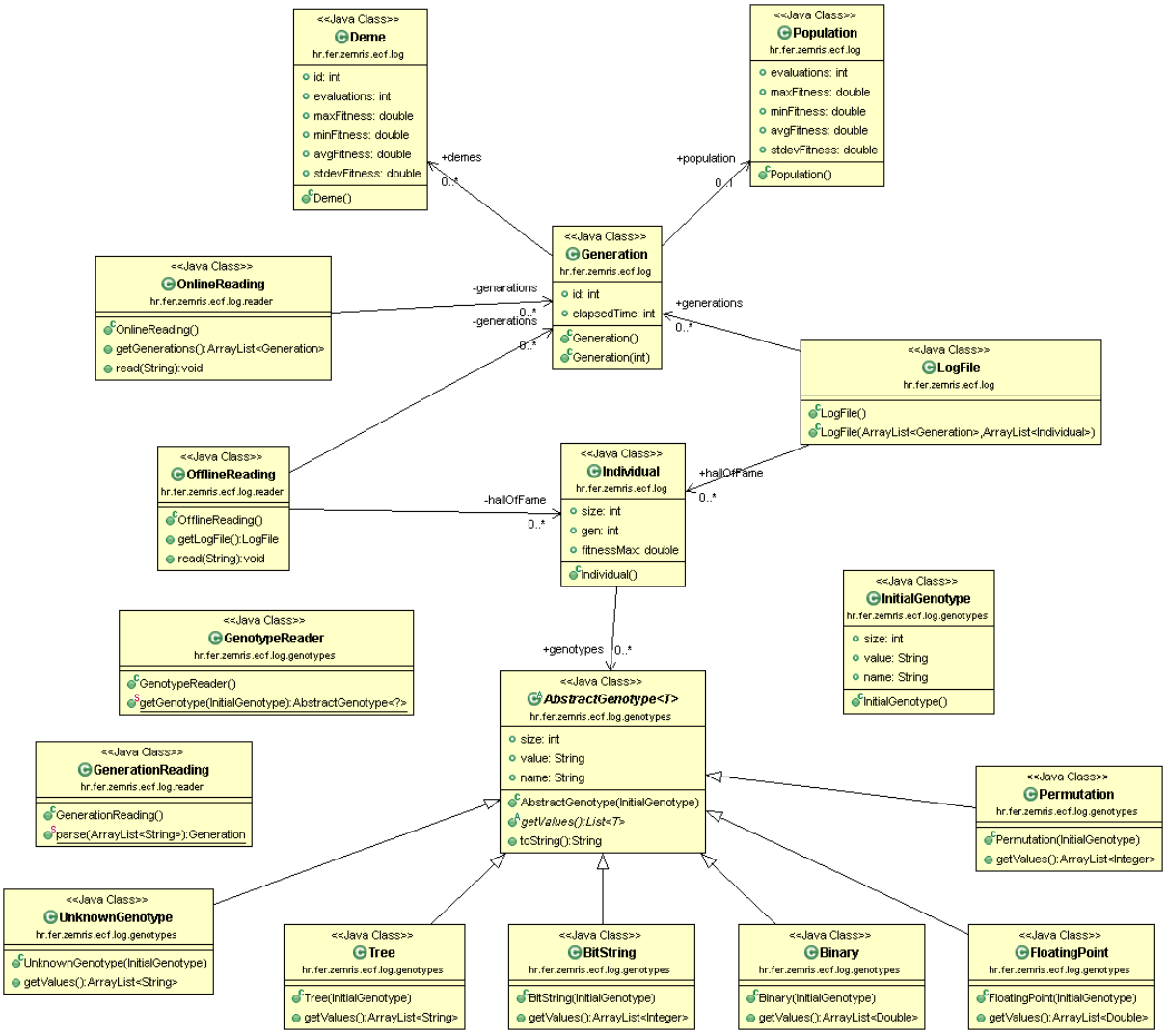
dohvaćanja svih mogućnosti ECF-a odnosno za pisanje i čitanje iz konfiguracijske datoteke. Razred *XmlWriting* nudi statičku metodu *write* za pisanje skupa parametara u XML datoteku dok razred *XmlReading* nudi statičke metode za čitanje iz XML datoteke. Na slici 2.6 prikazan je pripadajući dijagram razreda.



Slika 2.6 Razredi potrebni za pisanje/čitanje u/iz konfiguracijskih datoteka

Na slici 2.7 nalazi se dijagram razreda koji vizualno opisuje razrede potrebne za čitanje datoteka s rezultatima. Čitati se može na dva načina: tokom rada ECF-a (engl. *Online*) i nakon što ECF završi s radom (engl. *Offline*). Kako *OnlineReading* nije u potpunosti funkcionalan, grafičko sučelje ni ne nudi mogućnost takvog načina čitanja, stoga ćemo se zadržati samo na drugom načinu. Metoda *read* razreda *OfflineReading* parsira zadanu

datoteku i stvara primjerak razreda *LogFile* koji je podijeljen na niz generacija te zapis najboljih rješenja (engl. *Hall of fame*). Jedna generacija može se sastojati od više zapisa za podpopulaciju (engl. *Deme*) te od jednog zapisa za cjelokupnu populaciju. Razredi *Deme* i *Population* sadrže podatke o iznosu najveće dobrote, iznosu najmanje dobrote, srednjoj vrijednosti dobrote te o srednjem kvadratnom odstupanju dobrote. *Hall of fame* sastavljen je od niza primjeraka razreda *Individual* od kojih svaki sadrži niz genotipa koji predstavljaju najbolja rješenja. Genotip može biti raznih vrsta, a trenutne implementacije tog razreda su *Tree, BitString, Binary, FloatingPoint, Permutation* te *UnknownGenotype.* Razredi *GenotypeReader* i *GenerationReading* služe kao pomoćni razredi pri parsiranju datoteke s rezultatima.

