SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Domagoj Kusalić

Evolucijski algoritmi inspirirani ljudskim psihosocijalnim ponašanjem

Zagreb, 2010.

Ovaj rad izrađen je u Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu pod vodstvom Doc. dr. sc. Marina Goluba i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2009./2010.

Sadržaj

1. Uvod

1.1. Opis rada

U okviru ovog rada osmišljeni su i ispitani novi evolucijski algoritmi inspirirani ljudskim psihosocijalnim ponašanjem.

U prvom poglavlju dan je osnovni uvod u područje koje rad izučava. U drugom poglavlju je osmišljen generalizirani genetski algoritam. Poglavlje počinje uvodom u genetski algoritam, zatim definira dobrotu jedinke i korištene operatore, da bi na kraju bio definiran i konačan algoritam. U trećem poglavlju razmatra se ostvareni programski sustav koji ostvaruje i ispituje ponašanje algoritma iznesenog u drugom poglavlju. Detaljnije se razrađuje implementacija, navode se primjeri i analiza izvođenja, te se algoritam uspoređuje sa genetskim algoritmom. Na kraju trećeg poglavlja dan je "no free lunch" teorem. Četvrto poglavlje objašnjava paralelnu izvedbu algoritma osmišljenog u drugom poglavlju, te donosi dodatna razmatranja u vezi paralelne izvedbe genetskih algoritama. U petom poglavlju definira se osmišljeni pomoćni algoritam osobnog prostora. U šestom poglavlju osmišljen je algoritam optimizacije procesiranja misli. Poglavlje počinje motivacijom, te je poslije definiran algoritam. Kasnije u poglavlju je objašnjen ostvareni programski sustav koji ostvaruje i ispituje osmišljeni algoritam. U sedmom poglavlju su navedena dodatna razmišljanja koja daju smjernice za daljnja istraživanja. Osmo poglavlje donosi zaključak.

1.2. O evolucijskim algoritmima

Kroz milijune godina različite vrste se razvijaju kako bi što bolje preživjele u okolišu koji ih okružuje. Razvijene prilagodbe se prenose unutar vrste kroz generacije, te se proces prilagodbe nastavlja. Prilagodbe su većinom takve da

omogućuju jedinki bolje šanse za preživljavanje u svom okolišu. Promjene koje su štetne postepeno iščezavaju, dok promjene koje su korisne imaju veće šanse za očuvanjem jer osiguravaju preživljavanje jedinke koja ih posjeduje. Na taj način priroda uspijeva razviti sve naprednije organizme koji su sve sposobniji preživljavanju u okolišu koji ih okružuje.

Inspirirana procesom evolucije, znanost je razvili računalnu paradigmu zvanu evolucijsko računarstvo (engl. *evolutionary computation*) koja oponašanjem prirodnih evolucijskih procesa pokušava riješiti razne procesorski zahtjevne računalne probleme. U području umjetne inteligencije, evolucijski algoritmi[5] predstavljaju podskup evolucijskog računarstva[21]. Evolucijski algoritmi koriste neke mehanizme inspirirane biološkom evolucijom: selekcija, reprodukcija, mutacija i rekombinacija, kako bi pronašli što bolje rješenje određenog problema. Potencijalna rješenja optimizacijskog problema kojeg evolucijski algoritam pokušava riješiti igraju ulogu "jedinke" u "populaciji". Pritom se potencijalna rješenje (jedinke) međusobno kombiniraju i razvijaju na sličan način kao što to u prirodi čine biološke jedinke neke vrste. Najpoznatiji primjer evolucijskih algoritama predstavlja genetski algoritam.

Evolucijski algoritmi često dobro aproksimiraju rješenja bilo kojeg problema jer nemaju ugrađene pretpostavke o izgledu domene pretrage nad kojim se provode. To opće svojstvo čini evolucijske algoritme korisnim u raznovrsnim područjima poput: umjetnosti, ekonomije, marketinga, robotike, sociologije, fizike, politike i kemije.[20] Bitne komponente evolucijskih algoritama čine: genetski algoritmi, genetsko programiranje i evolucijske strategije.[15]

Osim evolucijskih algoritma, postoje i mnoge druge popularne (i vrlo primjenjive) tehnike evolucijskog računarstva koje oponašaju postupke koji se mogu zapaziti u prirodi. Npr. oponašanju kretanja mrava ili ptica. Treba napomenuti da su sve te tehnike nastale usavršavanjem pojedine biološke vrste, te da su se sve te biološke vrste razvile korištenjem osnovnog alata majke prirode: *preživljavanje najsposobnijih.*

1.3. Hipoteze

Promotrimo ukratko postojeće popularne tehnike inspirirane prirodnim procesima i ponašanjima pojedinih bioloških vrsta, kako bi poslije mogli napraviti korak dalje u razvijanju novih evolucijskih algoritama.

1.3.1. Postojeći algoritmi

Istraživanja ponašanja raznih vrsta u prirodi rezultirala su raznim algoritmima koji simulirajući prirodu uspijevaju pronaći dobra rješenje određenih računalnih problema. Pritom se većina postupaka temelji na ponašanjima sociološki aktivnih životinjskih vrsta. Sociobiologija[2] je dala najveći doprinos razvitku takvih algoritama.

Sociobiologija je kombinacija znanstvenih disciplina koje pokušavaju objasniti socijalno ponašanje određenih životinjskih vrsta uzimajući u obzir Darvinističke prednosti pojedinog ponašanja. Sociobiologija je kod ljudi usko vezana sa evolucijskom psihologijom.

Socijalna ponašanja su izrazito primjetna kod socijalnih insekata poput mrava ili ptica. Pojedinačni mravi se podređuju dobrobiti zajednice, tako da zajedno ostvaruju vrlo složene sustave. Algoritam optimizacije kolonije mrava[17] (engl. *ant colony optimization*) oponaša način na koje se mravi kreću kroz prostor kako bi pronašli što kraći put od mravinjaka do izvora hrane. Ptice se prilikom traženja hrane kreću u jatima budući da "više očiju bolje vidi", te tako lakše nalaze hranu (algoritam particle swarm optimization[19]). Postoje i mnogi drugi algoritmi grupne inteligencije (engl. *swarm intelligence*) u kojima grupa jedinki socijalnim međudjelovanjem uspješno rješava određeni optimizacijski problem.

Većina korisnih socijalnih ponašanja raznih životinjskih vrsta razvila su se kroz evoluciju, prenošenjem (preživljavanjem) dobrih gena kroz generacije jedinki. Evolucija sa svojim operatorima (selekcija, križanje, mutacija) djeluje na pojedinačnu jedinku, međutim, evolucija je sposobna razviti mehanizme koji ne pomažu jedinki da osigura opstanak samo sebi, nego i drugim jedinkama oko nje, usprkos tome što se tada i druge jedinke moraju ponašati socijalno.

1.3.2. Kako dalje?

Do sada napravljena istraživanja o ponašanju procesa iz prirode su se uglavnom bavila načinom prenošenja genetskog materijala među jedinkama, te istraživanjem ponašanja socijalnih insekata i nekih životinja. Postavlja se pitanje "kako dalje?".

Treba razmišljati drukčije da bismo uspjeli u centar istraživanja postaviti *sebe*. Prirodna evolucija je vrlo dobra i uspješna u tome što čini, te je stvorila čovjeka i životinje. No, nije li čovjek mnogo više od životinje? Nije li čovjek mnogo napredniji i uspješniji u tome što radi, nego životinje? Na kraju, nije li čovjek vrh evolucije?

Očito je čovjek odskočio od ostalih produkata evolucije. Ta razlika čini *nâs* onima koje treba proučavati kako bi pronašli i uspjeli računalom oponašati ono najbolje iz prirode. Samo oponašanje procesa evolucije (genetski algoritam) nije dovoljno kako bi postigli najbolje, jer su tim procesima bile podložene i životinje, a nisu se uspjele približiti čovjekovom razvitku. Treba detaljnije proučiti čovjekovo ponašanje i način na koji čovjek provodi evoluciju izuzev onoga što je zajedničko sa životinjama.

Dodatan naglasak treba staviti na ono u čemu je čovjek toliko bolji od svih ostalih životinja. Čovjekova najveća prednost nad drugim životinja je ljudski *razum*. Unatoč tome što je razum i način na koji on logički funkcionira ljudima često poprilično nejasno, treba u njemu tražiti postupke koji dovode do optimalnijih ponašanja. Ipak je to glavna razlika koja ljude kao vrstu čini toliko superiornijom od svih drugih vrsta.

Svaki evolucijski algoritam u načelu čini istu stvar: pokušava usmjeriti pretragu u povoljnom smjeru skupljajući informacije o stanju procesa pretrage. Pritom algoritam može biti bolji ukoliko raspolaže sa više informacija. Dakle, prilikom dizajniranju novog algoritma, cilj je vrlo jednostavan:

Maksimizirati količinu informacija o stanju procesa pretrage!

Pritom je potrebno filtrirati dobivene informacije, te zanemariti one koje su redundantne.

Korisne ljudske postupke može se pronaći analizirajući ljudsko socijalno pona-

šanje u okviru socijologije[6], dok se ljudske kognitivne procese može analizirati od strane psihologije[31]. Da bi se došlo do korisnih ljudskih postupaka mogu se provoditi zamišljeni eksperimenti s ljudima, koji se često svode na opažanje vlastitog ponašanja:

- Kako biramo mjesto gdje sjedamo u autobus?
- Krećemo li se pravocrtno kroz prostor kad nam se žuri?
- Na koji način biramo voće u samoposluzi?
- Na koji način odabiremo partnere?
- Kako odlučujemo koji automobil kupiti?
- Kako se dosjećamo?

Odgovori na neka od ovih pitanja potaknuti će rasprave koje će voditi do korisnih informacija o načinu kako ljudi optimiraju probleme s kojima se susreću...

1.4. Cilj rada

Cilj rada je razviti nove evolucijske algoritama zasnovane na ljudskom psihosocijalnom ponašanju. Treba detaljno istražiti ljudska ponašanja, primijetiti ona koja se pokazuju korisnima, te osmisliti evolucijske algoritme koji ih primjenjuju. Također je potrebno ispitati bitnije predložene algoritme, te predložiti daljnja istraživanja algoritama osnovanih na primijećenim zanimljivim ljudskim ponašanjima.

2. Generalizirani genetski algoritam

2.1. Uvod u genetske algoritme

Genetski algoritmi[7] su klasa računalnih metoda koje oponašaju principe prirodne evolucije za potrebe pretrage i optimizacije. Genetski algoritmi koriste dvije bitne karakteristike prirodne evolucije:

- prenošenje informacija iz jedne generacije u drugu (nasljeđivanje) i
- nadmetanje za preživljavanje (preživljavanje najsposobnijih).

Glavne prednosti genetskih algoritama, koje ih čine podobnima za rješavanje problema iz stvarnog života su:

- Prilagodljivi su.
- Učinkoviti su u rješavanju kompleksnih problema u kojima je glavni cilj brzo naći dobro, no ne nužno i optimalno rješenje.

Posebno su podobni za aplikacije koje zahtijevaju pretragu i optimizaciju kada je prostor pretrage izrazito velik i kompleksan, te nema potrebe naći baš najbolje rješenje. Najviše se primjenjuju za razlikovanje uzoraka, procesiranje slika, prepoznavanje scena, dizajniranje neuronskih mreža, problem rasporeda, planiranje puta, optimizaciju problema trgovačkog putnika, bojanje grafova, numeričku optimizaciju, te mnoge druge probleme.

Genetski algoritmi (kraće GA) modeliraju principe prirodnog genetskog sustava, pritom genetski materijal pojedine jedinke (potencijalnog rješenja) biva zapisan u strukturu zvanu "kromosom". GA koriste određeno znanje o prirodi problema kako bi oblikovali funkciju dobrote, pomoću koje se pretraga usmjeruje u više

obećavajuća područja. Svaka jedinka (jedinku čini kromosom) ima pridruženu vrijednost funkcije dobrote koja ocjenjuje koliko je jedinka dobra u odnosu na rješenje koje predstavlja. Grupa jedinki se naziva "populacija". Različiti biološki inspirirani operatori poput selekcije, križanja i mutacije bivaju primijenjeni na kromosome iz populacije, kako bi stvorili potencijalno bolje rješenje danog problema. GA se provodi sljedećim koracima:[15]

1. Inicijalizira se populacija tako što se stvori određen broj jedinki sa slučajno popunjenim kromosomima (jedinke predstavljaju slučajno rješenje danog problema).

2. Procjenjuje se dobrota pojedine jedinke (kvaliteta pojedinog rješenja koje jedinka predstavlja) pomoću funkcije dobrote.

3. Selektiraju se jedinke koje se ne eliminiraju (provodi se selekcija).

4. Nad selektiranim jedinkama se provode genetski operatori (križanje i mutacija).

5. Novonastale i/ili promijenjene jedinke tvore novu populaciju.

6. Prekida se algoritam ili nastavlja od 2. koraka.

U 2. koraku se svaka jedinka procjenjuje funkcijom dobrote koja brojčanom vrijednošću pokušava opisati sposobnost pojedine jedinke (potencijalnog rješenja) da riješi dani problem.

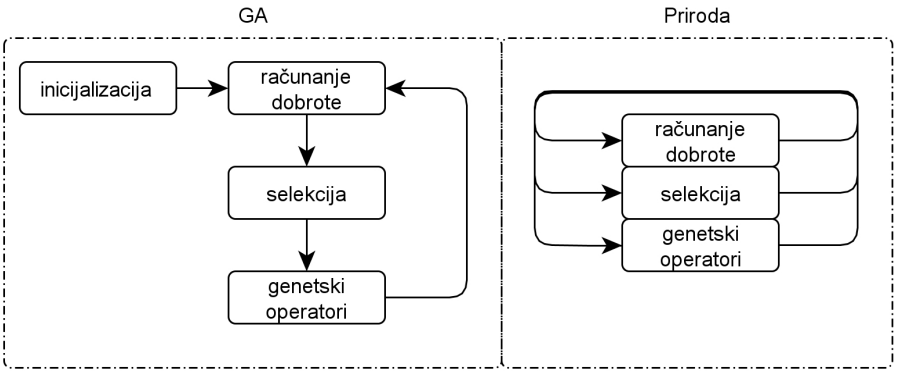
U 3. koraku se s većom vjerojatnošću odabiru bolje jedinke (preživljavanje najsposobnijih), kako bi sljedeća populacija imala bolju prosječnu dobrotu jedinki (napredak algoritma).

U 4. koraku se selektirane (preživjele) jedinke međusobno križaju i mutiraju kako bi stvorile nove jedinke.

Operatori križanja i mutacije se u genetskom algoritmu provode na način sličan kao u prirodi. Mutacija se provodi tako što se napravi manja promjena u strukturi koja zapisuje potencijalno rješenje problema (kromosom), kao što se u prirodi napravi manja mutacija nad genima jedinke. Pri križanju se pomoću dvije (ili više) jedinke (zvane "roditelji") stvara nova jedinka (zvana "dijete") čija

struktura podatka (koja opisuje potencijalno rješenje danog problema) nastaje kombiniranjem "roditeljskih" struktura podataka. Na sličan način se u prirodi geni potomaka stvaraju kombiniranjem gena roditelja.

Navedeni koraci koji opisuju standardan postupak provedbe genetskog algoritma poprilično odstupaju od načina na koji se oni provodi u prirodi. U prirodi se 2., 3. i 4. korak nad populacijom provode paralelno, bez primjetne granice među njima (slika 2.1). Također, veličina populacije nije fiksna, nego ovisi o sposobnostima jedinki prilikom preživljavanja.



Slika 2.1: Provođenje evolucije u genetskom algoritmu i u prirodi

Navedeni postupak provedbe genetskog algoritma se u prirodi provodi na sličan način gleda li se životni ciklus pojedine jedinke, no nikako ne nad cijelom populacijom.

Unatoč dobrim rezultatima koje daju genetski algoritmi pri rješavanje raznih problema, genetski algoritam ne može efektivno riješiti probleme u kojima funkcija dobrote raspolaže samo sa informacijama točno/netočno (npr. problemi odluke). U takvim problemima genetski algoritam ne može konvergirati prema rješenju (no hill to climb), nijedan algoritam ne može. Tada slučajna pretraga može pokazati jednako dobre rezultate kao i genetski algoritam.

Sličan problem, poznat pod nazivom "induktivna pristranosti", se može susresti u području strojnog učenja. Radi se o nemogućnosti generalizacije bez pretpostavke o obliku hipoteze, dobivamo rote-learner.

U nastavku rada bit će razmotreni načini popravka i nadogradnje genetskog algoritma. Biti će razvijen algoritam koji je toliko različit od genetskog algoritma,

da ga je pogrešno zvati genetskim algoritmom. Nazovimo razvijeni algoritam: "generalizirani genetski algoritam" (engl. *generalized genetic algorithm*).