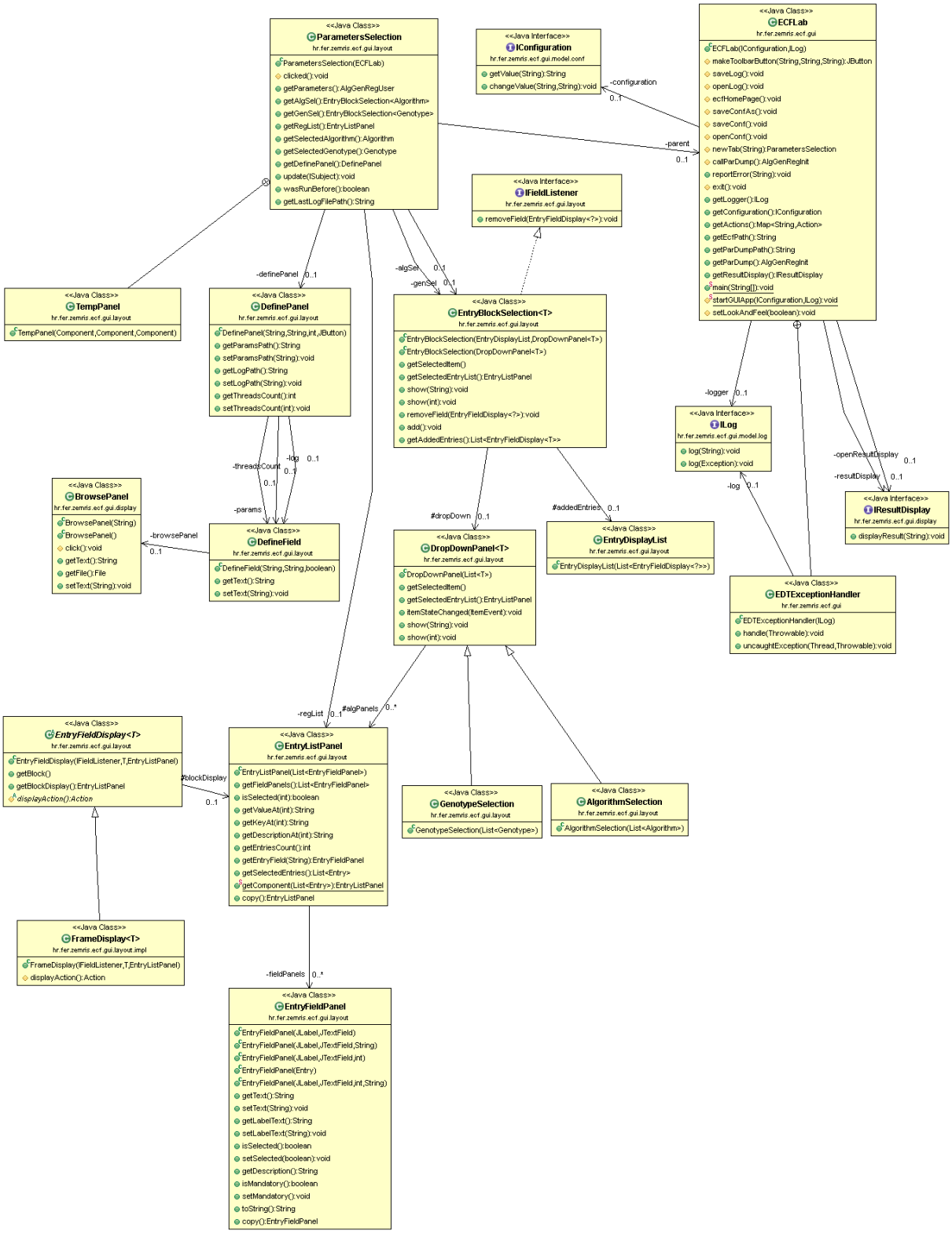


Slika 2.7 Razredi potrebni za čitanje datoteka s rezultatima

2.2.3 Grafičko sučelje

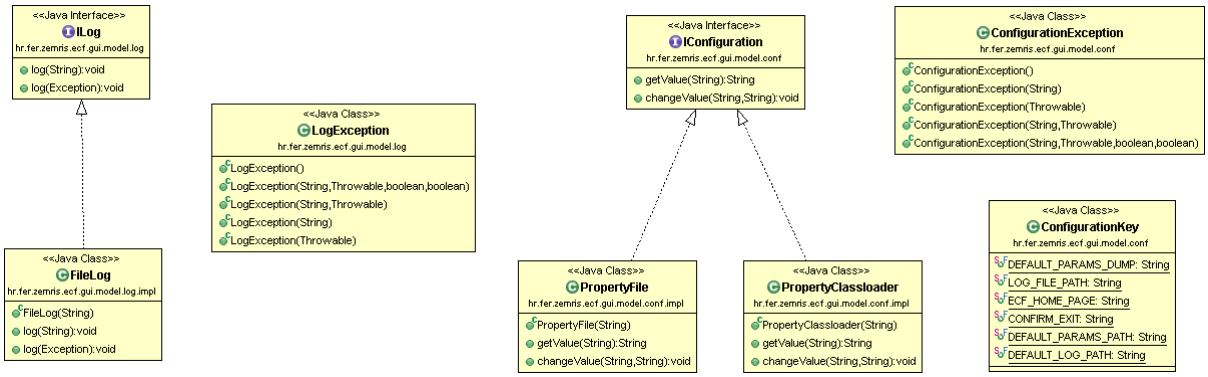
Slika 2.8 prikazuje glavne razrede aplikacije *ECF Lab.* Osnovni je razred razred *ECFLab* koji predstavlja glavni prozor aplikacije. Stvaranjem primjerka razreda *ECFLab* inicijaliziraju se sve potrebne akcije te izbornička i alatna traka. U sredini se nalazi prozor s karticama u kojem je svaka kartica primjerak razreda *ParametersSelection* koji predstavlja jedan pokus. Taj razred prikazuje zaslon za definiranje konfiguracije te pokretanje pokusa. Sastoji se od dva primjerka razreda *EntryBlockSelection* za odabir algoritama

i genotipa, jednog primjerka razreda *EntryListPanel* za odabir dodatnih parametara problema te od primjerka razreda *DefinePanel* koji služi za izbor mjesta pohrane konfiguracije datoteke i datoteke s rezultatima te za odabir broja dretvi. Objekt *EntryBlockSelection* se ustvari sastoji od niza objekata *EntryListPanel* koji su umotani u *DropDownPanel* te od objekta *EntryDisplayList* koji je zadužen za prikaz već dodanih algoritama odnosno genotipa. Objekt *EntryListPanel* sastoji se od niza objekata *EntryFieldPanel* koji predstavljaju parametar odabranog algoritma odnosno genotipa. Objekt *EntryDisplayList* sastoji se od niza objekata *EntryFieldDisplay* koji odgovaraju jednom dodanom unosu. Konkretna implementacija razreda *EntryFieldDisplay* definira način na koji će se prikazati parametri pojedinog unosa. Trenutačno je implementiran razred *FrameDisplay* koji otvara novi prozor u kojem prikazuje sve parametre koje je također moguće i mijenjati.



Slika 2.8 Glavni razredi za prikaz

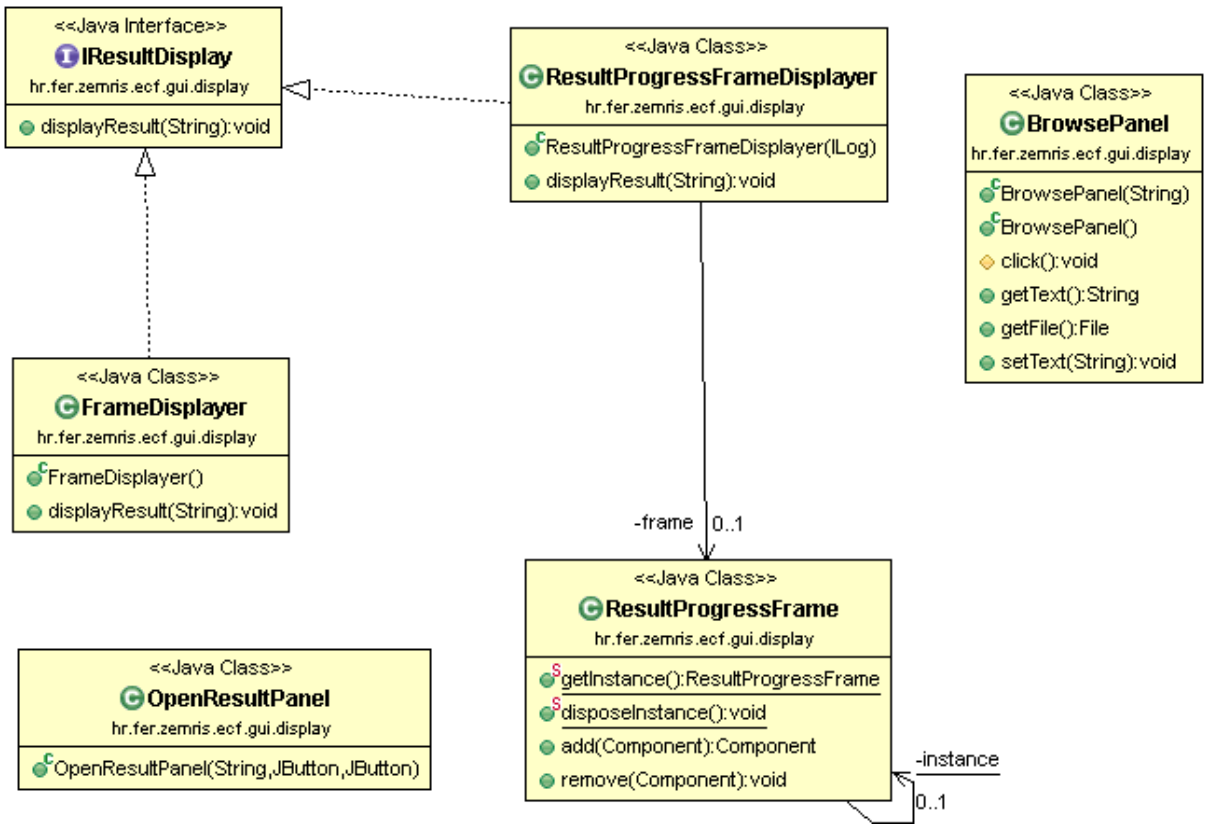
Razred *ECFLab* preko sučelja *IConfiguration* čita početne postavke grafičkog sučelja te mijenja određene stavke ako je to potrebno. Zasad u sklopu korisničkog sučelja nije omogućena komponenta koja bi mijenjala konfiguraciju, ali postoji plan da se to i omogući kroz poseban izbornik postavki. Trenutačno su implementirana dva načina čitanja početnih postavki od kojih je samo jedan u uporabi, a to je razred *PropertyFile* koji čita i piše stavke u *.properties* datoteku. U slučaju da dođe do greške prilikom čitanja ili pisanja postavki, baca se iznimka tipa *ConfigurationException*. Sve stavke koje se mogu dohvatiti navedene su kao konstante u razredu *ConfigurationKey*. U slučaju da tijekom rada aplikacije *ECFLab* dođe do pogreške, te se pogreške zapisuju preko sučelja *ILog*. Trenutačno je implementiran jedan način bilježenja pogreški i to preko razreda *FileLog* koji pogreške zapisuje u tekstualnu datoteku. Ako eventualno dođe do pogreške tijekom bilježenja pogreški, baca se iznimka tipa *LogException.*



Slika 2.9 Razredi vezani za čitanje početnih postavki grafičkog sučelja i za bilježenje pogrešaka