2.2. Dobrota jedinke

Pojam dobrote jedinke se odnosi na sposobnost jedinke pri rješavanju problema za koji je algoritam osmišljen. Prilikom procjene dobrote jedinke koristi se funkcija koja za pojedinu jedinku daje brojčanu vrijednost koja opisuje sposobnost jedinke prilikom rješavanja zadanog problema. (Ukoliko nije moguće definirati funkciju dobrote, moguće je turnirskom selekcijom[7] raspoznavati bolje jedinke od gorih. U tom je slučaju moguće pratiti npr. vjerojatnost pobjede jedinke. U ovom radu neće biti razmatrani problemi kojima nije moguće definirati funkciju dobrote.)

Promotrimo prvo kako odrediti dobrotu jedinke, tako da izbacimo redundantne informacije, te sačuvamo samo one informacije koje su sigurno korisne. Pritom je dobro proučiti načine na koje ljudi procjenjuju dobrotu stvari oko sebe.

2.2.1. Ljudsko poimanje dobrote

Ljudskim poimanjem svijeta oko nas bavi se grana filozofije pod nazivom epistemologija [18]. Ljudska spoznaja je relativna, jer su sve informacije u relativnim odnosima. Ljudi utvrđuju da je nešto veliko u odnosu na nešto malo, ili da je nešto bolje u odnosu na nešto prosječno. Osoba koja je gluha od rođenja teško shvaća pojam "tiho" jer nije upoznala pojam "glasno". Kada se nikad ne bi upoznali sa pojmom "toplo", teško bi shvatili pojam "hladno"; zato većina takvih pridjeva dolaze u parovima.

Problem sa genetskim algoritmom (ili nekim drugim evolucijskim algoritmom) je u tome što je informacija o dobroti jedinke dana kao brojčana vrijednost. Kad bi nam netko rekao da mu je određeni automobil lijep 15, malo bi nam to značilo, budući da ne znamo je li to "dobra" ili "loša" vrijednost. Ukoliko nam netko kaže da mu je automobil A lijep 15, a automobil B lijep 17, jedino što znamo jest da je neki automobil ljepši od drugoga. Ukoliko raspolažemo dodatnom informacijom da veći broj predstavlja ljepši automobil, tada znamo da je automobil B ljepši. (Kad bi brojevima procijenili miris nekih predmeta, teško da bi osoba mogla pretpostaviti da je više ujedno bolje, jer ne zna procjenjujemo li

ugodan ili neugodan miris.) Kada znamo da je B ljepši od A, trebamo informaciju o tome koliko je B ljepši od A. Očito je B za 2 ljepši, ali taj broj 2 nam ne govori mnogo. Ovisno o tome daje li procjenitelj brojeve iz intervala $[0, 20]$ ili $[-100, 100]$ možemo steći bolji dojam o tome koliko je B ljepši od A. Kada nam je poznata informacija o intervalu iz kojeg su odabrane vrijednost procjene, i dalje ne možemo dobiti potpun dojam o tome koliko je B ljepši od A, ukoliko ne znamo razdiobu kojom procjenitelj dodjeljuje procjene. Ukoliko znamo da su procijene iz intervala $[0, 20]$, te znamo da je prosječna procjena 4, te da je za 90% automobila procjena iz intervala $[2, 8]$, i dalje nam trebaju ostali intervali povjerenja[12] kako bi dobili potpuni dojam. Na kraju, treba nam matematički zapis razdiobe kojom se procjenitelj služi.

Zbog navedenih problema sa brojčanim procjenama, ljudi koriste procjene poput: "lijep", "jako lijep", "neusporedivo ljepši", i slično.

Ljudi se, kao vrh evolucije, uglavnom ponašaju optimalnije od ostalih životinjskih vrsta, te je stoga njihovo ponašanje vrijedno dodatne pažnje. Ljudi su pritom najbolji u radnjama koje često obavljaju, te u onima radnjama o kojima im ovisi opstanak. Zanimljivo je primijetiti da ljudi ponekad spontano oponašaju neke sofisticirane evolucijske tehnike.

Promotrimo način na koji ljudi barataju populacijom jedinki... Vjerojatno ste nekad promatrali način na koji neka starica bira voće koje želi kupiti u samoposluzi ili na tržnici. Osoba koja odabire voće zapravo radi selekciju nad populacijom koja se sastoji od jedinki voća. Osoba čini procjenu dobrote pojedinog ploda, te pokušava maksimizirati kvalitetu odabranog voća za danu količinu novaca. U početnim društvenim zajednicama postojala je određena podjela poslova koja je osiguravala opstanak zajednice. U takvim zajednicama su žene, među ostalom, skupljale plodove voća.[3] Prije je kvaliteta odabranog voća utjecala na preživljavanje, a danas utječe na financijske izdatke. Očito ljudi sličnu radnju ponavljaju već tisućama godina, te su je poprilično usavršili.

Osoba koja provodi selekciju plodova voća nema u mislima sliku savršenog ploda koji traži, niti zna kako izgleda "jediničan" plod, ali zato pred sobom lako uočava izgled prosječnog ploda, te i bez znanja statistike spontano uočava standardno odstupanje plodova od prosječnog ploda, pretpostavivši da su radi o nor-

malnoj razdiobi. Time osoba raspolaže sa dovoljno informacija za procjenu koliko je neka jedinka dobra ili loša. Treba primijetiti da osoba pri selekciji plodova ne provodi selekciju nad cijelom populacijom odjednom, nego održava veličinu populacije odabranih plodova relativno sličnom početnoj veličini populacije, te dodaje pojedine plodove u populaciju (sa većom vjerojatnošću bira bolje) i izbacuje pojedine plodove (sa većom vjerojatnošću bira gore). Zapravo se radi o ponavljanje sljedeća dva jednostavna koraka:

- Dodaj dobru jedinku u populaciju odabranih.
- Podesi veličinu populacije (ne čini ništa ili izbaci nekoliko jedinki).

Pritom se veličina populacije smanjuje u odnosu na željenu ukoliko su odabrane jedinke relativno različite dobrote (veće standardno odstupanje), a povećava ukoliko su jedinke međusobno slične dobrote (pronađen je neki maksimum funkcije dobrote). Posljedica je u tome da osoba često na kraju kupi veću količinu plodova nego je planirala na početku.

Sličan obrazac ponašanja se može vidjeti kod djece koja skupljaju lijepe kamenčiće na plaži. Npr. kad dijete nađu znatno bolji kamenčić od prosjeka dotad pronađenih, odbacuju veći dio populacije. Često su mala djeca previše mlada da bi do tada stigla naučiti opisani obrazac ponašanja, što podupire ideju da je opisano ponašanje preneseno evolucijom.

2.2.2. Veličina populacije

Prilikom razmatranja dobrote jedinki, potrebno je na određen način normalizirati dobrotu jedinki kako bi tako dobivene vrijednosti dobrote bila korisna. Kasnije se na temelju normaliziranih vrijednosti dobrote može odrediti veličina populacije (kao što to rade ljudi pri kupnji voća).

Pretpostavimo da je na raspolaganju funkcija koja prima jedinku, te vraća broj koji predstavlja njenu dobrotu:

$$x = d(jedinka). \tag{2.1}$$

Jednadžba (2.1) definira standardnu funkciju kakvu smo prisiljeni koristiti pri rješavanju problema genetskim algoritmom. Ne posjedujemo nikakvu dodatnu informaciju o tome što funkcija vraća, osim da je više ujedno i bolje (prisjetite

se primjera sa ljepotom automobila). Osoba pri kupnji voća raspolaže dodatnim informacijama o prosječnoj dobroti jedinki, te uočava standardno odstupanje od prosječnog voća. Do sličnih informacija možemo i mi doći ukoliko više puta pozivamo funkciju dobrote za slučajno odabrane (slučajno stvorene) jedinke. Na taj način možemo sa određenom sigurnošću pretpostaviti dodatne informacije o tome što funkcija vraća. Ukoliko nam funkcija d vrati n vrijednosti $x_1, x_2, \ldots, x_n$, tada možemo dobiti očekivanu vrijednost

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.2}$$

i procjenu standardnog odstupanje

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}, \tag{2.3}$$

te pomoću $\overline{x}$ i s normalizirati dobivene vrijednosti.[12] Normalizacijom vrijednosti dobivenih od funkcije d možemo ukloniti redundantne informacije o dobroti jedinki. Normalizacijom vrijednosti $x_1, x_2, \ldots, x_n$ dobivamo vrijednosti $y_1, y_2, \ldots, y_n$ koristeći formulu (2.4).

$$y_i = \frac{x_i - \overline{x}}{s} \tag{2.4}$$

Dobivene vrijednosti $y_1, y_2, \ldots, y_n$ su valjana neovisno o tome zamijeni li funkcija d sa funkcijom $f = \alpha d(jedinka) + \beta$, pritom su α i β proizvoljne konstante.

Veličina populacija se treba mijenjati u ovisnosti o trenutnom standardnom odstupanju (devijaciji) dobrote populacije. Veličina populacije n je obrnuto proporcionalna standardnom odstupanju trenutne populacije u odnosu na standardno odstupanje početne populacije:

$$n_{trenutno} = n_{početno} \frac{s_{početna}}{s_{trenutna}}. \tag{2.5}$$

To znači da sa smanjenjem devijacije u populaciji, raste veličina populacije. Zapravo se time populacije opire smanjenju devijacije, kako algoritam na bi zapao u lokalni optimum. Izrazito smanjenje devijacije pokazuje da se populacija ujednačuje, te se na taj način može ustanoviti kada je smisleno završiti izvođenje algoritma. Veličina populacije je uvjetovana količinom različitog genetskog materijala unutar populacije. Za posljedicu je genetska raznolikost održana približno

konstantnom. Kad jedinke postaju slične, povećava se količina jedinki, jer slične jedinke sadrže sličan genetski materijal.

2.2.3. Dvije populacije

Ljudi pri svom ponašanju ne pokušavaju uvijek maksimizirati dobrotu. Ponekad se ljudsko ponašanje svodi na minimiziranje lošega. Takvo ponašanje je kod ljudi i životinja prepoznatljivo u slučaju straha ili nelagodae. Npr. mnoge životinje, kao i bebe, bježe od buke kretajući se pritom u smjeru koji minimizira nelagodu. Na temu bježanja od nelagode su napravljena mnoga sociološka istraživanja.[6]

Prilikom rješavanja nekih problema može se pokazati laganim utvrđivanje koliko je neko rješenje loše, a teškim raspoznavanje dobrih u odnosu na prosječna rješenja. U tom slučaju može se pokazati korisnim upogoniti genetski algoritam sa dvije populacije. Pritom jedna populacija pokušava pronaći što lošije rješenje, dok se druga populacija pokušava što više udaljiti od prve populacije. Takva izvedba je vremenski zahtjevnija, ali zato uspijeva doći do dodatnih informacija o dobroti jedinke.

U ovom radu se više neće razrađivati ideja o dvije populacije.

2.3. Operatori

U ovom dijelu rada bit će objašnjeni genetski operatori i dodatne funkcije korištene pri izgradnji generaliziranog genetskog algoritma (kraće GGA).

U GGA se mutacija koristiti na jednak način kao što se koristi i u genetskom algoritmu. Vjerojatnost mutacije se definira parametrom algoritma koji poprima vrijednost iz intervala $[0, 1]$, dok provedba mutacije ovisi o pojedinom problemu koji se rješava, te može više ili manje promijeniti jedinku.

Pri korištenju dobrote pojedine jedinke, uvijek se misli na normalizirane vrijednosti koje se dobivaju korištenjem izraza (2.4)

U GGA se selekcija provodi nad normaliziranim dobrotama jedinki, korištenjem tehnike ruletskog kola (engl. *roulette wheel selection*)[7]. Selekcijom se u GGA biraju jedinke nad kojima se mogu obaviti genetski operatori.

U nastavku će postepeno biti uvedeni novi pojmovi:

– sličnost,

– srodstvo,

– srodstvena dobrota i

– specijacija.

Također će biti pojašnjen način na koji se provodi križanje.

2.3.1. Sličnost

Pokazuje se korisnim maksimizirati količinu informacija o stanju procesa pretrage. Pritom je korisno moći utvrditi sličnost između neke dvije jedinke. Za to je potrebna funkcija koja će odrediti koliko su dvije jedinke međusobno slične:

$$sl = f_{sl}(jedinka_A, jedinka_B), sl \in [0, 1] \tag{2.6}$$

Vrijednost koju f_{sl} vraća predstavlja udio istih gena u jedinki. Ukoliko je $f_{sl} = 1$ tada jedinke sadrže iste gene, a ukoliko $f_{sl} = 0$ tada jedinke nemaju zajedničkih gena. Pritom je smisleno da vrijedi:

$$f_{sl}(jedinka_A, jedinka_A) = 1 \tag{2.7}$$

i

$$f_{sl}(jedinka_A, jedinka_B) = f_{sl}(jedinka_B, jedinka_A). \tag{2.8}$$

2.3.2. Srodstvo

U genetskom algoritmu se jedinka sastoji samo od svog kromosoma, tj. rješenja koje opisuje. Često je taj zapis rješenja netrivijalan, te zauzima primjetan prostor u memoriji. U tom slučaju se pokazuje korisnim zapisati još neke informacije o jedinki, kako bi se maksimizirala količina informacija o jedinki. Osim kromosoma, korisno je zapisati i informacije o jedinkinim roditeljima, kako bi se moglo usporediti imaju li neke jedinke slično "obiteljsko stablo". Ukoliko su jedinke nastale od sličnih predaka (što je čest slučaj u maloj populaciji), može se očekivati da sadrže sličan genetski materijal.

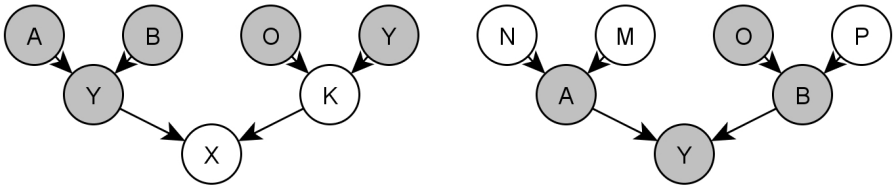
Budući da nije poželjno zapisivati cijele roditeljske jedinke, a čuvanje pokazivača na roditelje može rezultirati gubitkom informacija (ukoliko se roditeljska

jedinka eliminira), dobro je svim jedinkama dodijeliti jedinstveni identifikacijski broj (kraće ID). Tada je dovoljno u pojedinoj jedinki pamtiti identifikacijske brojeve njenih predaka.

Ukoliko npr. želimo pamtimo dvije razine predaka, te je svaka jedinka nastala križanjem dva roditelja, dovoljno je zapisati 7 ID–ova (6 ID–ova predaka, te vlastiti ID jedinke). Pritom je vlastiti ID jedinke na razini 0, 2 ID–a koja predstavljaju neposredne roditelje su na razini 1 (prvi preci), te su 4 ID–a koji predstavljaju roditelje roditelja na razini 2 (drugi preci). Sada se može izračunati sličnost između srodstva dvije jedinke funkcijom:

$$f_{sr}(A,B) = \frac{1}{2} \sum_{A_{IDi}=B_{IDj}} \frac{1}{2^{i_{razina}} 2^{j_{razina}}}. \tag{2.9}$$

Dana funkcija radi procjenu udjela istih gena u jedinkama, te vraća vrijednost iz intervala $[0, 1]$.



Slika 2.2: Primjer za izračunavanje srodstva

Prema funkciji (2.9), srodstvo jedinki sa slike 2.2 jest

$$f_{sr}(X,Y) = \frac{0.5*1 + 0.25*1 + 0.25*0.5 + 0.25*0.5 + 0.25*0.25}{2} = 0.53125$$

Ukoliko se npr. radi procjenu sličnosti dvoje djece istih roditelja, dobiva se $f_{sr} = \frac{2*0.5^2+4*0.25^2}{2} = 0.375$. Ukoliko bi količina zapamćenih razina predaka težila u ∞, vrijednost f_{sr} dvoje djece istih roditelja bi težila u 0.5, što odgovara očekivanoj sličnost gena između jedinki s istim roditeljima.

Srodstvo procjenjuje sličnosti dvije jedinke (ukoliko je križanje ispravno osmišljeno).

2.3.3. Križanje

Budući da je priroda razvila vrlo sofisticirane postupke kojima se biraju partneri prije provedbe križanja, može se naslutiti postojanje tehnika koje poboljšavaju

razvitak populacije pažljivim biranjem jedinki koje se križaju. Osim životinja, ljudi također koriste različite metode pri odabiru partnera. Načini na koje ljudi biraju partnere često se objašnjava teorijom socijalne razmjene[8][6]. Teorija socijalne razmjene pretpostavlja da je ono što ljudi čine u znatnoj mjeri određeno željom da maksimalno povećaju svoje dobitke i maksimalno smanje svoje gubitke. Posljedica teorije jest u tome da ljudi često odlučuju biti s ljudima slične "klase". Uspješni s uspješnima, lijepi sa lijepima, bogati sa bogatima.

Općenito, jedinka prije potencijalnog križanja procjenjuju kvalitetu druge jedinke, te odlučuju o razmnožavanju uzimajući u obzir informacije kojima raspolaže. Pritom jedinka može raspolagati informacijama o dobroti druge jedinke, sličnosti sa drugom jedinkom, te informacijama o srodstvu.

Sve se jedinke mogu međusobno križati, ali češće dobre jedinke sa dobrima, dok loše (prema teoriji socijalne razmjene) sa svima. Ipak, dobre jedinke se ne žele križati sa lošima, te se loše jedinke moraju križati sa lošima.

Križanjem sličnih jedinki ublažuje se problem dobivanja smislenih jedinki. Često je za dani problem teško osmisliti operator križanje koji će koristeći kromosome dviju jedinki stvoriti potomka koji će biti smislen (imati dobrotu koja je u ranku dobrote pojedinog roditelja). Navedeni problem se umanjuje ukoliko se češće križaju sličnije jedinke jer je tada lakše stvoriti smislenog potomka.

Srodstvo je procjena sličnosti jedinki, međutim, često operator križanje nije moguće osmisliti savršeno, te tada očekivana sličnost jedinki odstupa od dobivene sličnosti. Zato je korisno definirati faktore sličnosti i srodstva kojima se može definirati privlači li se više srodno i slično, ili se pak slično privlači, a srodno odbija, i sl.

U generaliziranom genetskom algoritmu dvije selektirane jedinke mogu odbiti križanje ukoliko im je dobrota vrlo različita, te ukoliko sličnost i srodstvo nisu zadovoljavajući.

2.3.4. Srodstvena dobrota

Provedena su mnoga istraživanja o načinu na koji si ljudi međusobno pomažu u nesrećama. Jedno od zapaženijih istraživanja je provedeno intervjuiranjem preživjelih u požaru u odmaralištu 1973. godine.[6] Otkriveno je da su ljudi, koda su postali svjesni požara, prije nego su upozorili ostale stanovnike zgrade, bili skloniji potražiti članove svoje obitelji nego prijatelje. Evolucijski psiholozi ne tvrde da ljudi svjesno odvaguju biološku važnost vlastitog ponašanja prije nego odluče hoće li pomoći. Ipak, prema evolucijskoj teoriji, geni ljudi koji slijede ovo pravilo "biološke važnosti" imaju veću vjerojatnost preživljavanja od gena ljudi koji ga ne slijede.[9] Tijekom tisućljeća, selekcija srodnika vjerojatno je postala usađena u ljudsko ponašanje.

Pokazuje se korisnim uvesti socijalne čimbenike u proces evolucije. U prirodi se dobre jedinke međusobno brane kako bi preživjele i osigurale preživljavanje svog genetskog materijala. Pritom dobre jedinke štite svoje loše rođake do eliminacije.

Loše jedinke (jedinke sa malom dobrotom) često mogu sadržavati poželjne (dobre) gene, ukoliko su nastale od dobrih roditelja. Preživljavanje takvih loših jedinki je poželjno. Posljedica toga je da loše jedinke sa dobrim genima (od dobrih roditelja) kasnije mogu stvoriti dobre potomke.

Srodstvena dobrota utvrđuje potencijal dobrih gena, tj. "plemenitost" pojedine jedinke, te otežava eliminaciju "plemenitih" jedinki. Srodstvena dobrota predstavlja očekivanu dobrotu jedinke (određenu dobrotom roditelja), te je definirana na sljedeći način:

$$d_{sr}(X) = \frac{1}{n}\sum_{i=1}^{n} d(X_{roditelj_i}). \tag{2.10}$$

U navedenom izrazu je dobrota d definirana izrazom (2.1). Ukoliko jedinka nastaje križanjem dva roditelja, izraz (2.10) prelazi u $d_{sr}(X) = \frac{d(X_{roditelj_1})+d(X_{roditelj_2})}{2}$.

2.3.5. Specijacija

Biološka specijacija (engl. *speciation*)[27]je evolucijski proces kojim nastaju nove biološke vrste. Pritom se dio početne populacije promijeni do te mjere da se više ne može križati sa ostatkom početne populacije. Specijacija zahtjeva da se samo slični mogu križati, tako da križanje sličnih podupire specijaciju.

Inspiriran procesom biološke specijacije, uveden je operator specijacije koji danu jedinku znatno mijenja i poboljšava, te ju ujedno odvaja od njenih roditelja (mijenja ID–ove roditelja). Neke jedinke treba pokušati pomaknuti u lokalni maksimum (u maksimumu se stvara podvrsta). Pomicanje u lokalni maksimum se može provesti gradijentnim penjanjem[23], ili hill climbing algoritmom[24]. Budući da se gradijentno penjanje često ne može provesti, hill climbing je bolji izbor. Hill climbing se uvijek može provesti tako da jedinka predstavlja "čvor", a "susjedni čvorovi" nastaju mutacijom jedinke. Pritom provodimo manji fiksan broj koraka algoritma.

Vjerojatnost specijacije u generaliziranom genetskom algoritmu se definira parametrom algoritma iz intervala $[0, 1]$ koji predstavlja vjerojatnost provedbe specijacije nad novostvorenom jedinkom.

2.4. Konačan algoritam

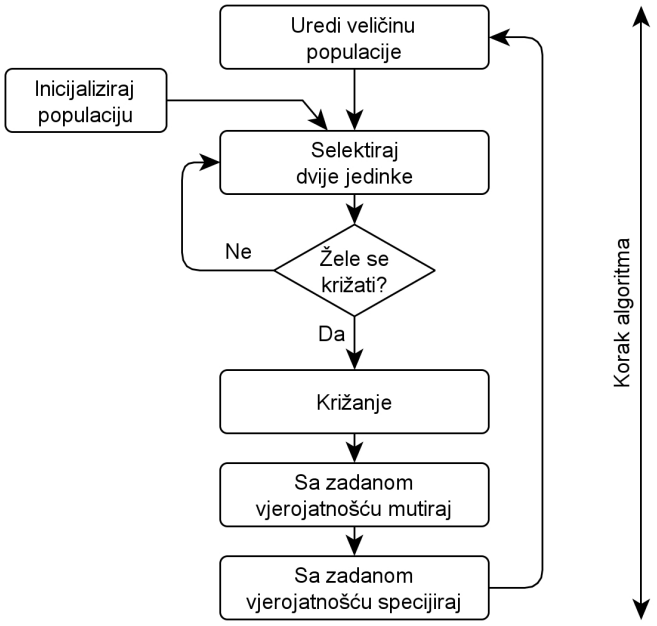
Pokazuje se korisnim ugraditi "elitizam" u algoritam.[7] Elitizam je mehanizam koji štiti najbolju jedinku u populaciji od eliminacije ili promjene. Podrazumijeva se da generalizirani genetski algoritam mora podržavati elitizam, te se to neće dodatno napominjati.

Opći nacrt provođenja GGA opisan je slikom 2.3.

Prilikom pokretanja algoritma stvara se populacija određene veličine. Populacija sadrži slučajno stvorene jedinke. Slučajno stvorenim jedinkama se izračunaju vrijednosti funkcije dobrote, te se izračuna početna očekivana dobrota jedinki i početna standardna devijacija dobrote jedinki.

Svaki korak algoritma provodi jedno križanje, te se svakim korakom algoritma stvara jedna nova jedinka koja se dodaje u populaciju. Pri svakom dodavanju jedinke u populaciju, ili eliminiranju jedinke iz populacije, ažurira se vrijednost trenutne očekivane dobrote jedinki u populaciji, te trenutna standardna devijacija dobrota jedinki u populaciji.

Jedinka se može **stvoriti** križanjem ili pri uređivanju veličine populacije. Jedinka se može **promijeniti** mutacijom ili specijacijom. Jedinka se može **eli-**



Slika 2.3: Ponašanje generaliziranog genetskog algoritma

minirati jedino pri uređivanju veličine populacije.

Algoritam započinje selektiranjem dvije jedinke iz populacije. Pritom se koriste normalizirane vrijednosti dobrote, te se odabir vrši pomoću ruletskog kola[7]. Nakon što se selektiraju dvije jedinke, provjerava se žele li se selektirane jedinke križati. Jedinke mogu odbiti križanje ukoliko su vrlo različite dobrote (npr. najbolja jedinka se želi križati jedino sa boljom polovicom jedinki), te ukoliko nemaju odgovarajuću sličnost i srodstvo. Prilikom objašnjavanja ostvarenog algoritma bit će detaljnije pojašnjeno kada jedinke odbijaju križanje. Bitno je da dvije selektirane jedinke u prosjeku ne odbijaju križanje previše često, kako broj selekcija ne bi počeo usporavati algoritam (prilikom objašnjenja implementacije bit će naveden izrazito brz postupak selekcije). Ukoliko jedinke odbiju križanje, selektira se novi pâr jedinki, te se taj postupak ponavlja dok jedinke ne prihvate križanje. Nakon što jedinke prihvate križanje, provodi se križanje, te nastaje nova jedinka. Ta nova jedinka se sa zadanom vjerojatnošću podvrgava mutaciji, te sa nakon mutacije sa zadanom vjerojatnošću podvrgava specijaciji. Nakon toga se jedinka ubacuje u populaciju (te se uređuje trenutno očekivanje i devijacija populacije). Nakon toga se provodi uređivanje veličine populacije. Uređivanje veličine

populacije se može provoditi kao prva ili kao zadnja radnja u pojedinom koraku algoritma (svejedno je). Prilikom uređivanja veličine populacije prema formuli (2.5) određuje se željena veličina populacije, te se dodaje ili eliminira jedna po jedna jedinka u populaciji dok god $|trenutnaVeličina - željenaVeličina| > 1$. Ukoliko se jedinka dodaje, dodaje se slučajno stvorena jedinka. Ukoliko se provodi eliminacija, slučajno se odabire jedinka, te ju se pokušava eliminirati uzimajući u obzir jedinkinu dobrotu i srodstvenu dobrotu. Pokušaj eliminacije ne mora uspjeti (slično kao što ni križanje ne mora uspjeti).

Algoritam se pokreće sa parametrima kojima se određuje barem:

- Početna veličina populacije,
- Vjerojatnost mutacije,
- Vjerojatnost specijacije,
- Faktor sličnost i
- Faktor srodstva.

Slijedi kratak pseudokod algoritma:

```
Inicijaliziraj populaciju
Ponavljaj (zadani broj koraka, ili dok na odlučiš prekinuti)
    Uredi veličinu populacije
    Dok (nije provedeno križanje)
        x=selektiraj()
        y=selektiraj()
        Ako (x nije y i jedinke se žele križati)
            z=križaj(x,y)
    Ako (ispitivanje vjerojatnosti za mutaciju)
        z=mutiraj(z)
    Ako (ispitivanje vjerojatnosti za specijaciju)
        z=specijiraj(z)
    Dodaj jedinku z u populaciju
```

U algoritmu se u pojedinom koraku sve radnje mogu obaviti u konstantnoj ili logaritamskoj složenosti. Jedine vremenski zahtjevnije radnje u algoritmu čine:

križanje, mutacija, izračun sličnost, stvaranje slučajne jedinke, kopiranje jedinke, te izračun dobrote. Sve navedene radnje ovise o specifičnom problemu koji se pomoću GGA rješava, te im složenost ovisi o veličini zapisa potencijalnog rješenja (kromosom).

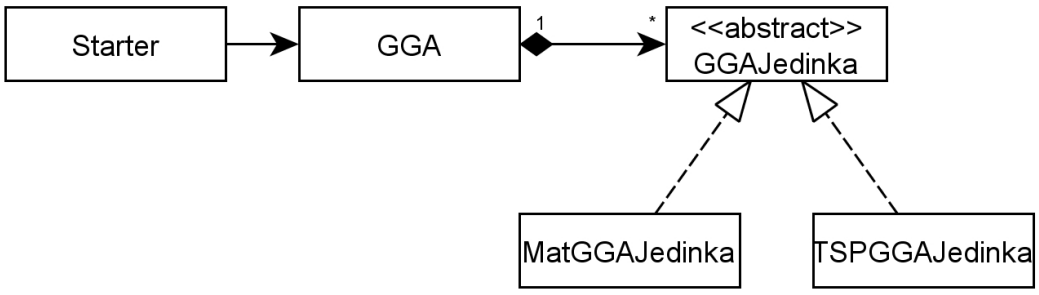
GGA provodi više "kućanskih poslova" od GA, te ima malo veću konstantu izvođenja, ali zauzvrat daje bržu konvergaciju, bolje čuvanje genetske raznolikosti i više informacija o stanju procesa pretrage, a sve to utječe na brži pronalazak rješenja.

3. Ostvareni GGA

U okviru rada je razvijena programska potpora koja ostvaruje osmišljeni generalizirani genetski algoritam. U ovom dijelu rada će ista biti opisana.

3.1. Općenito o izvedbi

Za potrebe ispitivanja generaliziranog genetskog algoritma, razvijen je programski sustav u Javi definiran u paketu "com.kusalic.GGA". Razvijen je lako nadogradiv kompleksan sustav s dodanim sučeljem za ispitivanje performansi algoritma. Na slici 3.1 je prikazan pojednostavljen UML dijagram bitnijih klasa iz paketa "com.kusalic.GGA".



Slika 3.1: UML dijagram bitnijih klasa

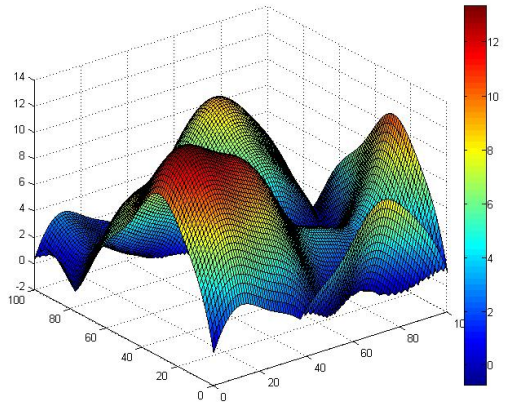
Klasa Starter ostvaruje grafičko korisničko sučelje kroz koje se može pokretati generalizirani genetski algoritma. Klasa Starter koristi nestandardnu vanjsku klasu JFreeChart kako bi stvorila dijagrame koji prikazuju izvođenje generaliziranog genetskog algoritma. Klasa GGA ostvaruje ponašanje algoritma. Prilikom stvaranja klase GGA vrši se inicijalizacija algoritma, te se provedba pojedinog koraka algoritma vrši pozivom metode step(). Klasa raspolaže sa mnoštvom dodatnih metoda koje vraćaju razne informacije o trenutnom stanju

populacije. Klasa GGA sadrži populaciju jedinki agregirajući jedinke definirane klasom koja nasljeđuje apstraktnu klasu GGAJedinka. Apstraktna klasa GGA-Jedinka obavlja sve poslove vezane za informacije o jedinki, koji nisu vezani za sam kromosom. GGAJedinka uređuje dobrotu i srodstvenu dobrotu te definira ID jedinke i predaka, tako da klasa koja nasljeđuje GGAJedinku mora definirati jedino metode koje barataju kromosomom (zapisom potencijalnog rješenja).

Za potrebe ispitivanja su riješena dva optimizacijska problema. Prvi problem je traženja maksimuma funkcije dvije varijable

$$f(x,y) = \frac{|x(x-37)(x-100)+y(y-77)(y-100)|}{10^4} + \frac{1}{2}(sin(\frac{x}{5})+cos(\frac{y}{5})), \quad (3.1)$$

na području $x \in [0,100], y \in [0,100]$. Funkcija je prikazana na slici 3.2.



Slika 3.2: Funkcija koju GGA optimizira

Drugi optimizacijski problem ostvaren za potrebe prikaza rada GGA jest problem trgovačkog putnika (engl. *Travelling salesman problem*), kraće TSP.

Oba problema su ostvarena nasljeđivanjem klase GGAJedinka. Prilikom nasljeđivanja je dovoljno u naslijeđenoj klasi ostvariti zapis jedinke, te apstraktne metode krizanje(x,y), mutacija(), slicnost(x), dobrota(), slucajnoPopuni() i kopiraj(x), koje barataju jedino sa zapisom jedinke (za ostalo se brine klasa GGA-Jedinka).

Navedena izvedba omogućuje vrlo jednostavno dodavanje novih problema koje treba ostvariti.

Osim navedenog paketa, za potrebe izrade rada je napisano mnogo dodatnog koda u Javi i Matlab–u.