Sučelje *IResultDisplay* definira način na koji je moguće prikazati rezultate izvođenja pokusa. Trenutačno su dostupna dva načina prikaza rezultata, *FrameDisplayer* koji u novom prozoru prikazuje graf dobrote populacije kroz generacije te *ResultProgressFrameDisplayer* koji prikazuje popis datoteka s rezultatima te omogućava njihovo vizualiziranje odnosno

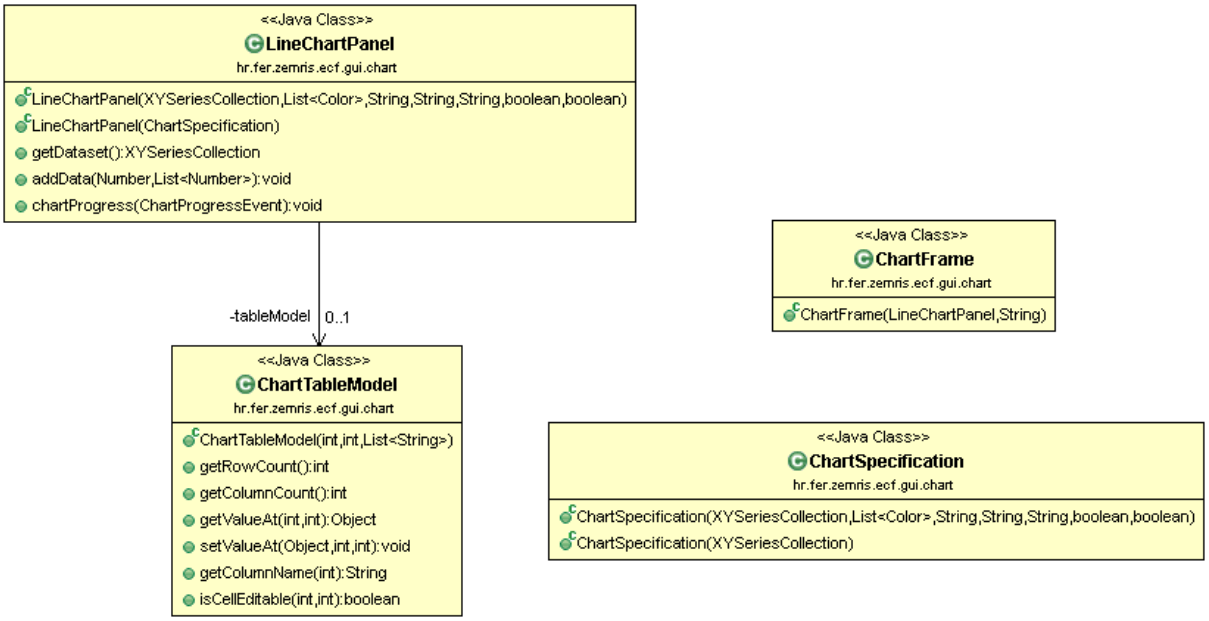
uklanjanje s popisa. Vizualizacija pojedine datoteke ostvaruje se uz pomoć razreda *FrameDisplayer.* Jedna stavka tog popisa oblikovana je razredom *OpenResultPanel,* a sastoji se od prikaza imena datoteke, gumba za vizualizaciju te gumba za uklanjanje s popisa. Prozor u kojem se prikazuje popis datoteka ostvaren je razredom *ResultProgressFrame*, a primjerak tog razreda je jedinstven jer se želi postići da postoji samo jedan takav prozor sa popisom svih rezultata. U ovom paketu nalazi se i pomoćni razred *BrowsePanel* koji definira *panel* za odabir bilo koje datoteke na računalu.



Slika 2.10 Razredi za prikaz rješenja te pomoćni paneli

U paketu *hr.fer.zemris.ecf.gui.chart* nalaze se razredi koji omogućavaju vizualizaciju rezultata u obliku grafa. Za izradu grafa korištena je biblioteka *JFreeChart* [2]. Razred *ChartFrame* predstavlja prozor u kojem se nalazi primjerak razreda *LineChartPanel* koji je zadužen za iscrtavanje

grafa. Taj razred koristi razred *ChartTableModel* za prikaz označenih podataka u tablici. Razred *ChartSpecification* koristi se kao pomoćni razred koji sadržava zadane vrijednosti postavki grafa te olakšava korištenje razreda *LineChartPanel.*



Slika 2.11 Razredi vezani za vizualizaciju rezultata u obliku grafa

2.3 Instalacija aplikacije ECF Lab

Za pokretanje aplikacije potrebno je instalirati Javin virtualni stroj koji se može skinuti sa službene Javine stranice [3]. Aplikacija će biti isporučena u obliku *zip* datoteke koju je potrebno raspakirati i pokrenuti *jar* arhivu „ecflab.jar“ koja će potom pokrenuti *ECF Lab*. Izvorni kod aplikacije javno je dostupan u repozitoriju *Git* na *web* stranici *ECF Lab* projekta [4].

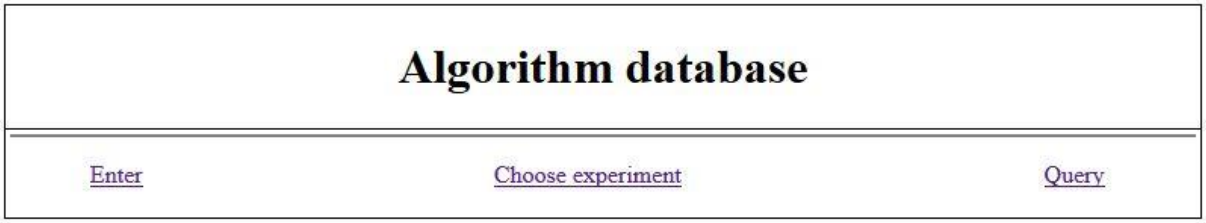
3. Korisničko sučelje za skladište algoritama

Radom ECF-a nastaje veliki broj datoteka s rezultatima izvođenja pokusa od kojih svaka sadrži veliki broj podataka. Kako je postalo nepraktično držati veliki broj velikih tekstualnih datoteka, javila se potreba za prikladnijom organizacijom pohrane rezultata izvedenih pokusa. Logičan izbor bio je izvući te podatke iz tekstualnih datoteka i spremiti ih u bazu podataka. Drugi problem bio je izvlačenje željenih podataka iz tih datoteka. Kako bi se izvukli traženi podaci, morale su se ručno pretraživati sve datoteke i izvlačiti željeni podaci. Iz tih razloga organizirana je baza rezultata pokusa te izrađeno korisničko sučelje za upravljanje bazom na jednostavan način.

Korisničko sučelje za skladište optimizacijskih algoritama *AlgorithmsDB* nastalo je kao nastavak na diplomski rad Karla Kneževića [12] u kojem je bila ostvarena baza i programsko sučelje za unos i dohvat potrebnih podataka. Koristi se baza podataka *H2 [5]* te biblioteka objektno-relacijskog mapiranja *Hibernate [6]. Hibernate* služi za povezivanje razreda s tablicama u bazi podataka za što koristi *JPA [7]* (engl. *Java Persistence API*) anotacije. Korisničko sučelje ostvareno je u obliku *web* aplikacije napisane u programskom jeziku Java.