3.2. Algoritam

Generalizirani genetski algoritam je ostvaren u klasi GGA. Klasa ima samo jedan konstruktor koji prima pet parametara s kojima se algoritam provodi:

```
public GGA(int velicinaPopulacije, double vMutacija,
        double vSpecijacija, double fSrodstvo, double fSlicnost);
```

velicinaPopulacije je početna veličina populacije

vMutacija je vjerojatnost provedbe operatora mutacije

vSpecijacija je vjerojatnost provedbe operatora specijacije

fSrodstvo predstavlja faktor srodstva (njegova vrijednost će kasnije biti pojašnjena)

fSlicnost predstavlja faktor sličnosti (njegova vrijednost će kasnije biti pojašnjena)

Klasa ima privatne članove koji pamte parametre algoritma, populaciju, te statistički korisne informacije o populaciji. Populacije se pamti u strukturi TreeSet koja ostvaruje algoritma crveno-crnog stabla[13][4] (engl. *red-black tree*). TreeSet održava jedinke sortirane po dobroti (sortiranost je ista, radi li se po apsolutnim ili normaliziranim vrijednostima dobrote), te omogućuje u složenosti $O(log\ n)$ ubacivanje, vađenje i pretraživanje jedinki u populaciji (n u zapisu složenosti predstavlja veličinu populacije). Zapis pomoću TreeSet–a ima mnoge prednosti nad zapisom u HashSet–u, što će biti jasnije kasnije. Član Sx pamti trenutnu sumu dobrote jedinki u populaciji, dok član $Sx2$ pamti trenutnu sumu kvadrata dobrote jedinki u populaciji. Članovi Sx i $Sx2$ se ažuriraju prilikom svakog izbacivanju ili ubacivanju jedinke u populaciju. Korištenjem članova Sx i $Sx2$ može se izračunati očekivanje dobrote izrazom $\overline{x} = \frac{Sx}{n}$ (n je trenutna veličina populacije), dok se devijaciju dobrote može dobiti izrazom:

$$s = \sqrt{\frac{Sx2 - 2\overline{x}Sx + \overline{x}^2 n}{n-1}}. \tag{3.2}$$

U klasi se dodatno pamte početno očekivanje i devijacija populacije (očekivanje i devijacija slučajne jedinke).

Korak algoritma je definiran metodom step() koja oponaša proces prikazan na slici 2.3, te pritom poziva odgovarajuće privatne metode klase.

Specijacija se obavlja pomoću 10 koraka hill climbing[24] algoritma.

Križanje se obavlja ukoliko za dobrote jedinki d_x i d_y vrijedi:

$$|\frac{d_x - \overline{x}}{0.675s} - \frac{d_y - \overline{x}}{0.675s}| < 2 \tag{3.3}$$

($\pm 0.675s$ predstavlja interval povjerenja od 50%), te ukoliko je zadovoljen izraz (3.4).

$$f_{sr} * \text{fSrodstvo} + f_{sl} * \text{fSlicnost} > rand[-1, 2] \tag{3.4}$$

($rand[-1, 2]$ predstavlja slučajan broj iz intervala $[-1, 2]$)

Slučajna jedinka sa dobrotom d i srodstvenom dobrotom d_{sr} (vidi izraz (2.10)) preživljava **eliminaciju** ukoliko vrijedi izraz (3.5)

$$rand[0, 1] * ((\frac{d - \overline{x}}{1.645s} + 1) + (\frac{d_{sr} - \overline{x}}{1.645s} + 1)) > 0.5 \tag{3.5}$$

($rand[0, 1]$ predstavlja slučajan broj iz intervala $[0, 1]$, dok $\pm 1.645s$ predstavlja interval povjerenja od 90%.) Izraz (3.5) osigurava da prosječna jedinka bude eliminirana u 25% slučajeva.

Selekcija se provodi tehnikom ruletskog kola, te se pritom dobrota d, koja ulazi u ruletsko kolo, normalizira izrazom (3.6).

$$d_{selekcija} = \frac{d - \overline{x}}{1.645s} + k \tag{3.6}$$

Konstanta k predstavlja pomak standardne razdiobe sa 90% vrijednosti u intervalu $[-1, 1]$, te je u ostvarenom algoritmu podešena na 1. Time se može očekivati da 5% jedinki ima vrijednost $d_{selekcija}$ manju od 0, te ih se izbacuje iz ruletskog kola.

Selekcija predstavlja dio algoritma koji je najteže ubrzati. Da bi se ubrzala selekciju potrebno je umjesto TreeSet–a ostvariti vlastitu hibridnu kombinaciju

crveno-crnog stabla sa ugrađenim turnament stablom[28][29] koje i u unutrašnjim čvorovima održava sumu čvorova podstabla, te broj čvorova u podstablu. Takvom izvedbom se može prvo u $O(log\ n)$ pronaći najbolja jedinku kojoj $d_{selekcija}$ poprima negativnu vrijednosti, te se zatim može nad hibridnom strukturom u $O(log\ n)$ pronaći jedinku koju je selekcija odabrala.

Koristeći navedenu strukturu podataka, može se postići da sve radnje koje vrši GGA, a ne pojedina jedinka, budu izvedeni u složenosti $O(1)$ ili $O(log\ n)$.

Ipak, navedena iznimno zahtjevna implementacija bi uvelike nadmašila opseg ovog rada, te je zato selekcija ostvarena u $O(n)$, budući da se i operatori koje TSP jedinka provodi vrše u istoj složenosti, te se time ne povećava stupanj složenosti ostvarenog algoritma.

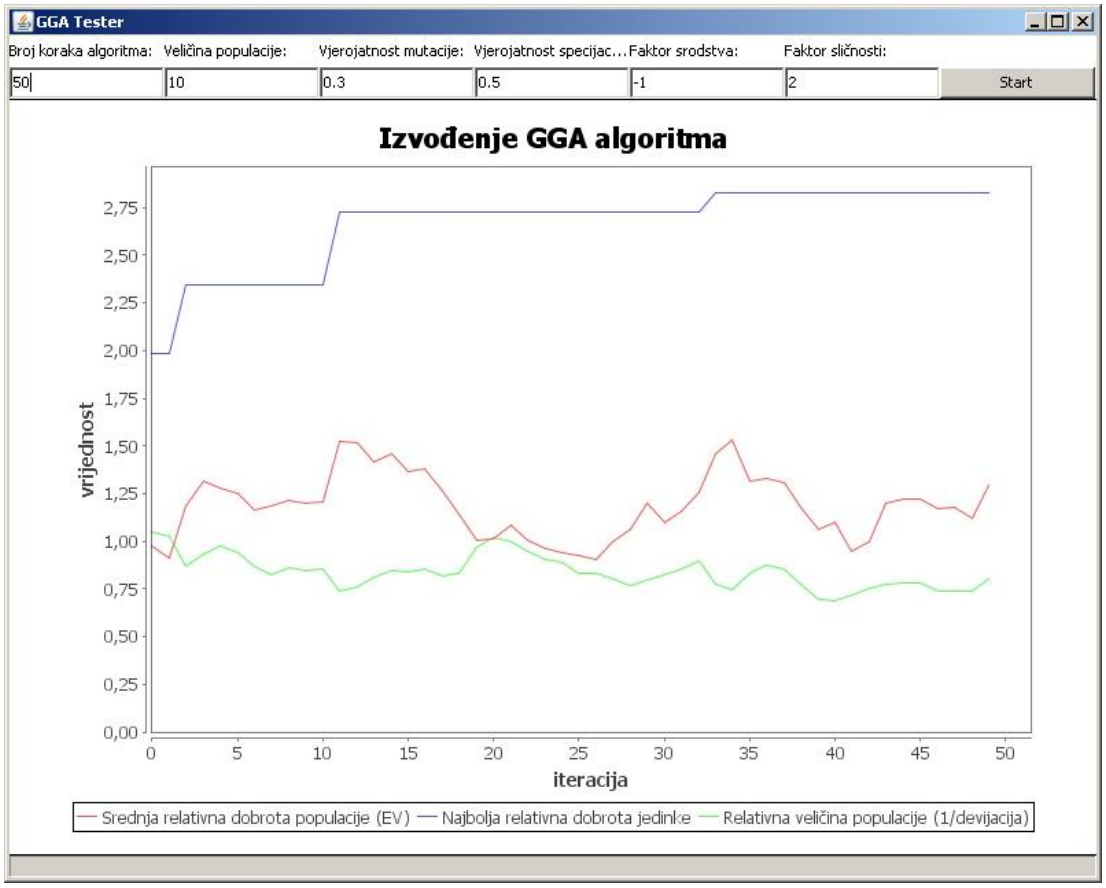
U sve ostvarene klase je ugrađena *javadoc* dokumentacija koja pojašnjava točno ponašanje svake metode u svim klasama paketa "com.kusalic.GGA".

3.3. Izvođenje

Na slici 3.3 je dan primjer izvođenja algoritma za problem traženja maksimuma funkcije sa dvije varijable, dok je na slici 3.4 dan primjer izvođenja za problem trgovačkog putnika (sa 50 gradova).

Grafičko sučelje koje program prikazuje je relativno lagano za koristiti. Pri vrhu se nalaze polja u koja se upisuju parametri s kojim se algoritam pokreće, dok se iznad njih nalazi opis parametra koji se unose u pojedino polje. Ukoliko se unesu neispravni parametri program daje upozorenje. Pokretanje programa se vrši pritiskom tipke "Start". Nakon što se pritisne tipka "Start", na progresnoj se traci, pri dnu grafičkog sučelja, može pratiti izvođenje algoritma. Nakon što se izvršavanje završi, prikazuje se graf koji opisuje izvođenje programa, te se taj isti graf snima u tekućem direktoriju u datoteku sa nazivom "chart.jpg".

Graf prikazuje na x osi korake algoritma (jedan korak je jedno križanje), dok na y osi prikazuje relativnu kvalitetu izvođenja algoritma. Na grafu je zelenom bojom prikazana relativna veličina populacije. Pritom y os prikazuje koliko puta je veličina populacije veća u odnosu na početnu (zadanu) veličinu populacije (npr. 0.5 označava da je populacija u datom trenutku duplo manja od

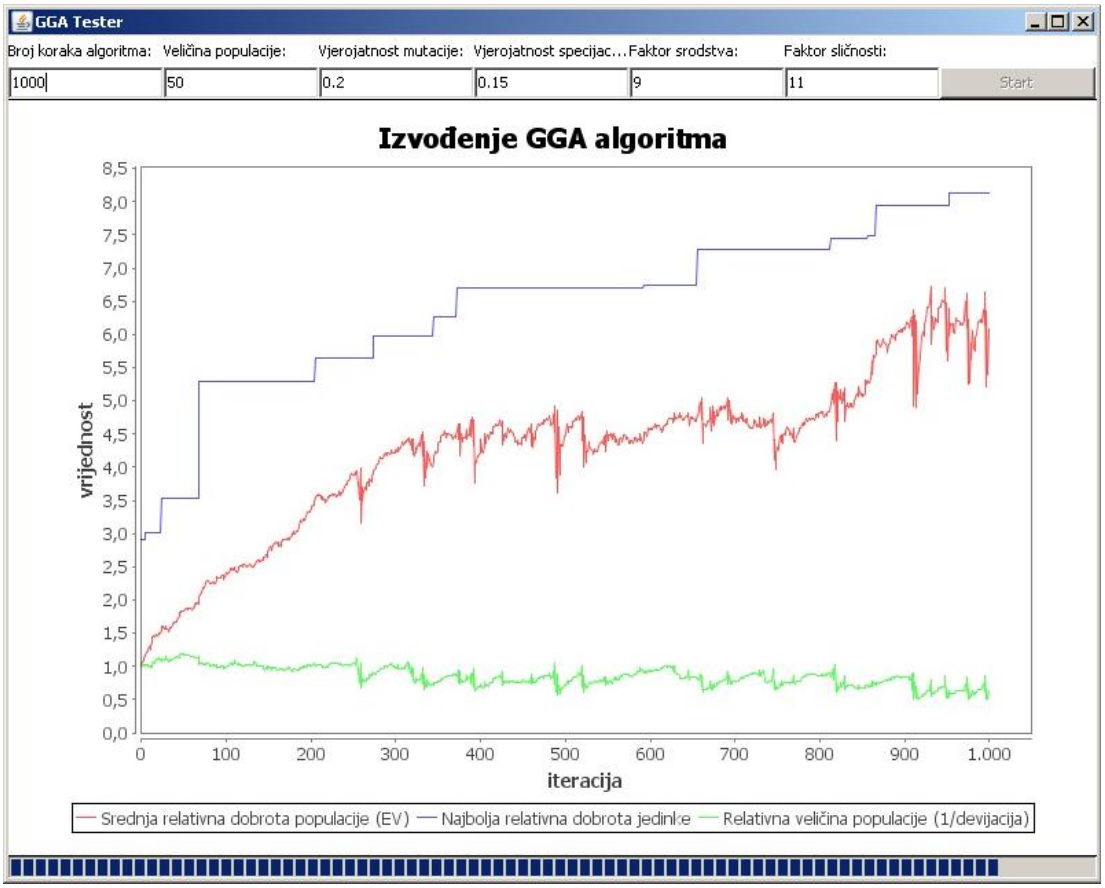


Slika 3.3: Primjer izvođenja GGA za problem traženja maksimuma matematičke funkcije

početne). Na grafu se crvenom bojom prikazuje srednja relativna dobrota populacije (očekivanje), dok se plavom bojom prikazuje najbolja relativna dobrota jedinke. Pritom y os prikazuje pomak dobrote u broju standardnih devijacija, od početnog očekivanje koje je na $y = 1$. Dakle, y os na 1 prikazuje očekivanu vrijednost slučajno generirane jedinke, dok npr. na 3 prikazuje vrijednost koja je za dvije standardne devijacije (dobrota slučajno generiranih jedinki) veća od očekivanja dobrote slučajne jedinke. Općenito, vrijednost v na y osi označava da je dobrota za $v - 1$ standardnih devijacija bolja od očekivane dobrote slučajne jedinke. y os prikazuje

$$\frac{vrijednost - \overline{x}_{početno}}{s_{početno}} + 1. \tag{3.7}$$

Klasa TSPGGAJedinka ostvaruje operatore potrebne za baratanje sa potencijalnim rješenjima TSP problema. Sve metode su ostvarene u složenosti $O(n)$.

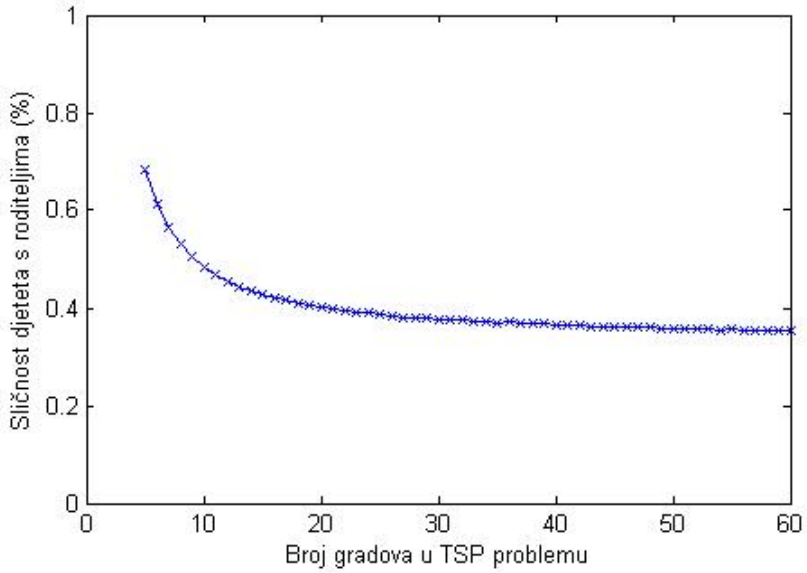


Slika 3.4: Primjer izvođenja GGA za TSP problem sa 50 gradova

Zapis jedinke je permutacija brojeva koji predstavljaju indeks pojedinog grada (čvora). Križanje je izvedeno kao što je predloženo u [10], kako bi se zajednička svojstva roditelja prenijela na dijete. Operator mutacije je proveden na tri načina. Sa vjerojatnošću od 50% se provodi pomak čvora do slučajne pozicije[11], sa vjerojatnošću od 25% se radi zamjena dva slučajno odabrana čvora[10], te se sa vjerojatnošću od 25% radi okretanje poretka kojim obilazimo čvorove iz slučajno odabranog intervala[10]. Za zadnju izvedbu mutacije je svejedno radi li se okretanje intervala za čvorove između dva izabrana čvora, ili za sve ostale.

Roditelji operatorom križanja stvaraju dijete koje im je slično. Graf 3.5 prikazuje sličnost djeteta s roditelja nakon križanja ovisno o broju gradova u TSP problemu koji se rješava. Svaka točka slike je izračunata kao prosjek 10000 operacija križanja.

Na slici 3.6 je prikazano izvođenje GGA za TSP problem sa 50 gradova. Veličina populacije koja je manja od početne pokazuje da algoritam uspijeva



Slika 3.5: Sličnost djeteta s roditeljima u TSP problemu

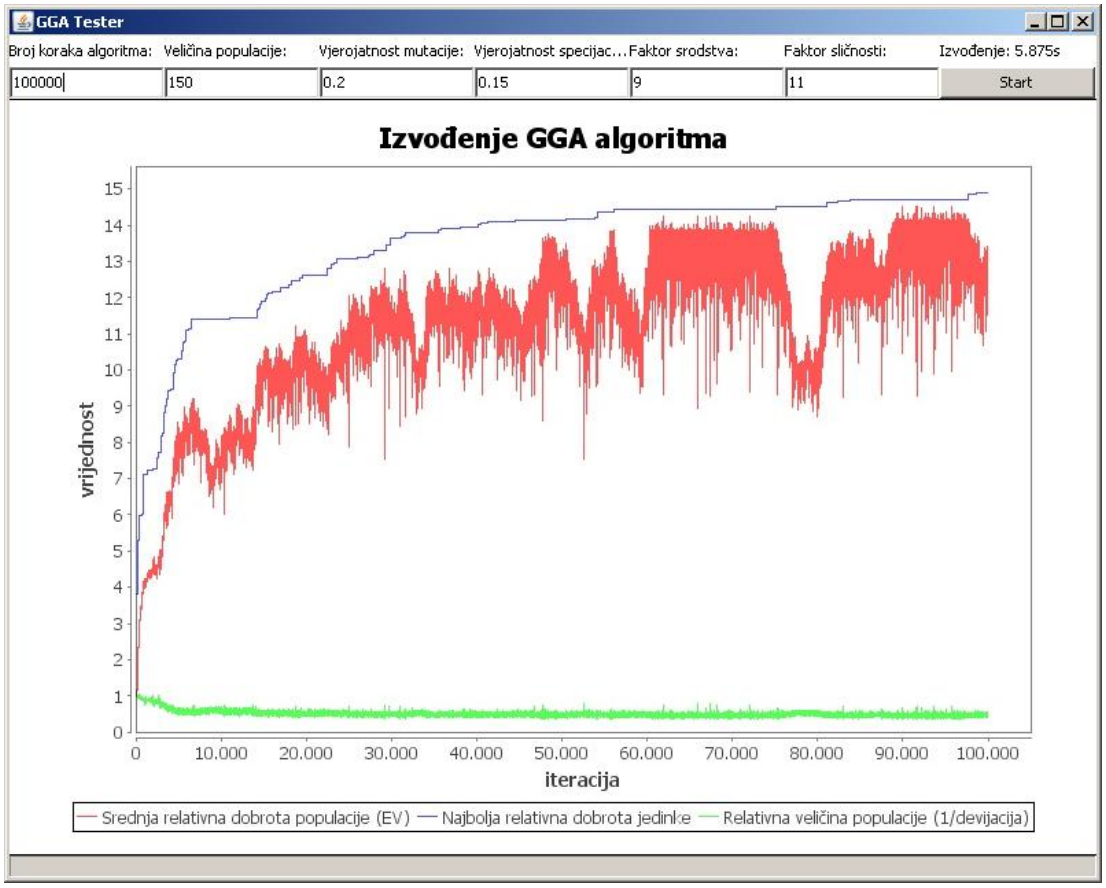
zadržati genetsku raznolikost unatoč znatnom poboljšanju prosječne dobrote. Algoritam postiže rješenje koje je za čak 15 devijacija bolje od slučajnog (što predstavlja znatno smanjenje prosječne udaljenosti između dva grada). Zanimljivo je primijetiti kako algoritam ne zapada u lokalni maksimum, te uspijeva neprestano napredovati, čak i nakon 50000 mutacija. Na slici 3.6 se može vidjeti da je izvođenje 100000 koraka algoritma zahtijevalo samo 5.875 sekundi.

Na slici 3.7 su dana dva primjera rješenja TSP problema koje je GGA algoritam stvorio za par sekundi izvršavanja, stvaranjem rješenja koje je 13 do 16 devijacija bolje od prosječnog.

3.4. Smisao pojedinog operatora

Veličina populacije utječe na brzinu konvergacije algoritma, te više jedinki uspijevaju bolje pretražiti prostor. Na slici 3.8 se može vidjeti kako mijenjanje veličine populacije utječe na izvršavanje algoritma prilikom rješavanja problema maksimizacije funkcije sa dvije varijable. Svi parametri osim veličine populacije su isti.

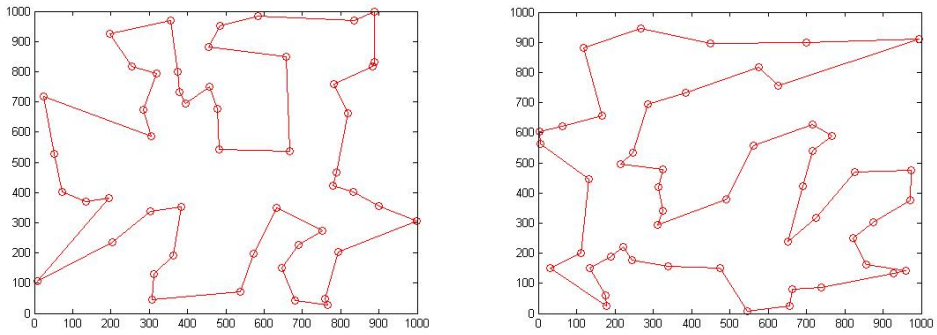
Vjerojatnost mutacije utječe na unos novog genetskog materijala u populaciju. Ukoliko se vjerojatnost mutacije postavi na previše malu vrijednost, algoritam



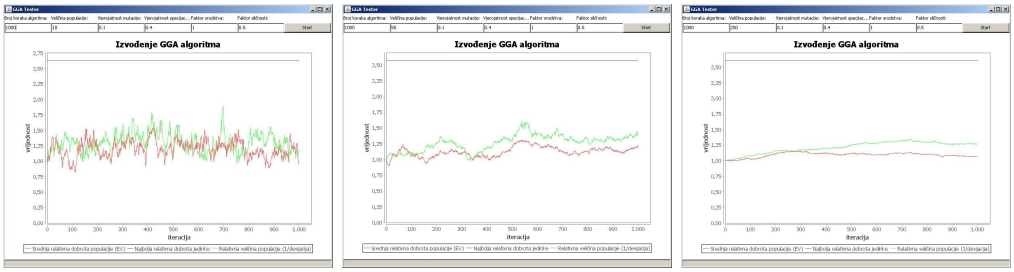
Slika 3.6: Primjer izvođenja 100000 koraka GGA

biva skloniji zapasti u lokalne maksimume. Ukoliko se pak vjerojatnost mutacije postavi na previše veliku vrijednost, algoritam teško otkriva globalni maksimum. Preporučena vrijednost vjerojatnosti mutacije je u intervalu $[0, 0.5]$.

Vjerojatnost specijacije omogućuje brže dohvaćanje boljih rješenja (potiče konvergaciju), ali zato sprječava algoritam pri dostizanju najboljih rješenja, te je preporučeno da bude iz intervala $[0, 0.3]$. Prevelika vjerojatnost specijacije, ne samo da će spriječiti algoritam u pronalasku globalnog maksimuma, nego može i usporiti izvođenje koraka algoritma. Na slici 3.9 se mogu vidjeti performanse algoritma za TSP problem sa 50 gradova, 1000 koraka algoritma, vjerojatnosti mutacije 0.2, te vjerojatnosti specijacije 0 i 1. Treba primijetiti da je izvođenje sa vjerojatnosti specijacije 0 nakon 1000 koraka dostiglo jednako dobro najbolje rješenje kao i izvođenje sa vjerojatnosti specijacije 1, ali zato ima bolju prosječnu dobrotu populacije.



Slika 3.7: Primjer TSP rješenja koje stvara GGA

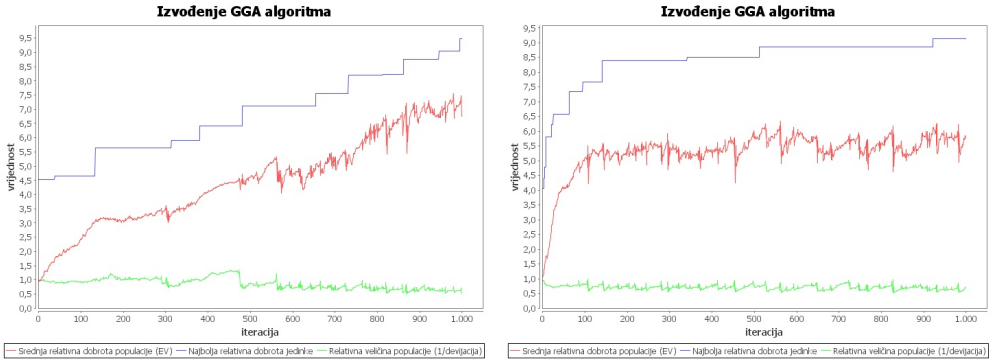


(a) 10 jedinki **(b)** 50 jedinki **(c)** 250 jedinki

Slika 3.8: Utjecaj veličine populacije na performanse algoritma

Faktor srodstva i faktor sličnosti utječu na prihvaćanje križanja između dvije jedinke prema izrazu (3.4). U trećini slučajeva će se križanje provesti i ukoliko su faktor srodstva i sličnosti postavljeni na 0. Faktori srodstva i sličnosti mogu biti i negativni.

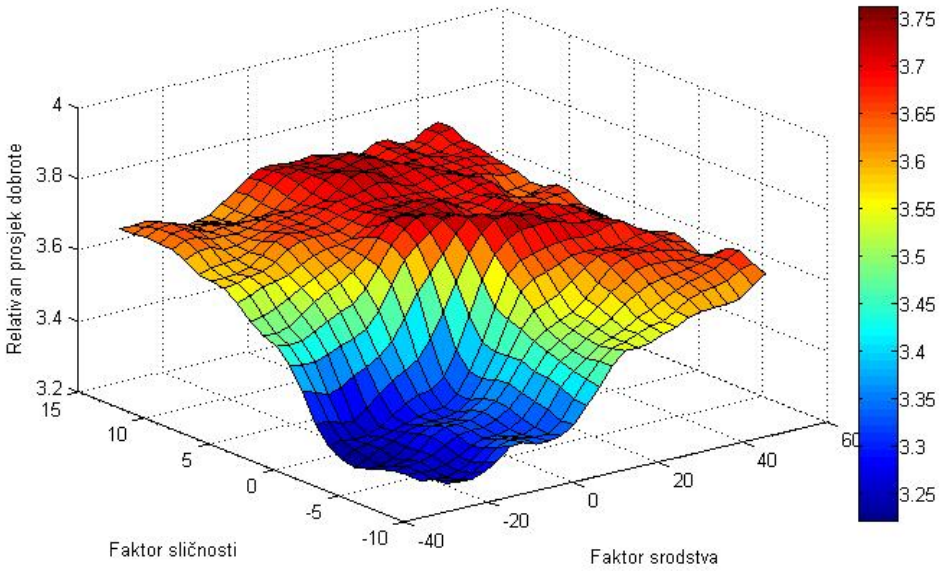
Da bi se mogli odrediti faktori srodstva i sličnosti, korisno je detaljnije poznavati prosječnu vrijednost srodstva i sličnosti za dani problem. Prilikom rješavanja problema trgovačkog putnika sa 50 gradova i populacijom veličine 50, prosječno srodstvo u populaciji tijekom izvođenja iznosi 0.03, dok je prosječna sličnost 0.1. Slika 3.10 prikazuje relativan prosjek dobrote (isto kao na y osi grafa koji stvara realizirani program) za razne faktore srodstva i sličnosti. Svaka točka na slici predstavlja poravnat prosjek 50 poziva GGA algoritma za TSP problem sa 50 gradova; broj koraka je 400, veličina populacije 50, vjerojatnosti križanja 0.2, te vjerojatnost specijacije 0.15. Slika jasno pokazuje poboljšanje nakon što $f_{sr} * \text{fSrodstvo} + f_{sl} * \text{fSlicnost}$ pređe određenu granicu (primijetite razlike u skali



(a) vjerojatnost specijacije je 0 **(b)** vjerojatnost specijacije je 1

Slika 3.9: Utjecaj specijacije na performanse algoritma

za srodstvo i sličnost). Budući da je za TSP problem križanje napravljena "ispravno", srodstvo biva uistinu dobra procjena sličnosti. Ipak, algoritam pokazuje malo bolje performanse ukoliko je sličnost jedinki naglašenija, a malo gore ukoliko je srodstvo jedinki naglašenije, kao što se vidi na slici 3.10. Algoritam pokazuje najbolje performanse kada je faktor srodstva oko 9, a faktor sličnosti oko 11. Za neki drugi specifičan problem može se pokazati korisnim znatno drukčije podesiti faktore srodstva i sličnosti.



Slika 3.10: Utjecaj faktora srodstva i sličnosti na rješavanje TSP problema

3.5. Usporedba sa klasičnim GA

Za potrebu usporedbe klasičnog genetskog algoritma sa GGA algoritmom razvijenom u ovom radu, u Javi je ostvaren paket "com.kusalic.GGA.usporedba". U navedenom paketu se nalaze klase Usporedba, GA i TSPJedinka. Detalji rada pojedine klase su opisani u obliku *javadoc* dokumentacije ugrađene u klase. Klasa Usporedba upogonjuje GA i GGA algoritme, te iscrtava njihovo ponašanje na zajedničkom grafu. Klasa GA ostvaruje genetski algoritam, dok klasa TSPJedinka opisuje jedinku kojom se rješava problem trgovačkog putnika.

Radi kvalitetne usporedbe algoritama metode križanja, mutacije, procjena dobrote, te stvaranja slučajne jedinke su u klasi TSPJedinka ostvarene na potpuno isti način kao u klasi TSPGGAJedinka, koju koristi GGA algoritam.

Prilikom ostvarivanja genetskog algoritma implementirana je eliminacijska selekcija. Ostvareni genetski algoritam ima sljedeći oblik[7]:

```
Eliminacijski genetski algoritam{
  generiraj početnu populaciju;
  sve dok nije zadovoljen uvjet završetka evolucijskog procesa{
    izračunaj vjerojatnost eliminacije za svaku jedinku;
    M puta jednostavnom selekcijom odaberi i izbriši jedinku;
    križanjem preživjelih jedinki nadopuni populaciju;
  }
}
```

Selekcija je provedena ruletskim kolom, te je kazna pojedine jedinke k_i određena izrazom

$$k_i = d_{max} - d_i \tag{3.8}$$

U izrazu je d_{max} dobrota najbolje jedinke u populaciji, a d_i dobrota i–te jedinke u populaciji. Navedeni način izračuna kazne osigurava "elitizam" najbolje jedinke.[7]

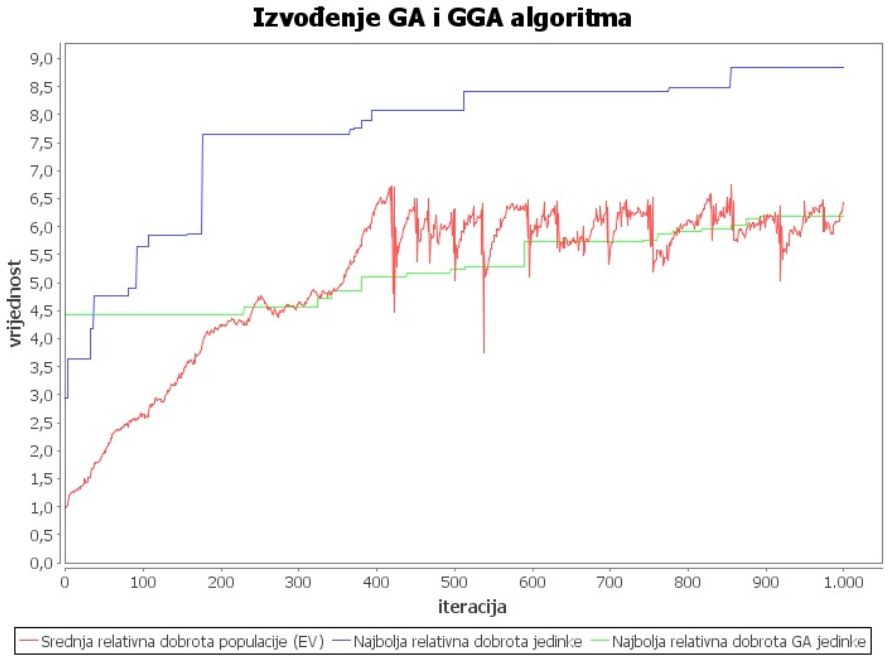
Za određivanje broja jedinki koje treba eliminirati u populaciji (M), korištena je preporuku iz [7], prema kojoj u populaciji od 30 jedinki treba eliminirati $\frac{30}{2}$, a u populacija od 100 jedinki treba eliminirati $\frac{100}{4}$ jedinki. Zato je za određivanje

broja jedinki koje treba eliminirati u populaciji veličine n korištena formula

$$M = \frac{n}{2 + \frac{n-30}{100-30}(4-2)} \tag{3.9}$$

Veličina populacije je jednaka veličini populacije za koju se pokrene GGA algoritam. Također je i vjerojatnost mutacije u genetskom algoritmu jednaka vjerojatnosti mutacije u GGA algoritmu.

Na slici 3.11 je dan primjer grafa koji stvara programa za usporedbu performanse GA i GGA algoritma. Na slici je dana usporedba ponašanja algoritama za provedbu 1000 koraka algoritma (1000 križanja) s populacijom od 50 jedinki. Plavom bojom je prikazana najbolja relativna dobrota GGA algoritma, a zelenom bojom najbolja relativna dobrota GA algoritma. Crvenom bojom je prikazana srednja relativna dobrota GGA populacije. y os ima isto značenja kao što je prije objašnjavano. Najbolja vrijednost GA algoritma je reskalirana sa istom srednjom dobrotom slučajne jedinke i devijacijom kao što je napravljeno u GGA algoritmu (prikazana skala je ispravna).