3.1 Početni zaslon

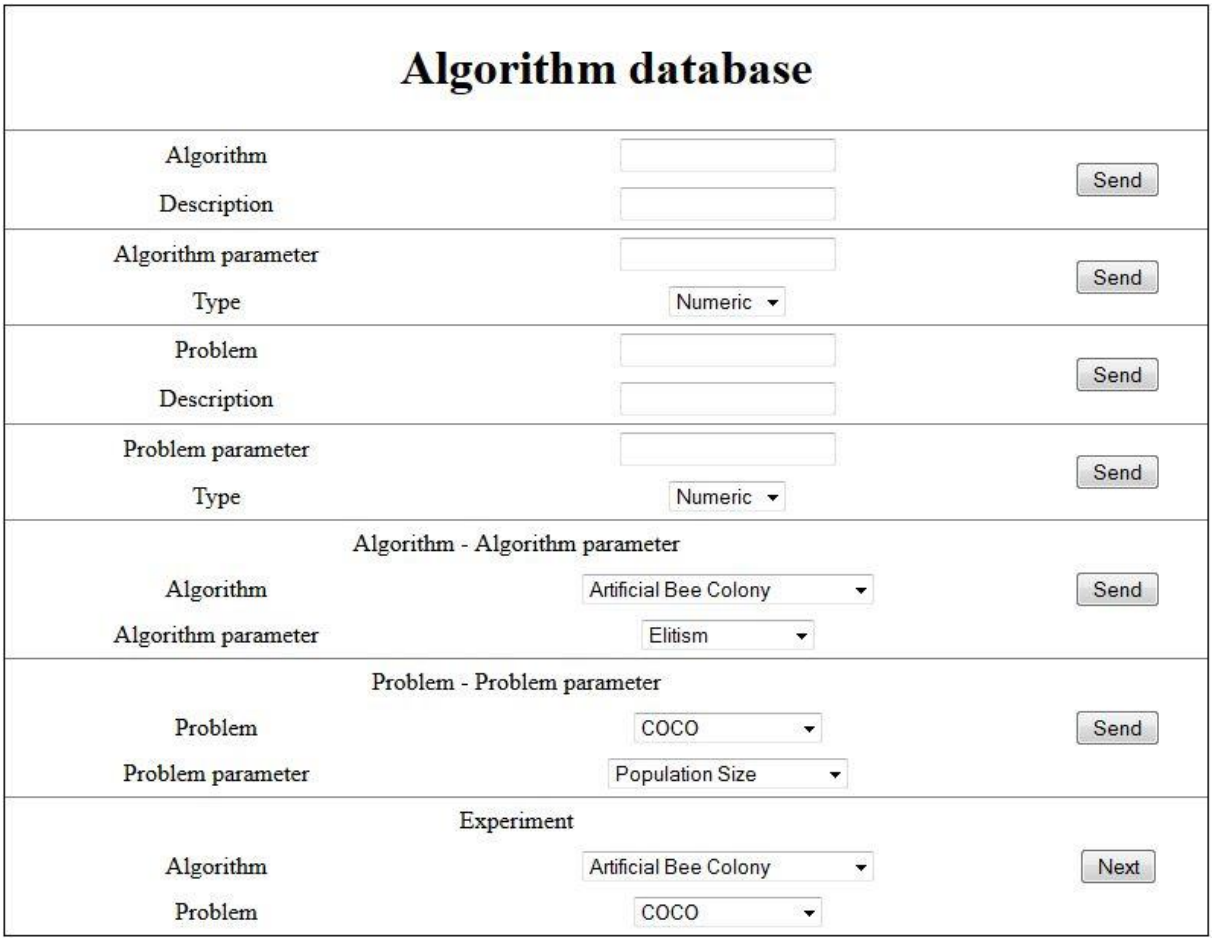
Na početnom zaslonu moguće je odabrati želi li se obaviti unos u bazu ili dohvat podataka iz baze. Dohvat se može obaviti na dva načina: preko korisničkog sučelja za uobičajeni način dohvata podataka (poveznica *Choose experiment*) ili zadavanjem proizvoljnog SQL upita nad bazom (poveznica *Query*). Slika 3.1 prikazuje početni zaslon.



Slika 3.1 Početni zaslon

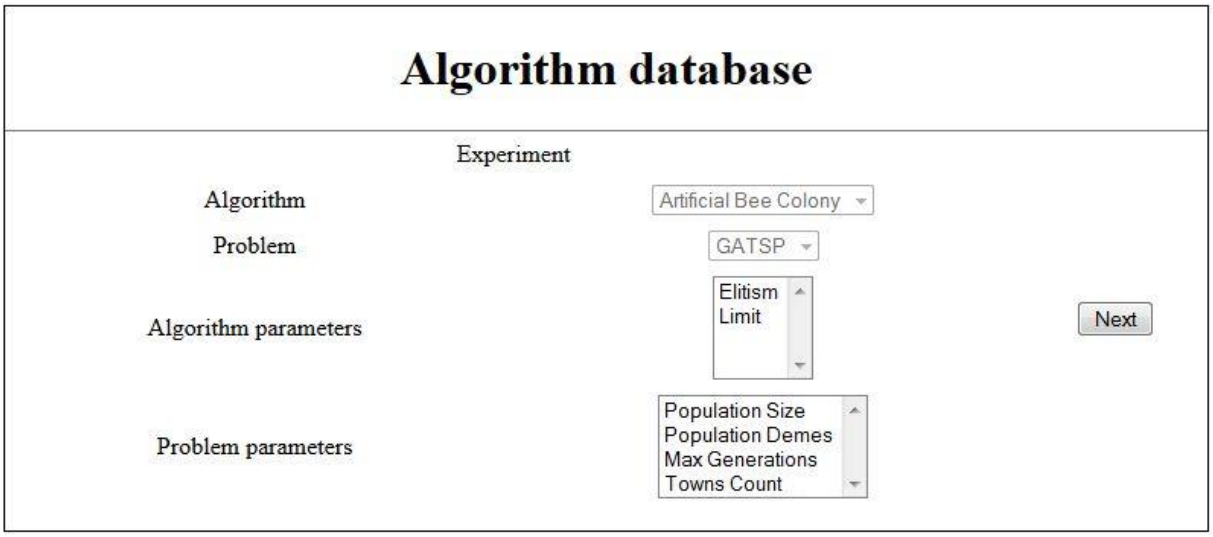
3.2 Dodavanje podataka u bazu

Jedna od funkcionalnosti aplikacije *AlgorithmsDB* je dodavanje novih podataka u bazu. Mogu se dodavati algoritmi, problemi, parametri za algoritme i probleme te pokusi. Kada se unesu algoritmi i parametri za algoritme moguće ih je povezati kako bi aplikacija znala koji parametar se veže za koji algoritam. To također vrijedi i za probleme te problemske parametre. Na slici 3.2 vidi se početni zaslon za unos podataka koji se sastoji od više dijelova. U prvom i trećem redu unose se algoritmi odnosno problemi, a u drugom i četvrtom redu paramteri za algoritme odnosno probleme. U petom i šestom redu povezuju se algoritmi odnosno problemi s odgovarajućim parametrima. U zadnjem retku moguće je unijeti pokus koji je definiran kao veza algoritma i problema. Pritiskom na gumb *Next* odlazi se na sljedeći zaslon na kojemu se definiraju parametri vezani za taj pokus.



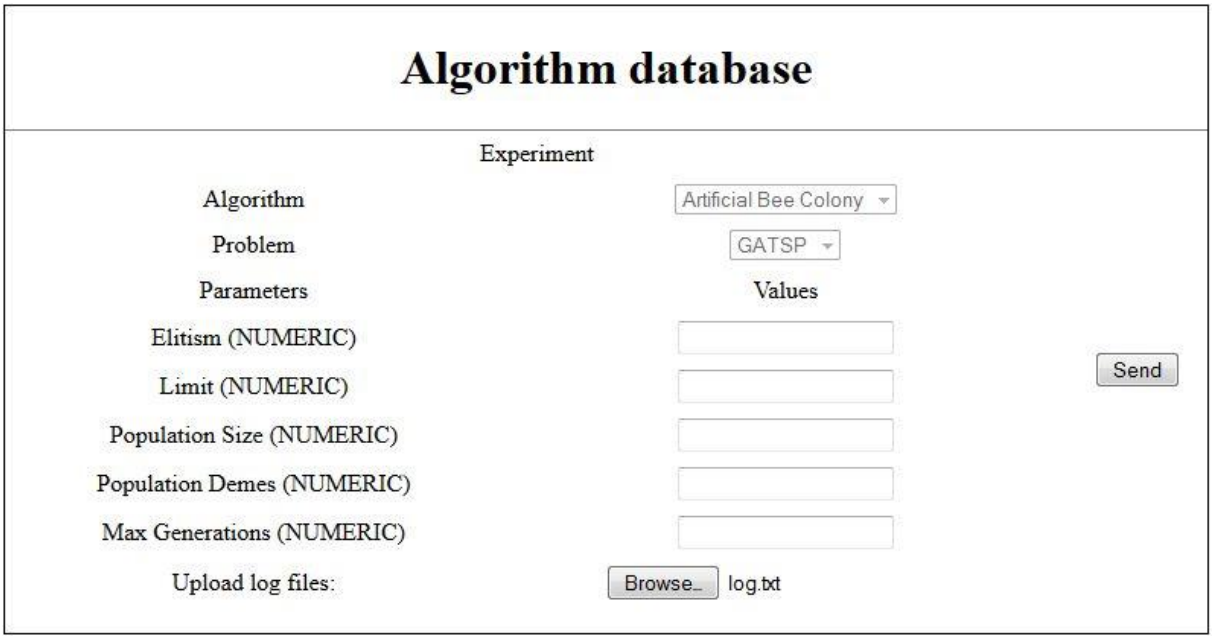
Slika 3.2 Početni zaslon za dodavanje podataka u bazu

Idući zaslon prikazan je na slici 3.3. Na početku su prikazani odabrani algoritam i problem, a ispod je omogućen odabir parametara koji su bili korišteni u pokusu. Moguće je odabrati više algoritamskih parametara i više problemskih parametara. Pritiskom na gumb *Next* prelazi se na idući zaslon koji omogućava unos vrijednosti za pojedini parametar te *upload* datoteka s rezultatima.