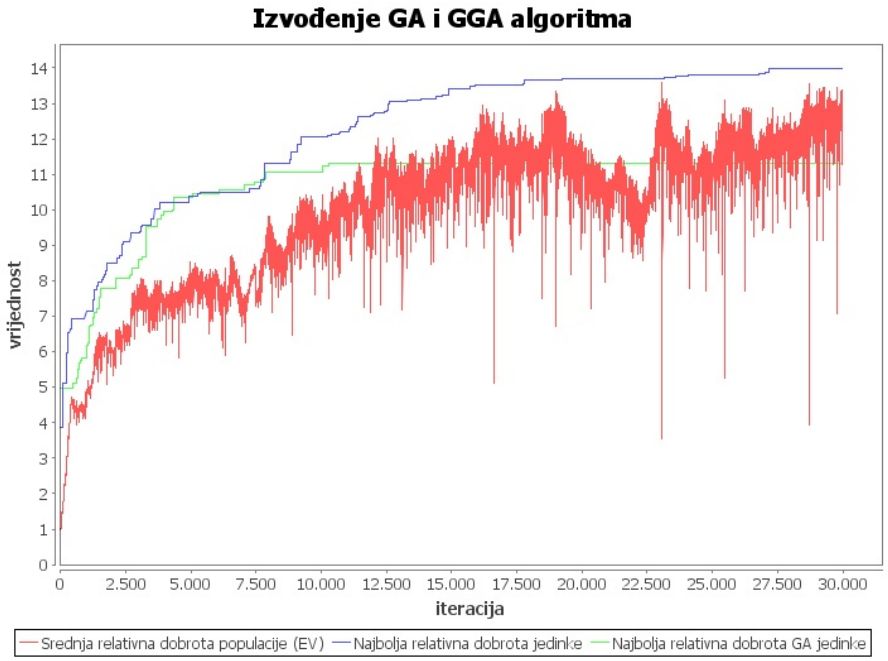


Slika 3.11: Usporedba GA sa GGA prilikom izvršavanja 1000 križanja nad populacijom od 50 jedinki

Oba algoritma provode jednako križanja, GGA čini u jednom koraku jedno križanje, dok GA u jednom koraku napravi M križanja. Zato se korak GGA

algoritma poziva M puta češće. Zanimljivo je primijetiti da na slici 3.11 čak i prosječna dobrota GGA uspijeva nadmašiti najbolju u GA.

Treba napomenuti da se ponekad dogodi da GA daje bolje rezultate od GGA, međutim, u prosjeku je GGA za nekoliko devijacija bolji. Prilikom dužeg izvođenju algoritama (više od sekunde), često nastaje graf kao onaj prikazan na slici 3.12. Često GA u određenom trenutku "zapadne" u lokalni maksimum, te ostane u njemu, dok se GGA konstantno probija prema sve boljim i boljim rješenjima, korisno iskorištavajući sve procesorsko vrijeme kojim raspolaže.



Slika 3.12: Usporedba GA sa GGA prilikom izvršavanja 30000 križanja nad populacijom od 100 jedinki

Bitno je napomenuti da GGA, kao i GA daju loše rezultate kada rješavaju TSP problem u kojem je broj gradova vrlo velik (npr. 700). Prilikom ispitivanja algoritma sa TSP problemom sa 575 gradova (test podatak "rat575.tsp.gz"), preuzetom sa [1], pokazuje se da GGA uspijeva u 1000000 iteracija sa populacijom od 250 jedinki (5 minuta izvršavanja) smanjiti duljinu puta TSP problema sa 575 gradova za samo $15\% - 20\%$. Za rješavanja takvih velikih TSP problema postoje drugi znatno prilagođeniji algoritmi. Razlog zašto se GA i GGA tako loše ponašaju prilikom rješavanja velikih TSP problema treba tražiti u omjeru dimenzije točke i broju točaka koje TSP obilazi. Drugi algoritmi agresivno iskorištavaju

činjenicu da se točke koje TSP obilazi nalaze u 2D prostoru (npr. optimalan put u 2D se nikad ne siječe). GA bi mogao biti konkurentan algoritam za rješavanje TSP problema sa mnogo točaka koje se nalaze u n–dimenzijalnom prostoru.

3.6. "No free lunch" teorem

Kako bi se ogradio od mišljenja da je GGA nadskup GA algoritma, te na taj način izbjegao moguću kritiku, u nastavku teksta je iznesen "no free lunch" teorem.

Prema "no free lunch" teoremu, čiji su autori David Wolpert i William G. Macready, u vezi problema optimizacije treba reći da *nema besplatnog ručka* (engl. *no free lunch*).[30] U metafori "nema besplatnog ručka", svaki "restoran" (algoritam) ima "meni" koji povezuje svako "jelo" (problem) sa "cijenom" (performanse algoritma prilikom rješavanja problema). Meniji raznih restorana su isti osim u jednom pogledu – cijene su izmiješane. Za posjetitelja restorana koji će sa jednakom vjerojatnošću naručiti bilo koje jelo, prosječna cijena ručka ne ovisi o odabiru restorana. No, posjetitelj koji je spreman naručiti neko od jela koja mu se sviđaju, pažljivim biranjem restorana može uštedjeti novce. Dakle, poboljšavanje performansi prilikom rješavanja problema ovisi o korištenju poznatih informacija kako bi odgovarajuće povezali algoritme sa problemima.

Drugim riječima, "nema besplatnog ručka" u potrazi ako i samo ako je distribucija funkcije cilja invarijantna na permutacije prostora kandidata rješenja. Ovaj uvjet nije previše točan u praksi, ali "(skoro) nema besplatnog ručka" teorem ukazuje da uvjet otprilike vrijedi.[25]

Određeni računarski problemi rješavaju se traženjem dobrog rješenja u prostoru potencijalnih rješenja. Za određeni problem, različiti algoritmi pretrage mogu postići različite rezultate, ali s obzirom na sve probleme, oni se ne razlikuju. Iz toga slijedi da algoritam koji postiže superiorne rezultate nad određenim problemom mora postići lošije rezultate na nekom drugom problemu. U tom smislu "nema besplatnog ručka" u pretrazi.

"No free lunch" teorem, kako su naveli Wolpert i Macready, tvrdi da su "bilo koja dva algoritma ekvivalentna kad se njihove performanse usrednje preko svih

mogućih problema". Rezultati "No free lunch" teorema pokazuju da pridruživanje odgovarajućeg algoritama pojedinom problemu daje bolje prosječne rezultate, nego korištenje istog algoritma za sve probleme.

Originalan "no free lunch" teorem tvrdi da za bilo koji par algoritama a_1 i a_2 vrijedi:

$$\sum_f P(h_m^y|f, m, a_1) = \sum_f P(h_m^y|f, m, a_2) \tag{3.10}$$

Ukratko, kad su sve funkcije f jednako vjerojatne, vjerojatnost opažanja proizvoljnog slijeda od m vrijednosti tijekom pretrage ne ovisi o algoritmu.

4. Paralelena obrada za GA/GGA

Paralelna izvedba algoritma može znatno povećati postignute rezultate za isto vrijeme izvršavanja, ukoliko algoritam sadrži dovoljno velike odsječke koji se mogu paralelno izvoditi. Genetski algoritmi se poprilično lagano mogu prilagoditi paralelnoj izvedbi prvenstveno zato što oponašaju prirodne procese koji se izvode paralelno. Taj veliki paralelni potencijal treba iskoristiti, te oblikovati algoritam koji ima što manje zajedničke komunikacije između pojedinih procesa (što manji slijedni udio – Amdahlov zakon).

Prije objašnjeni generalizirani genetski algoritam je podobniji za paralelnu izvedbu od običnog genetskog algoritma zato što u svakom trenutku koristi najviše dvije jedinke iz populacije. U nastavku će biti pojašnjen način paralelizacije generaliziranog genetskog algoritma.

4.1. Paralelni GGA

U GGA jedine kritične dijelove predstavlja selekcija, dodavanje jedinke u populaciju, te eliminacija jedinke iz populacije. Budući da sve tri radnje rade sa populacijom, logično se zaključuje da bi svaka komunikacija sa populacijom trebala predstavljati kritičan odsječak algoritma.

Razmatramo prvo selekciju. Ukoliko je populaciju zapamćena u obliku crvenocrnog turnament stabla koje pamti reference na pojedine jedinke (prije objašnjeno), jedinku se može selektirati u složenosti $O(log\ n)$ (n je veličina populacije), što je izrazito brza radnja u usporedbi sa npr. križanjem ili mutacijom. Nakon što je jedinka selektirana, nema smisla sprječavati provođenje drugih selekcija, te zato nakon selekcije treba zaključati samo jedinku koja je selektirana, a otključati

ostatak populacije (kako bi se istovremeno mogla provoditi neka druga selekcija). Ukoliko neka druga selekcija odabere jedinku koja je već selektirana, može se pričekati da selektirana jedinka postane slobodna, ili napraviti novu selekciju. Jedinka se otpušta (otključava) ukoliko odbije križanje ili nakon što se provede križanje – jedinka više nije potrebna.

Zanimljivo je primijetiti da se nikad ne vrši promjena nad jedinkom koja je u populaciji. Jedinka se mijenja jedino prilikom križanja, mutacije i specijacije, a tada još nije ubačena u populaciju. Zato je smisleno omogućiti paralelno čitanje jedinke. Ideja se može ostvariti tako što se prije ubacivanja jedinke u populaciju, napravi nekoliko dubokih kopija jedinke, te se u populaciju doda referenca na tablicu kopija jedinke. Tada je moguće nakon selekcije zaključati jednu od kopija jedinke, te omogućiti da se u isto vrijeme selektira i koristi druga kopija iste jedinke.

Nakon što križanjem nastane nova jedinka, slobodno se može vršiti mutacija, specijacija i izračun dobrote (svaki proces mora imati pristup vlastitoj memoriji sa opisom problema), neovisno o ostalim procesima.

Jedinku je na kraju potrebno dodati u populaciju. (Prethodno se jedinka kopira potreban broj puta, te se stvara tablica referenci na kopiju.) Potrebno je zaključati populaciju i izvršiti dodavanje reference u složenosti $O(log\ n)$ (budući da se radi o crveno-crnom stablu). Prilikom dodavanja jedinke potrebno je urediti varijable koje pamte statistički korisne informacije o populaciji. To se čini zaključavanjem potrebne varijable, te u složenosti $O(1)$ ažuriranjem sume dobrota jedinki i sume kvadrata dobrota jedinki.

Zadnji problem predstavlja uređivanje veličine populacije. Uređivanje veličine populacije se sastoji od dodavanja ili brisanja jedne po jedne jedinke. Ukoliko je potrebno dodati jedinku, slučajno se generira jedinka, izračunava njena dobrotu, zaključava populacija, dodaje jedinka u $O(log\ n)$, otključava populacija, te se uređuju varijable koje pamte statistički korisne informacije o jedinki. Ukoliko je potrebno smanjiti veličinu populacije, potrebno je zaključati populaciju, u složenosti $O(log\ n)$ slučajno odabrati jedinku, otključati populaciju, zaključati jedinku, te ju pokušati eliminirati. Ukoliko eliminacija uspije, potrebno je zaključati populaciju, u složenosti $O(log\ n)$ ukloniti iz populacije referencu na jedinku, te otključati populaciju.

Ipak, jednostavnije je napraviti zaseban proces koji održava veličinu populacije. Tada jedan proces održava veličinu populacije, a $x < \frac{n}{2}$ procesa vrše korake GGA algoritma. Dok god je broj procesa x umnožen sa $log\ n$ manji od dužine kromosoma pojedine jedinke i $x < \frac{n}{2}$, pojedini proces će u prosjeku više raditi koristan posao nego čekati kako bi ušao u neki kritičan odsječak algoritma.

4.2. Dodatna razmatranja

Priroda je na određen način "prisiljena" provoditi križanje i mutaciju odjednom, baš prilikom stvaranja organizma. Kasnije se stanice "kopiraju" kako bi organizam narastao, te više nije moguće provesti npr. mutaciju. Da bi se kasnije provela mutacija, priroda bi morala posjedovati mehanizam kojim može odjednom u svim (kopiranim) stanicama promijeniti genetski materijal. Budući da priroda nema te mehanizme, čini mutaciju samo jednom. Treba primijetiti da u genetskim algoritmima jedinka ne biva kopirana (zanemarimo kopiranja predloženo za poboljšanu paralelizaciju GGA), te uvijek postoji samo jedno mjesto gdje je zapisan genetski materijal jedinke. Zato genetski algoritam nema ograničenja s kojim se priroda suočava. Budući da nema ograničenja, genetski operatori u algoritmima se mogu provoditi bilo kada.

Priroda, makar ne može ponovno provoditi mutaciju, ipak utječe na jedinku da se promijeni. Životinje se aktivno prilagođavaju svom stilu življenja mijenjajući svoje ponašanje. Posljedica toga je bolje razvijanje određenih osjetila ili određene motorike. Biljke često prilagođavaju smjer u kojem rastu kako bi se bolje prilagodile okolišu koji ih okružuje. Priroda, makar ne može ponovno provesti mutaciju, ipak neprestano provodi prilagodbu jedinke. Na računalu je situacija jednostavnija, jer se uvijek može ponovno provesti mutacija.

Uklanjajući poredak kojim se genetski operatori provode može se dodatno ubrzati paralelna provedba algoritma. Moguće je stvoriti novi, paraleliziraniji algoritam koji nezavisno provodi operacije sa jedinkama. U takvom algoritmu bi jedinke "tekle" kroz populaciju, te bi postojalo nekoliko kategorija procesa:

- Nadzorni proces – uređuje veličinu populacije, skuplja informacije o stanju pretrage, te uređuje vjerojatnosti pojedinih genetskih operatora stvarajući procese i dodjeljujući im različite prioritete izvršavanja.

– Proces za selekciju boljih – vrši selekciju, te u izlazni red stavlja selektirane jedinke (jedinka se vadi iz populacije).

– Proces za slučajnu selekciju – vrši slučajnu selekciju, te u izlazni red stavlja selektirane jedinke (jedinka se vadi iz populacije).

– Proces za dodavanje jedinki u populaciju – sa ulaznog reda uzima jedinke i dodaje ih u populaciju.

– Procesi za stvaranje jedinki – uzimaju dvije jedinke sa izlaznog reda koji stvara selekcija boljih, stvaraju novu jedinku, računaju dobrotu stvorene jedinke, te stavljaju sve tri jedinke u red za dodavanje u populaciju.

– Procesi za mijenjanje jedinki – uzimaju jedinku sa izlaznog reda koji stvara slučajna selekcija, mutiraju jedinku, računaju joj dobrotu, te ju stavlja u red za dodavanje u populaciju.

– Procesi za eliminaciju jedinki – uzimaju jedinke sa izlaznog reda koji stvara slučajna selekcija, pokušavaju eliminirati jedinku, te ukoliko neuspiju stavljaju jedinku u red za dodavanje u populaciju.

– Procesi za specijaciju – uzimaju jedinku sa izlaznog reda koji stvara slučajna selekcija, vrše specijaciju, računaju dobrotu, te stavljaju jedinku u red za dodavanje u populaciju.

– Procesi za mutaciju neključnog gena – uzimaju jedinku sa izlaznog reda koji stvara slučajna selekcija, vrše mutaciju neključnog gena, računaju dobrotu, te stavljaju jedinku u red za dodavanje u populaciju. (Više o mutaciji neključnog gena nalazi se u poglavlju “Daljnja istraživanja”.)

Na opisani način jedinke bolje prolaze kroz “življenje”– stvaraju se, mijenjaju, stvaraju potomke i umiru. Navedeni sustav se vrlo lagano paralelizira. Ukoliko je potrebna još bolja paralelizacija, mogu se stvoriti nekoliko populacija sa vlastitim procesima za selekciju i dodavanje u populaciju, te tada ostali procesi mogu slučajno odabirati iz koje populacije uzimaju jedinke i u koju populaciju ih dodaju.

5. Algoritam osobnog prostora

5.1. Uvod

Zanimljivo je promatrati način na koji slučajne grupacije ljudi zauzimaju ograničen prostor. Ljudi na sličan način zauzimaju prostor, bilo da se radi o održavanju popularnog glazbenog koncerta, javnim protestima, popunjavanju stadiona za gledanje utakmice, skupljanju oko finisha na trkalištu ili skijalištu i sl. Npr. na glazbenom koncertu, blizu pozornice je uvijek nagurano mnogo ljudi, a kako se udaljavamo od pozornice, ljudi između sebe imaju sve više i više slobodnog prostora. Uzrok takvom ponašanju se pronalazi u ljudskoj potrebi za osobnim prostorom.[3]

Na tisuće knjiga i članaka napisano je o tome kako životinje, ptice, ribe i primati označavaju i brane svoj teritorij.[3] Većina životinja ima određeni prostor oko svoga tijela koji doživljavaju kao osobni, kao da se radi o dijelu njihovog tijela. Koliko se daleko taj prostor širi uglavnom ovisi o napučenosti područja u kojem je životinja odrasla i broju životinja u tom području. Teritorij lava odraslog u zabačenim područjima Afrike može imati polumjer od 50 i više kilometara, ovisno o gustoći lavlje populacije u tom području, dok će se osobni prostor lava odraslog u zatočeništvu s drugim lavovima smanjiti na svega nekoliko metara, što je izravna posljedica prenapučenosti. Životinje koje su prisiljene imati manji osoban prostor nego što im je potreban, u prosjeku žive kraće nego životinje koje imaju dovoljno osobnog prostora.