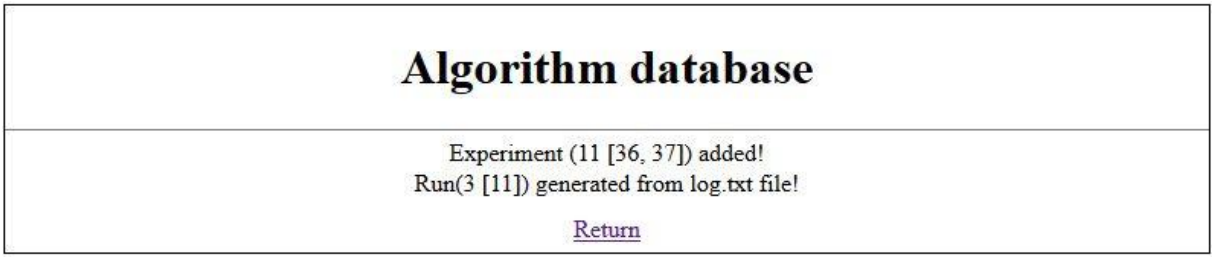
Slika 3.3 Zaslon za odabir parametara vezanih za pokus

Slika 3.4 prikazuje završni zaslon za konačni unos pokusa sa svim potrebnim vrijednostima. Na početku su prikazani odabrani algoritam i problem, a ispod je omogućen unos vrijednosti za svaki parametar odabran na prethodnom zaslonu. Uz ime parametra naveden je tip parametra, brojčani (engl. *Numeric*) ili tekstualni (engl. *String*). Ispod toga omogućen je *upload* datoteka s rezultatima gdje pojedina datoteka odgovara jednom pokretanju pokusa. Pritiskom na gumb *Send* u bazi se stvara novi pokus te se povezuju parametri i njihove vrijednosti s pokusom. Datoteke s rezultatima se parsiraju te se za svaku datoteku odnosno svako pokretanje pokusa stvara niz podataka koji opisuju dobrotu jedinki za određenu iteraciju. Za *upload* datoteka na *web* poslužitelj korištena je biblioteka *Apache Commons IO Fileupload [8]*. Ta biblioteka omogućava parsiranje *HTTP* zahtjeva koji je označen kao *multipart/form-data* što znači da u sebi sadržava više dijelova.



Slika 3.4 Završni zaslon za unos pokusa

Na kraju se pojavljuje zaslon s kratkim informacijama o dodanom pokusu. Uz to se nalazi i poveznica *Return* na početni zaslon. Ovaj se zaslon prikazuje za unos bilo kojeg podatka u bazu, ne samo za unos pokusa.



Slika 3.5 Zaslon s informacijama o dodanom podatku

3.3 Dohvat podataka iz baze

3.3.1 Korisničko sučelje za uobičajeni dohvat podataka

Na slici 3.6 prikazano je sučelje za odabir pokusa. Postoje dva slučaja:

1. Korisnik je odabrao jedan pokus
2. Korisnik je odabrao više od jednog pokusa

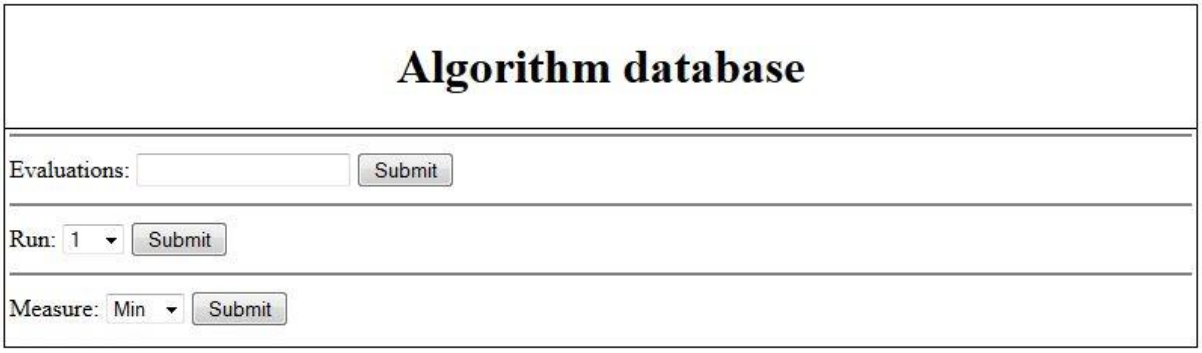
Algorithm database

	Algorithm	Problem	Algorithm parameters	Problem parameters
☐	Artificial Bee Colony	GATSP	Elitism = 3.0 Limit = 4.0	Population Size = 500.0 Population Demes = 4.0 Max Generations = 10000.0
☐	Artificial Bee Colony	GATSP	Elitism = 1.5	Population Size = 300.0 Max Generations = 20000.0
☐	Differential Evolution	GA One Max	CR = 0.9 F = 0.6	Population Size = 400.0
☐	Particle Swarm Optimization	GP Symb Reg	Max Velocity = 20.0 Weight = 10.0	Population Size = 600.0

Submit

Slika 3.6 Odabir pokusa

Slika 3.7 prikazuje zaslon nakon što se odabere jedan pokus. Unosom broja evaluacija prikazuje se ispis zapisa dobrote populacije svih pokretanja odabranog pokusa za određeni broj evaluacija. Odabirom identifikatora pokretanja prikazuje se ispis zapisa dobrote populacije kroz sve evaluacije za zadano pokretanje. Odabirom mjere prikazuje se ispis minimalne, maksimalne i prosječne vrijednosti te medijana vrijednosti odabrane mjere kroz sve evaluacije za sva pokretanja odabranog pokusa. Rezultati dohvata prikazuju se u tekstualnom formatu kao na slici 3.8.

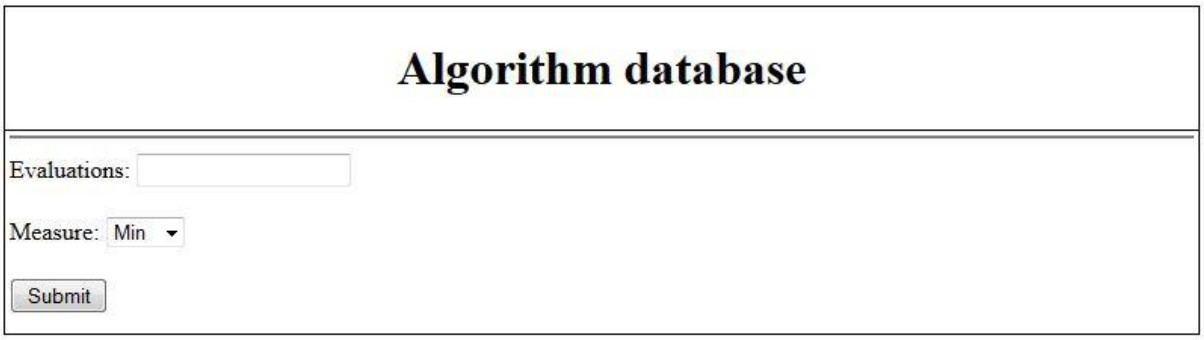


Slika 3.7 Odabir akcija nakon što je odabran jedan pokus

eval	min	max	avg	median
150	2.18147	2.42164	2.3567	2.37855
227	2.00245	2.00245	2.00245	2.00245
229	2.27639	2.27639	2.27639	2.27639
234	2.12868	2.12868	2.12868	2.12868
238	2.21751	2.21751	2.21751	2.21751
241	2.3052	2.3052	2.3052	2.3052
246	2.5178	2.5178	2.5178	2.5178
316	1.79668	1.79668	1.79668	1.79668
320	2.10772	2.10772	2.10772	2.10772
322	1.7735	1.7735	1.7735	1.7735
324	2.08649	2.08649	2.08649	2.08649
327	2.36909	2.36909	2.36909	2.36909
328	2.1851	2.1851	2.1851	2.1851
393	1.91265	1.91265	1.91265	1.91265
399	1.93635	1.93635	1.93635	1.93635
410	2.41904	2.41904	2.41904	2.41904
414	1.65094	1.65094	1.65094	1.65094
415	1.90625	1.90625	1.90625	1.90625
418	1.93641	1.93641	1.93641	1.93641
477	1.91852	1.91852	1.91852	1.91852
479	1.84189	1.84189	1.84189	1.84189
495	2.21988	2.21988	2.21988	2.21988
499	1.73038	1.73038	1.73038	1.73038
500	1.75266	1.75266	1.75266	1.75266

Slika 3.8 Ispis dohvaćenih podataka

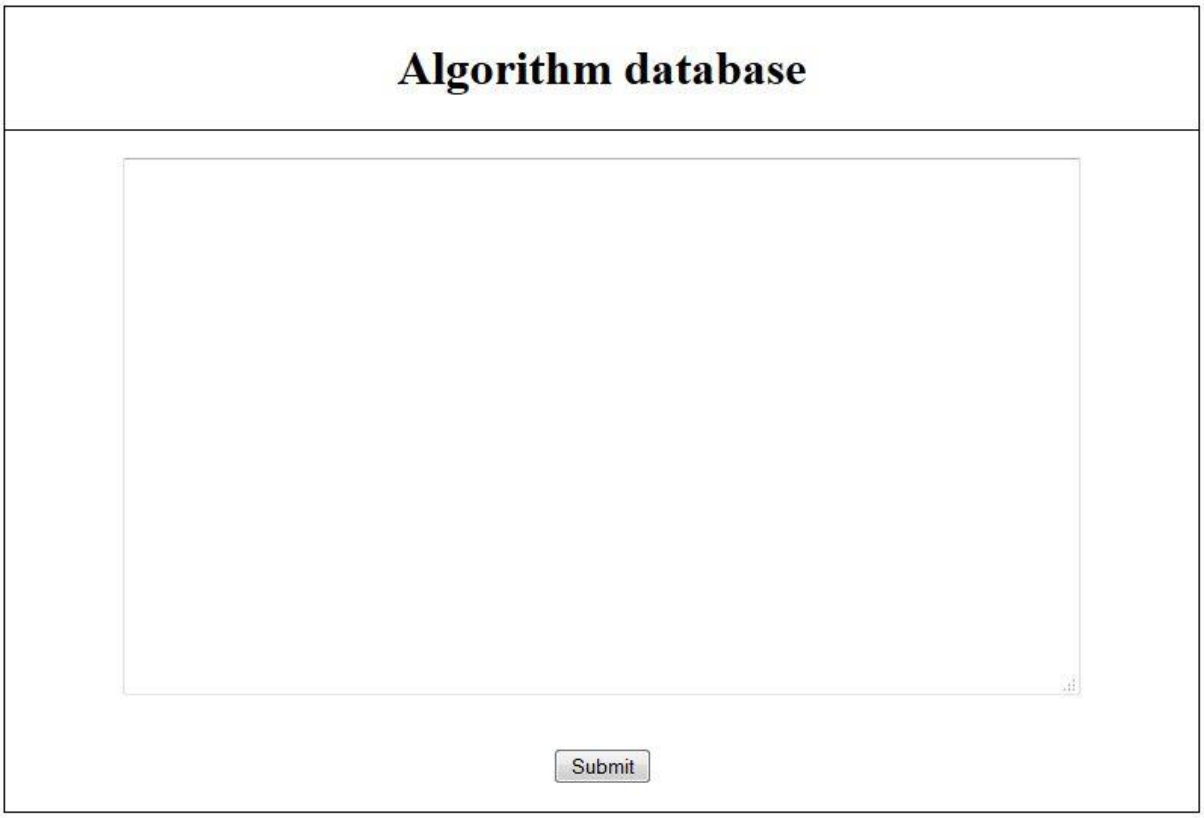
U slučaju da je korisnik odabrao više od jednog pokusa prikazuje se zaslon kao na slici 3.9. Potrebno je upisati željeni broj evaluacija i mjeru nakon čega će se prikazati ispis minimalne, maksimalne i prosječne vrijednosti te medijana vrijednosti odabrane mjere za odabrani broj evaluacija kroz sva pokretanja odabranih pokusa.



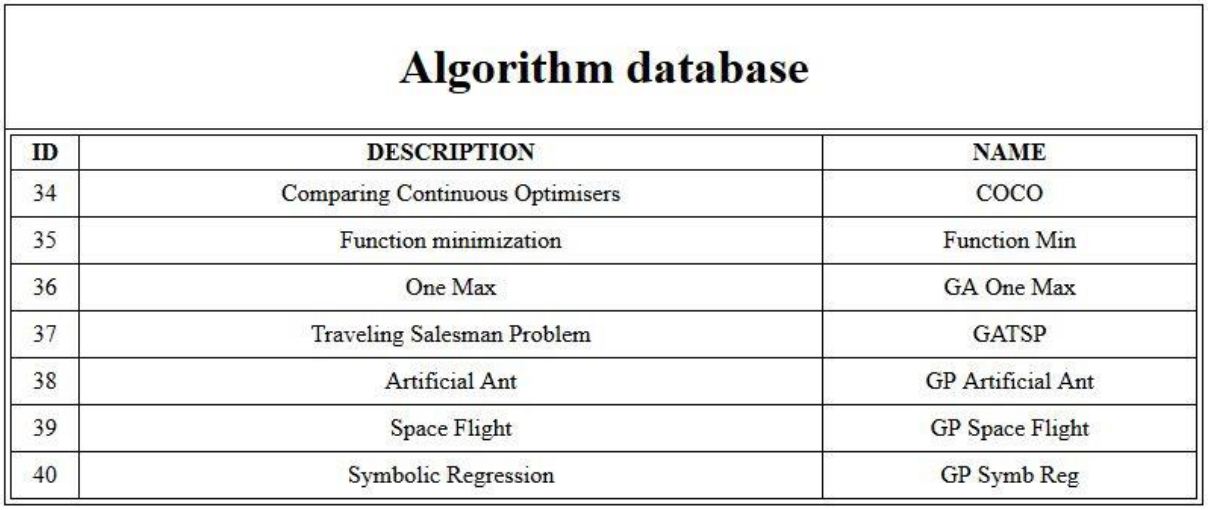
Slika 3.9 Definiranje akcije nakon što je odabrano više pokusa

3.3.2 Dohvat podataka unosom SQL upita

Na slici 3.10 prikazan je zaslon za unos proizvoljnog SQL upita nad bazom. Rezultat dohvata bit će HTML dokument koji će sadržavati tablicu s imenima stupaca te podacima u pojedinom retku kao što je prikazano na slici 3.11. U slučaju da se unese SQL naredba koja nije upit, sustav će dojaviti grešku.



Slika 3.10 Unos SQL upita za dohvat podataka iz baze



Algorithm database

ID	DESCRIPTION	NAME
34	Comparing Continuous Optimisers	COCO
35	Function minimization	Function Min
36	One Max	GA One Max
37	Traveling Salesman Problem	GATSP
38	Artificial Ant	GP Artificial Ant
39	Space Flight	GP Space Flight
40	Symbolic Regression	GP Symb Reg

Slika 3.11 Prikaz rezultata proizvoljnog SQL upita

3.4 Struktura baze rezultata

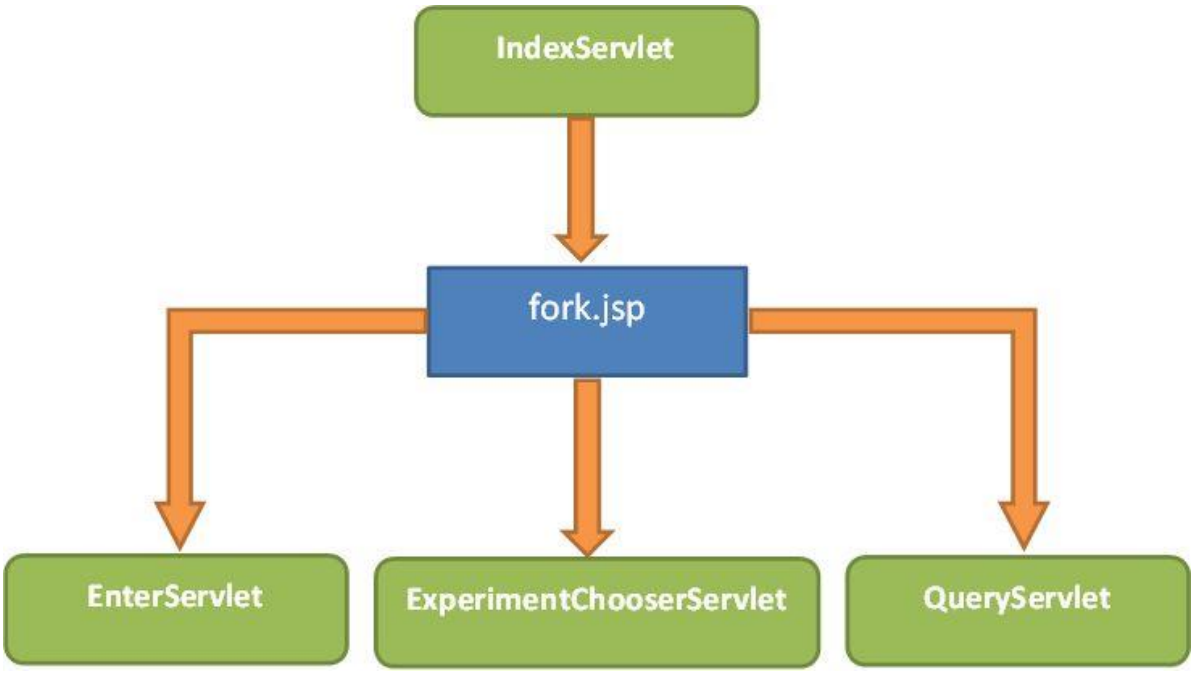
Aplikacija *AlgorithmsDB* podijeljena je u tri sloja:

- **Podatkovni sloj** čini baza podataka koja je preko *Hibernate*-a povezana s aplikacijskim slojem.
- **Aplikacijski sloj** čine *Java Servleti* koji nadograđuju mogućnosti *web* poslužitelja.
- **Prezentacijski sloj** čini *JSP* (engl. *Java Server Pages*) koji na poslužitelju generira *HTML* dokument te ga šalje korisniku.