Poput većine životinja i čovjek ima svoj osobni prijenosi "zračni mjehur" koji posvuda vuče sa sobom. Ljudski osobni prostor se obično dijeli na

- Intimna zona (između 15 i 45 centimetara)
- Osobna zona (između 45 centimetara i 1.2 metra)

– Društvena zona (između 1.2 i 3.5 metara)

– Javna zona (više od 3.5 metara)

Prilikom ulaska druge osobe u naš osobni prostor osjećamo nelagodu, te pokušavamo osvojiti natrag izgubljeni prostor. Osoba koja želi slušati glazbeni koncert, kako bi bilo bliže izvođaču, mora pretrpjeti određenu nelagodu zato što gubi dio osobnog prostor. Postojanje ljudskog osobnog prostora objašnjava razne socijalne pojave kao što su npr. agresivnije ponašanje osoba u prepunjenim busevima, tramvajima i sl.

Budući da priroda naglašava postojanje osobnog prostora bioloških jedinki, vrijedi razmotriti ne bi li slično ponašanje jedinke u evolucijskom algoritmu donijelo određene prednosti.

U mnogim problemima genetski algoritmi imaju tendenciju konvergirati prema lokalnom optimumu ili čak slučajnoj točki pretrage, umjesto prema globalnom optimumu problema. Algoritam ne zna kako žrtvovati kratkoročnu dobrotu da bi dobio dugoročnu dobrotu. Vjerojatnost konvergacije prema lokalnom optimumu ovisi o obliku prostora pretrage – određeni problemi omogućuje lagano penjanje prema globalnom optimumu, dok neki drugi omogućuju lakši pronalazak lokalnog optimuma. Ovaj problem se može pokušati izbjeći koristeći drukčiju funkciju dobrote, povećavajući vjerojatnost mutacija ili korištenjem tehnika koje održavaju populaciju raznolikom; makar "no free lunch" teorem sugerira da nema općenitog rješenja ovog problema. Uobičajena tehnika koja održava populaciju raznolikom uvodi "kaznu blizine", pri čemu slična rješenja gube vjerojatnost preživljavanja u sljedeću populaciju, omogućujući drugim (manje sličnim) jedinkama da prežive u sljedeću populaciju.[22] U genetskim algoritmima je raznolikost bitna jer križanja u homogenoj populaciji ne donosi novo rješenje.

U svim problemima u kojima jedinke definiraju nekakvu točku u prostoru, smisleno je držati jedinke udaljenima jedne od drugih kako bi što bolje pretražile prostor (kako ne bi isti prostor bio nepotrebno više puta pretražen). U nastavku će biti osmišljen pomoćni algoritam koji međusobno udaljava jedinke, kako bi unio genetsku raznolikost u populaciju, te "izvadio" populaciju iz lokalnog optimuma u koji je upala.

5.2. Algoritam

Promotrimo ponovno situaciju koja se odvija na glazbenom koncertu. Osoba na glazbenom koncertu (jedinka) se pokušava približiti izvođaču (maksimumu) pritom pokušavajući maksimizirati količinu ugode (sumirana dobrota) koja se definira kao suma neugode zbog gubljenja osobnog prostora (blizina drugih jedinki) i ugode zbog blizine izvođaču (funkcija dobrota).

Usamljene jedinke traže više prostora, dok jedinke u "gužvi" traže manje prostora (sakupljanje oko optimuma). Jedinke koje su u gužvi imaju malo prostora oko sebe, pa se niti ne mogu puno pomaknuti, te time pokvariti dobro rješenje.

Želimo osmisliti algoritam osobnog prostora (engl. *Personal space algorithm – PSA*) koji u svakoj iteraciji uspijeva "razmaknuti" danu populaciju, kako bi populacija ponovno postala raznolika, te pritom ne pokvariti dobra rješenja (pogotovo ne najbolje rješenje – elitizam).

Slijedi opis algoritma. Neka n predstavlja veličinu populacije. Neka funkcija $dist(a_i, a_j)$ daje vektor udaljenosti od jedinke a_i do a_j. Neka za funkciju $dist$ vrijedi

$$dist(a_i, a_j) = -dist(a_j, a_i) \tag{5.1}$$

i

$$dist(a_i, a_i) = \vec{0}. \tag{5.2}$$

Tada

$$\overline{dist} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} |dist(a_i, a_j)| \tag{5.3}$$

predstavlja prosječnu udaljenost između dvije jedinke.

Neka

$$\vec{P_x} = \frac{\overline{dist}}{n-1} \sum_{i=1}^{n} \frac{dist(a_i, a_x)}{|dist(a_i, a_x)|^2} \tag{5.4}$$

predstavlja normaliziran ukupan pritisak koje druge jedinke stvaraju nad jedinkom a_x (bliže jedinke stvaraju veći pritisak). $\vec{P_x}$ je normalizirani vektor smjera u kojem treba pomaknuti jedinku a_x.

Neka d_x opisuje dobrotu jedinke a_x, neka d_{max} predstavlja najveću dobrotu jedinke iz populacije, a d_{min} najmanju dobrotu jedinke iz populacije. Tada se izrazom

$$R_x = 1 - \frac{d_x - d_{min}}{d_{max} - d_{min}} \tag{5.5}$$

može definirati otpornost jedinke a_x na pomak (otpornost najbolje jedinke je 0, te osigurava elitizam). Algoritam treba svaku jedinku a_i pomaknuti za vektor

$$\vec{V_i} = k \; R_i \; \vec{P_i}. \tag{5.6}$$

U izrazu faktor $k \in [0, \infty]$ definira jačinu pomaka, te ga se može postaviti na $k = 1$.

U navedenom algoritmu se $\overline{dist}$ u svakoj iteraciji algoritma povećava. Treba primijetiti da se bolje jedinke teže pomiču, dok se najbolja jedinka niti ne pomiče.

Algoritam je korisna pomoć drugim algoritmima kada žele u populaciju unijeti raznolikost, te pritom minimalno pogoršati kvalitetu populacije. Kako bi populacija ponovno postala raznolika, te kako bi se izašlo van nekog lokalnog optimuma, potrebno je monotonu populaciju dati navedenom algoritmu na par iteracija (korisno za npr. Harmony search). Algoritam omogućuje jedinkama da ne pretražuju sličan prostor, te je primjenjiv na algoritmima rojne inteligencije (engl. *Swarm intelligence*)[26] kao što su Ant colony optimization, Particle swarm optimization, Stochastic diffusion search i drugi.

6. Optimizacija procesiranja misli

6.1. Ljudsko ponašanje

Ljudski mozak je najsposobniji sustav koji je priroda evolucijom uspjela stvoriti. Ljudski mozak je ljudima često vrlo apstraktan, te su ga ljudi tisućljećima koristili bez previše razumijevanja načina na koji funkcionira. Tijekom prethodnih par stoljeća medicinska znanost uspijeva sve detaljnije raspoznati način na koji se dijelovi mozga povezuju i kako propagiraju informacije, dok psihologija[31] omogućuje bolje shvaćanja načina na koji se ponašaju neki složeni mehanizmi u ljudskom razumu.

Na apstraktnijoj razini se može tvrditi da razum procesira "misli"[14]. Misli su oblici formirani od razuma, za razliku od informacija koje se percipiraju pomoću osjetila. Misli se mogu definirati kao skupine informacija koja se u razumu mogu pohranjivati i obrađivati kako bi stvorile nove skupine informacija. Skupina sličnih informacija se često objedinjuje kao jedna misao, pri čemu je teško naći granicu između pojedinih misli.

Način na koji baratamo mislima (razmišljamo) često spoznajemo samoanalizom [6]. Razmišljanjem o načinu na koji smo razmišljali, povezivali asocijacije, prelazili s jedne teme razmišljanja na drugu, dosjećali se nečega što smo htjeli reći ili napraviti, i slično.

Način obrade misli se može podijeliti na svjesnu i podsvjesnu obradu[14]. Pritom svijesno razmišljanje zahtjeva određen trud, dok se podsvjesno razmišljanje često ne može kontrolirati. Donošenje kompliciranih logičkih zaključaka se uglavnom provodi svijesno.

6.2. Algoritam

Prilikom procesa razmišljanja, mogu se opaziti neke pravilnosti.

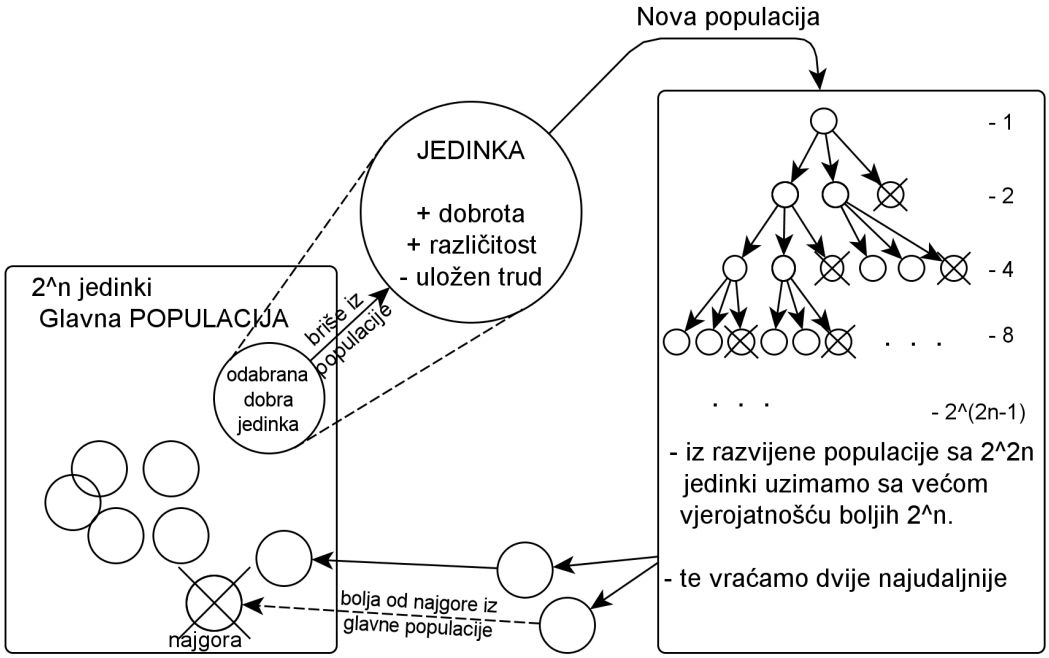
- Informacije koje su nam bitne obrađujemo svijesno.
- Prilikom obrade se početna informacija uzima iz dugoročne memorije, te se pohranjuje u kratkoročnu memorije kako bi je mogli svjesno obrađivati.
- Dohvaćanje određenih informacija uzrokuje stvaranja novih sličnih informacija.
- Raspoznajemo informacije koje već posjedujemo, te slične informacije ne obrađujemo više puta iznova.
- Rezultate razmišljanja pohranjujemo u dugoročnu memoriju, te ih kasnije po potrebi ponovno obrađujemo.
- Razmišljamo u onom smjeru u kojem se isplati, dok se isplati.
- Pamtimo količinu truda utrošenu u razmišljanje u nečemu, te sa većom vjerojatnošću razmišljamo o onome o čemu prije nismo razmišljali.

Naš razum istražuje dobra područja i istražuje ih dok je to smisleno (postoji napredak).

Treba osmisliti algoritam pretrage koji se ponaša po uzoru na ljudski razum jer se ljudski razum pokazuje vrlo uspješnim u pretraživanju nepoznatoga. Nazovimo takav algoritam Optimizacija procesiranja misli (engl. *Thought processing optimization – TPO*).

U algoritmu misao predstavlja jedinku, a svjesna obrada predstavlja istraživanje prostora pretrage. Pritom se jedinke koje su dovoljno različite pamte u dugoročnu memoriju (sakupljeno znanje), a jedinke koje se obrađuju se čuvaju u privremenoj populaciji. Prilikom odabira jedinki iz dugoročne memorije, bira se ona jedinka koja je što različitija od ostalih, što manje istraživana (nije o njoj puno "razmišljano"), te ima što veću dobrotu (korisno ju je istražiti). Odabrana jedinka se uklanja iz dugoročne memorije, te ju se obrađuje (jedinka se mutira, stvara se nova populacija). Najbolji i najrazličitiji rezultati obrade pamte se u dugoročnoj memoriji.

Algoritam se može opisati slikom 6.1.



Slika 6.1: Prikaz rada TPO algoritma

Za svaku jedinku u glavnoj populaciji pamti se njena dobrota, koliko je različita od njoj najsličnije jedinke, te koliko truda je uloženo da bi ta jedinka nastala (izraženo brojem mutacija). Algoritam prilikom inicijalizacije stvara populaciju veličine 2^n (n je parametar algoritma), koja se sastoji od slučajno stvorenih jedinki. Svaki se korak algoritma provodi na sljedeći način:

- Odaberi misao (biranje jedinke iz glavne populacije)
- Razmisli (stvori zasebnu populaciju od izabrane jedinke)
- Spremi misao (najbolje rezultate zapiši u glavnu populaciju)

Prvo se odabire jedinka koja će se obrađivati. Pritom se odabire ona jedinka koja je što različitija od ostalih jedinki iz glavne populacije, što manje obrađivana i što veće dobrote. Koliko je koji od navedena tri kriterija bitan, određeno je sa tri parametra s kojima se algoritam pokreće: faktor različitosti f_{ra}, faktor dobrote f_{do} i faktor truda f_{tr}. Selekcija se provodi pomoću ruletskog kola[7], pri čemu je veličina isječka u kolu v_i za pojedinu jedinku i definiran izrazom (6.1).

$$v_i = f_{ra}\frac{r_i - r_{min}}{r_{max} - r_{min}} + f_{do}\frac{d_i - d_{min}}{d_{max} - d_{min}} + f_{tr}(1 - \frac{t_i - t_{min}}{t_{max} - t_{min}}) \qquad (6.1)$$

U izrazu je r različitost jedinke od ostalih, d je jedinkina dobrota, a t je trud uložen u jedinkinu obradu.

Selektirana jedinka se uklanja iz populacije te se uređuje različitost r svim jedinkama iz populacije kojima je selektirana jedinka bila najsličnija. Pomoću selektirane jedinke se razvija nova populacija. Razvijanje nove populacije se provodi tako što se od selektirane jedinke stvori tri mutirane kopije, te se eliminira ona sa najlošijom dobrotom. Od svake od preživjele dvije jedinke se ponovno stvaraju tri mutirane kopije, te se ponovno eliminiraju jedinke sa najlošijom dobrotom. Proces nastajanja jedinki se može prikazati pomoću binarnog stabla, te se provodi sve do $(2n - 1)$. razine stabla. Na taj način nastaje ukupno $2^{2n} - 1$ jedinki koje se sortiraju po dobroti, kako bi se izvršila nova selekcija kojom se izabire najviše 2^n jedinki. Selekcija se vrši ruletskim kolom kojemu veličina isječka ovisi o ranku jedinke.

Između odabranih 2^n jedinki, izračuna se sličnost između svake dvije, te se odaberu one dvije koje su međusobno najrazličitije. Bolje od te dvije jedinke se odmah doda u glavnu populaciju (na mjesto gdje se nalazila jedinka izvađena zbog selekcije), dok se gora jedinka zamjenjuje sa najgorom jedinkom u glavnoj populaciji, ukoliko je bolja od nje. Prilikom ubacivanja jedinki u glavnu populaciju, ubačenim jedinkama se izračuna različitost od najsličnije jedinke u populaciji, te se ostalim jedinkama ažurira različitost sa upravo ubačenim jedinkama.

Pomoću navedenog postupka glavna populacija ostaje različita, te se stalno popravlja.

Obje selekcije u algoritmu podržavaju "elitizam" najbolje jedinke.

Algoritam oponaša jednu iteraciju ljudskog razmišljanja.

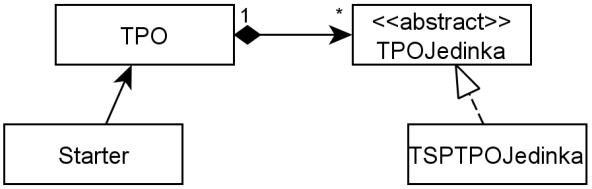
6.3. Ostvareni algoritam

Za potrebe ispitivanja osmišljenog algoritma, u Javi je razvijen paket "com.kusalic.TPO" koji ostvaruje navedeni algoritam.