Raslojavanje aplikacije vizualizirano je na slici 3.12.

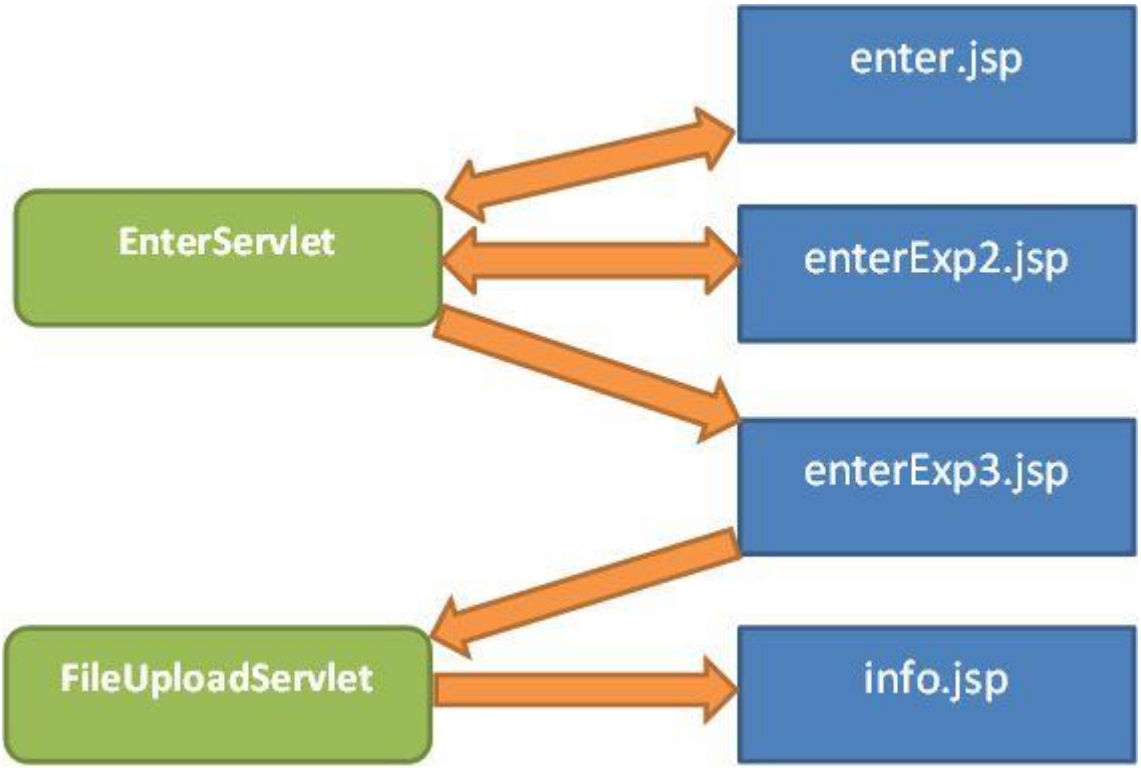
Slika 3.12 Troslojna arhitektura aplikacije

Slika 3.13 prikazuje interakciju *servleta* i *JSP*-a za prikaz početnog zaslona. *IndexServlet* samo preusmjerava prikaz zaslona *fork.jsp*-u koji nudi izbor unosa podataka i dva načina za dohvat podataka.



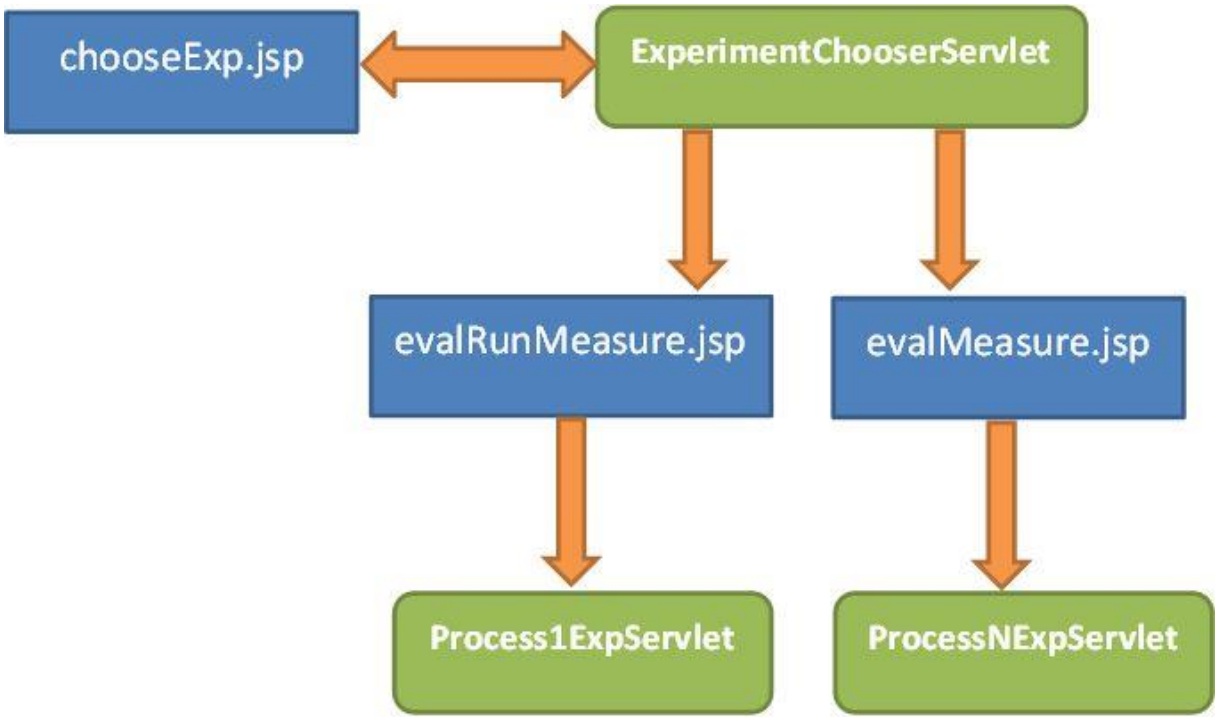
Slika 3.13 Interakcija servleta s JSP-ovima za prikaz početnog zaslona

Slika 3.14 opisuje interakciju između *servleta* i *JSP*-ova pri unosu podataka. *EnterServlet* generiranje *HTML* dokumenta proslijeđuje *enter.jsp*-u koji oblikuje prikaz zaslona za unos podataka. U slučaju da je unesen pokus, *EnterServlet* proslijedit će daljni odabir parametara *enterExp2.jsp*-u. Daljni unos vrijednosti parametara *EnterServlet* će proslijediti *enterExp3.jsp*-u čiji će zahtjev završiti na obradi u *FileUploadServlet*-u. *FileUploadServlet* obavit će unos pokusa sa svim predanim parametrima te će preko *info.jsp*-a prikazati poruku o unesenom pokusu. U slučaju da se nije unosio pokus nego neki drugi podatak, unos tog podatka u bazu obavio bi *EnterServlet*.



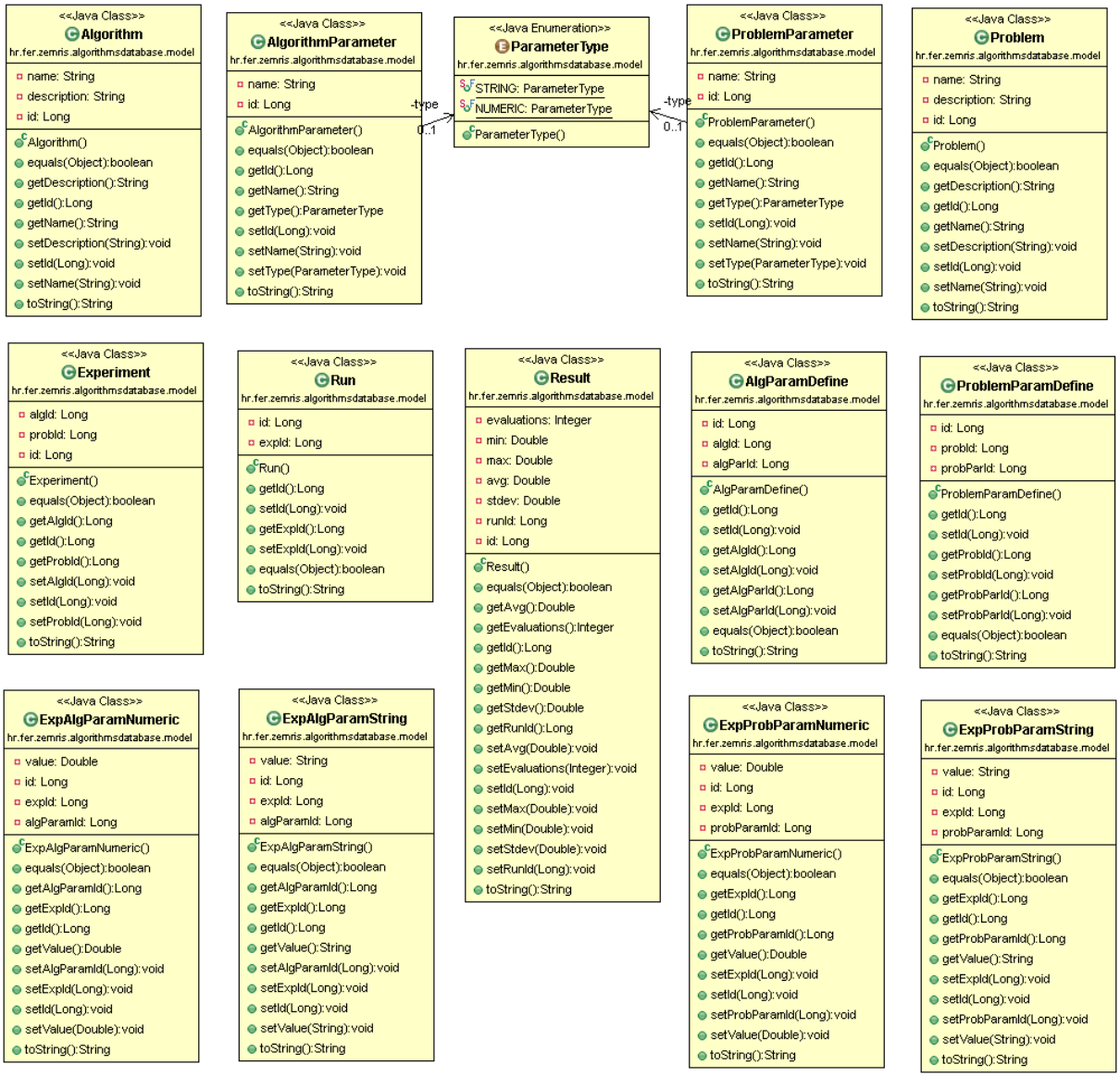
Slika 3.14 Interakcija servleta s JSP-ovima pri unosu podataka

Na slici 3.15 prikazana je interakcija *servleta* s *JSP*-ovima pri dohvatu podataka na uobičajeni način. *ExperimentChooserServlet* i *chooseExp.jsp* služe za odabir pokusa. U slučaju da se odabere jedan pokus, obrada se šalje *evalRunMeasure.jsp*-u odnosno *Process1ExpServlet*-u, a u slučaju da se odabere više od jednog pokusa, obrada se šalje *evalMeasure.jsp*-u odnosno *ProcessNExpServlet*-u.



Slika 3.15 Interakcija servleta s JSP-ovima pri dohvatu podataka na uobičajeni način

Na slici 3.16 nalaze se razredi koji sadrže podatke o rezultatima izvršenih pokusa. Razredi *Algorithm, Problem, AlgorithmParameter* i *ProblemParameter* oblikuju algoritam odnosno problem te pripadajuće parametre. Razredi *AlgParamDefine* i *ProblemParamDefine* povezuju algoritme odnosno probleme s pripadnim parametrima. Razred *Experiment* oblikuje pokus dok razred *Run* oblikuje jedno izvođenje zadanog pokusa. U razredu *Result* nalazi se zapis o stanju dobrote populacije za određeni broj evaluacija. Razredi *ExpAlgParamNumeric, ExpAlgParamString, ExpProbParamNumeric* i *ExpProbParamString* povezuju vrijednosti određenih parametara s odgovarajućim pokusom.



Slika 3.16 Model podataka koji se preslikava u tablice baze podataka

3.5 Instalacija baze rezultata

Za instalaciju baze rezultata na lokalnom računalu potrebno je za početak instalirati Javinu okolinu za izvođenje [9]. Uz to je potrebno još skinuti i bazu podataka H2 [5] te *web* poslužitelj *Apache Tomcat [10]*. Aplikacija će biti isporučena u *war* obliku koji je samo potrebno staviti u *Tomcat*-ov direktorij webapps te pokrenuti *Tomcat.* Također je potrebno

pokrenuti i sustav za upravljanje bazom podataka *H2 Console* koji se pokreće uz *JDBC URL* vrijednosti „*jdbc:h2:tcp:///localhost/~/algdbjpa“*. Trenutačno je omogućen pristup bazi podataka korisniku „sa“ bez lozinke.

4. Zaključak

Metaheuristike nam pomažu kako bismo za teško izračunljive probleme dobili zadovoljavajuće dobra rješenja. Brojni problemi u praksi su netraktabilni pa se uspješno rješavaju primjenom odgovarajućih metaheuristika. Kako mnoge implementacije metaheuristika imaju dosta toga zajedničkog, pokazalo se zgodno izdvojiti neke funkcionalnosti u biblioteke koje će nam olakšati daljnju implementaciju novih problema. Jedna od takvih biblioteka je ECF.

ECF je dosad bio pokretan isključivo iz konzole uz ručno pisanje konfiguracijske datoteke, a rezultati dobiveni radom ECF-a ostajali su u tekstualnom obliku te nisu bili vizualizirani. *ECF Lab* aplikacija je koja rješava te probleme, tj. omogućava jednostavno pokretanje pokusa i vizualizaciju rezultata kroz grafičko korisničko sučelje.

Radom ECF-a dobivaju se ogromne količine tekstualnih datoteka u kojima su zapisani rezultati izvođenja pokusa. Kako je postalo nepraktično držati sve te podatke u takvom obliku, organizirana je baza rezultata koja na efikasan način pohranjuje sve potrebne podatke. Uz to je bilo zgodno imati i grafičko korisničko sučelje kojim ćemo jednostavnije unositi odnosno dohvaćati podatke iz baze. Zato je u okviru ovog završnog rada implementirana *web* aplikacija *AlgorithmsDB* koja ispunjava upravo takve zahtjeve.

5. Literatura

[1] Službena stranica projekta ECF http://gp.zemris.fer.hr/ecf/

[2] Službena stranica projekta JFreeChart http://www.jfree.org/jfreechart/

[3] Službena javina stranica https://www.java.com/en/download/

[4] Stranica *ECF Lab* projekta https://github.com/Truba/Java_gui_4_ECF

[5] H2 http://www.h2database.com/html/main.html

[6] Hibernate http://hibernate.org/

[7] Java Persistence API http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[8] Apache Commons IO Fileupload http://commons.apache.org/proper/commons-fileupload/using.html

[9] Javino okruženje za izvođenje http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html

[10] Apache Tomcat http://tomcat.apache.org/

[11] Čupić, M. Prirodom inspirirani optimizacijski algoritmi. Metaheuristike., 2013.

[12] Knežević, K. Evolucijski algoritmi temeljeni na vjerojatnosnim razdiobama. Diplomski rad. Fakultet elektrothnike i računarstva, Zagreb, 2013.

[13] Kiš Miroslav, Englesko-hrvatski i hrvatsko-engleski informatički rječnik, Zagreb, Naklada Ljevak, 2000., str. 36

6. Sažetak

Ovaj rad opisuje dvije aplikacije koje služe za jednostavnije upravljanje ECF-om odnosno za efikasniju pohranu i dohvat rezultata izvođenja pokusa. Aplikacija *ECF Lab* predstavlja grafičko korisničko sučelje za upravljanje ECF-om. Aplikacija *AlgorithmsDB* predstavlja grafičko korisničko sučelje za upravljanje bazom rezultata.

Ključne riječi: ECF, *Evolutionary Computation Framework,* korisničko sučelje, *ECF Lab*, baza rezultata, *AlgorithmsDB,* metaheuristika, evolucijsko računanje.

7. Summary

This paper describes two applications that are used for easier handling *Evolutionary Computation Framework* and for more efficient storage of results generated by ECF. *ECF Lab* is a desktop application that provides graphical user interface for simple managing the ECF. Web application *AlgorithmsDB* represents graphical user interface for managing results database.

Keywords: ECF, *Evolutionary Computation Framework,* user interface, *ECF Lab*, results database, *AlgorithmsDB,* metaheuristics, evolutionary computation.