Od bitnijih klasa u paketu, vrijedi izdvojiti četiri koje su prikazane UML dijagramom na slici 6.2.

Klasa Starter ostvaruje grafičko sučelje pomoću kojega se može pokretati algoritam i pratiti rezultate, prikazane u grafičkom obliku.

Klasa TPO ostvaruje algoritam, te objedinjuje populaciju jedinki opisanih apstraktnom klasom TPOJedinka. Klasa TPO sadrži mnoštvo javnih metoda



Slika 6.2: UML dijagram bitnijih klasa paketa "com.kusalic.TPO"

pomoću kojih se mogu pratiti razne statistički bitne informacije o izvršavanju programa. Algoritam se inicijalizira stvaranjem instance klase, dok se za provedbu pojedinog koraka koristi javna metoda step(). Od bitnijih privatnih metoda klase TPO treba istaknuti selektiraj(), razmisli(x) i zapamti(a,b). Metoda selektiraj() provodi selekciju u složenosti $O(N)$ ($N = 2^n$ je veličina glavne populacije), te u amortiziranoj složenosti[16] $O(N)$ uređuju različitosti jedinkama kojima je selektirana jedinka bila najsličnija. Metoda razmisli(x) razvija populaciju od jedinke x. Metoda razvija populaciju u složenosti $O(N^2)$, vrši sortiranje u složenosti $O(N^2 log\ N^2)$, te računa dvije najrazličitije jedinke u složenosti $O(N^2)$. Metoda zapamti(a,b) ubacuje jedinke a i b u glavnu populaciju, te pritom u amortiziranoj složenosti $O(N)$ uređuje različitost jedinki. Inicijalizacija algoritma se provodi u složenosti $O(N^2)$; zato je pomoćna populacija kvadratno veća od glavne. Navedene složenosti pretpostavljaju izvršavanje metoda jedinke u složenosti $O(1)$.

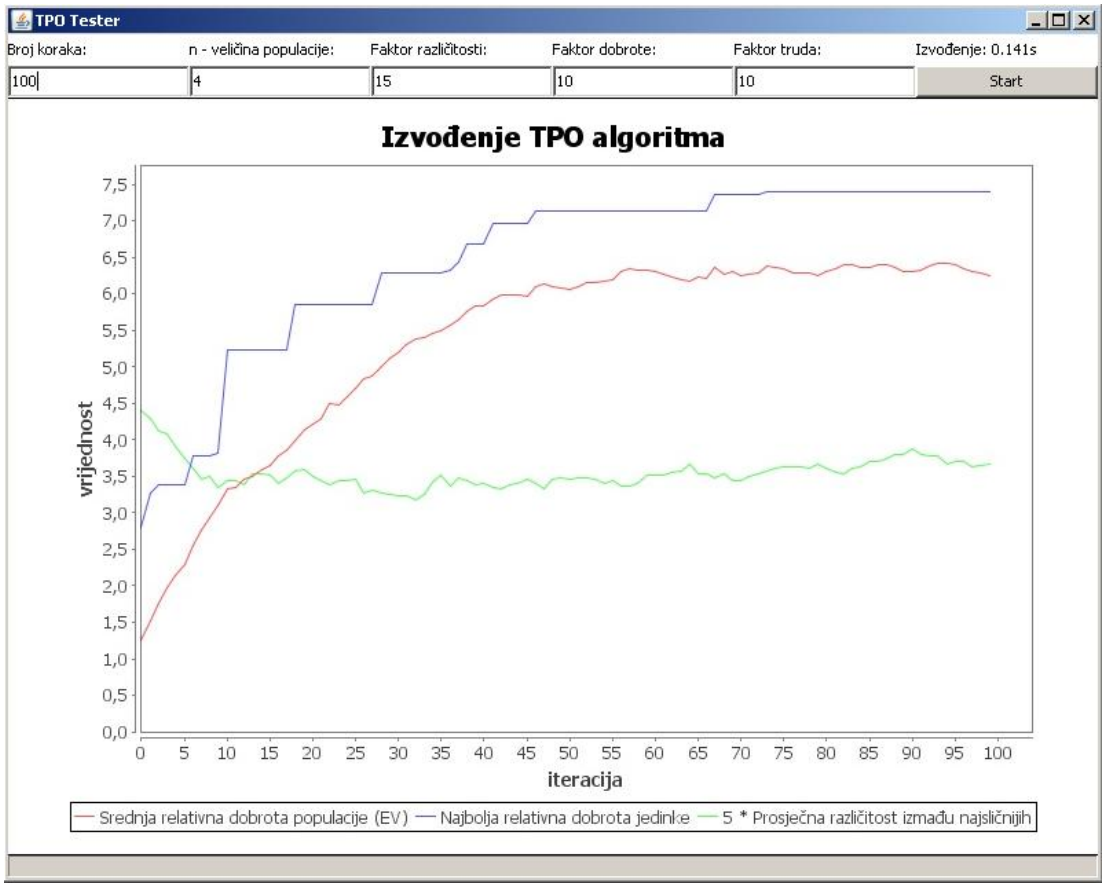
Apstraktna klasa TPOJedinka ostvaruje sve što je vezano za jedinku a ne tiče se samog kromosoma, kako bi algoritam bio što lakše ostvariv za razne probleme.

Klasa TPSTPOJedinka ostvaruje TSP problem na isti način kao što je to učinjeno za GGA algoritam, te za potrebe TPOJedinke sadrži metode dobrota(), mutacija(), kopiraj(x), slicnost(x) i slucajnoPopuni(). Različitost između dvije jedinke je definirana kao $1 - slicnost$.

Detaljniji opis klasa i pridruženih metoda dan je u kodu u obliku *javadoc* dokumentacije.

Ostvareni sustav je izrazito lako nadogradiv za rješavanje novih problema.

Na slici 6.3 je dan primjer izvođenja programa sa glavnom populacijom veličine 16 jedinki. Grafičko okružje programa je lako za korištenje, te većim dijelom odgovara okružju korištenom pri ostvarivanju GGA algoritma. Najbolja relativna dobrota (plavo) i srednja relativna dobrota (crveno) se prikazuju na y osi tako da se očekivana dobrota prosječne jedinke nalazi na $y = 1$. Vrijednost v na y osi



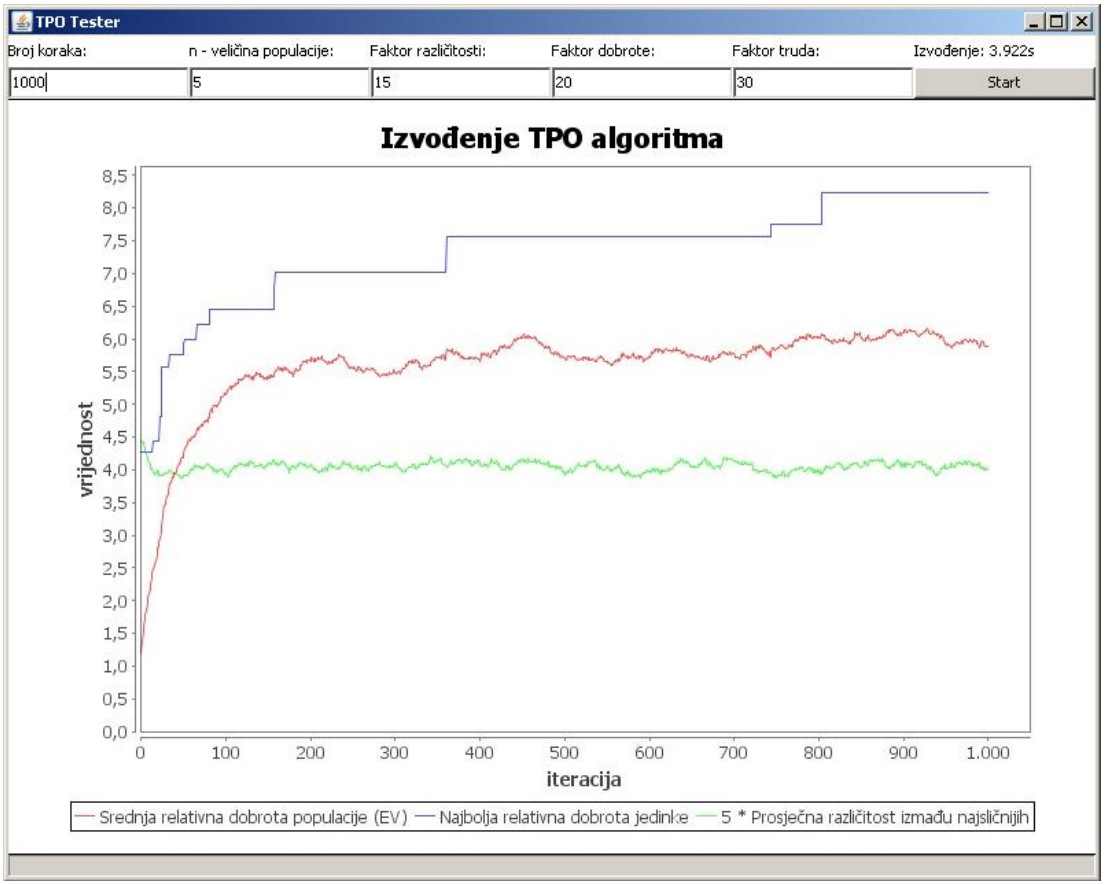
Slika 6.3: Primjer izvođenja algoritma sa glavnom populacijom od 16 jedinki

predstavlja popravak rješenja za $v - 1$ standardnu devijaciju slučajne populacije rješenja. Zelenom bojom je prikazana prosječna različitost jedinki, reskalirana sa intervala $[0, 1]$ na interval $[0, 5]$ kako bi se mogla bolje vidjeti.

Na slici 6.4 je moguće vidjeti primjer ponašanja algoritma za populaciju od 32 jedinke, te 1000 koraka algoritma. Kao što je zelenom bojom prikazano, glavna populacija uspijeva održavati različitost dok algoritam povećava dobrotu jedinki.

U algoritmu se selektirana dobra jedinka, nakon procesa razmišljanja, zamijenjuje sa dvije jedinke koju su donekle slične, te su nastale nakon većeg broja mutacija, pa zato dodane jedinke neko vrijeme ne bivaju ponovno odabrane. Previše obrađivane jedinke se sa manjom vjerojatnošću ponovno biraju jer su nastale nakon puno mutacija.

Pomoću veličine populacije i dodatna tri faktora može se podešavati ponašanja algoritma.



Slika 6.4: Primjer izvođenja algoritma sa glavnom populacijom od 32 jedinke

- Faktorom različitosti f_{ra} se upravlja brzinom konvergacije cijelog algoritma.
- Faktorom dobrote f_{do} se upravlja brzinom konvergacije prema lokalnim optimumima.
- Faktorom truda f_{tr} se odlučuje o ujednačenosti istraživanja prostora (potrebno znanje o domeni pretrage).

Navedeni algoritam se izrazito lagano paralelizira. Svaki proces može zasebno provoditi razmišljanje, dok jedine kritične odsječke koda predstavlja komunikacija sa glavnom populacijom.

7. Daljnja istraživanja i zanimljiva ljudska ponašanja

U nastavku će biti iznesene neke ideje zasnovane ne ljudskom ponašanju koje nisu našle mjesta u drugim poglavljima ovog rada.

Ljudi prilikom kretanja kroz prostor pokazuju mnoga zanimljiva ponašanja. Dobro znamo da je najkraći put pri kretanju kroz park ravna linija, međutim, ljudi se gotovo nikad ne kreću po ravnoj liniji, čak ni onda kada im se jako žuri. Slika 7.1 prikazuje putove koje su utabali ljudi kretajući se kroz park preko puta KD "Vatroslav Lisinski" u Zagrebu. Na slici je crvenom bojom naglašena utabana putanja.

Slika 7.1: Ljudsko kretanje kroz prostor

Znajući da su ljudi vrh evolucije i da bi se trebali ponašati optimalno, ljudsko kretanje nas može začuditi. Ono pokazuje da nema potrebe uvijek pronaći optimalnu procjenu, nego je često i osrednja procjena sasvim dobra smjernica kako bi mogli usmjeriti pretragu. Ponavljanje procjenjivanja dobrote za kompleksne

probleme često je najviše ograničavajući dio genetskog algoritma. U problemima iz stvarnog svijeta, poput strukturalnog optimiziranja, jedan poziv funkcije dobrote može zahtijevati nekoliko sati ili čak nekoliko dana izračunavanja.[22] U takvim slučajevima potrebno je zanemariti potpunu procjenu, te se ponašati kao što to ljudi čine prilikom biranja optimalnog puta kroz prostor – koristiti jednostavniju aproksimaciju funkcije dobrote.

Ljudi često prilikom pretraživanja, na temelju grubih informacija o području pretrage, donose zaključke o tome što ne treba pretraživati, umjesto zaključaka o tome gdje treba usmjeriti pretragu. Takva ponašanja su često primjetna pri traženju određenog artikla u trgovini. Pri hodanju između polica, radimo eliminaciju područja gdje se traženi artikl vjerojatno ne nalazi. Kada područje koje istražujemo pokaže više potencijala, tada malo detaljnije pregledamo police, pokušavajući što prije eliminirati područje koje gledamo. Tek na kraju, kada smo eliminirali ostatak trgovine, uspijevamo dovoljno detaljno pogledati onu policu na kojoj se nalazi traženi artikl, te ga uočavamo.

Ovakva sofisticirana metoda eliminacije radi grube procjene o većim područjima pretrage, kako bi uštedjela vrijeme, te kasnije manjim potencijalnijim područjima posvetila više pažnje.

Sličnu tehniku koristimo i prilikom razmišljanja. Ukoliko zaključimo da neka ideja nema previše potencijala, ne razrađujemo je dalje, nego utrošimo vrijeme u traženje novih.

Kaskadiranjem istog eliminacijskog algoritma sa različito podešenim parametrima može se postići znatno poboljšanje složenosti, često sa $O(n)$ na $O(log\ n)$. Oblikujemo li algoritam koji pokušava na temelju malo informacija eliminirati par posto najgorih područja, te zatim sa malo više informacija eliminirati novih par posto najgorih područja, te nastavimo to raditi, uskoro imamo malo dobro područje pretrage kojemu možemo posvetiti puno paženje.

U okviru ovog rada, u paralelnoj obradi za GA/GGA, spomenuta je “mutacija neključnog gena”.

Pojedina jedinka se može više ili manje promijeniti ukoliko joj je mutiran slučajno odabrani gen. Često mutacija može znatno promijeniti jedinku, kako

bi unijela malo genetske raznolikosti u populaciju. Mutacija neključnog gena je vođena idejom mijenjanja onog gena koji neće puno promijeniti dobrotu jedinke. Ukoliko se na jedinki pronađe gen koji se smije mutirati bez da joj se primjetno promijeni dobrotu, uspijeva se "besplatno" unijeti genetsku raznolikost u populaciju. Mutacijom neključnog gena niti se kvari jedinka, niti ju se tjera u lokalni optimum iz kojeg će teško izaći. Pri paralelnoj izvedbi, gdje su računalni resursi dovoljno veliki, može se pokazati vrlo korisnim uvesti mutaciju neključnog gena.

8. Zaključak

U okviru ovog rada su istraženi i osmišljeni novi evolucijski algoritmi inspirirani ljudskim psihosocijalnim ponašanjem. Osmišljeni algoritmi su logički i matematički opisani, te programski ostvareni i ispitani. Osmišljen je, ostvaren i ispitan generalizirani genetski algoritam uz pomoć koje se bolje pristupa optimiranju količine informacija o procesu pretrage, kako bi se pretraga usmjerila u što povoljnijem smjeru. Rezultati usporedbe osmišljenog algoritma sa genetskim algoritmom pokazuju da osmišljeni algoritam za isto procesorsko vrijeme uspijeva pronaći za jednu do dvije devijacije bolje rješenje, te da je otporniji na zapadanje u lokalne optimume prostora pretraga. Osmišljen je algoritam osobnog prostora koji služi drugim algoritmima kao pomoć pri održavanju populacije raznolikom. Osmišljen je, ostvaren i ispitan algoritam optimizacije procesiranja misli, koji upravlja populacijom jedinki na načinom kojim ljudski razum upravlja misaonim procesima. Algoritam ne koristi operator mutacije, te je zato lakše ostvariv za specifične probleme pretrage. Rezultati analize performansi osmišljenog algoritma pokazuju da algoritam uspijeva održavati populaciju raznolikom, te pritom neprestano poboljšavati kvalitetu prosječnog i najboljeg rješenja. Osmišljena je prilagodba ostvarenih algoritama paralelnoj izvedbi, te su ukazane prednosti pri paralelnoj izvedbi navedenih algoritama. Na kraju su dane ideje za daljnja istraživanja.

Literatura

[1] *TSPLIB*, 2010. URL `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`.

[2] John Alcock. *The Triumph of Sociobiology.* Oxford University Press, 2001.

[3] Allen Pease, Barbara Pease. *Velika škola govora tijela.* Mozaik knjiga, 2008.

[4] Andrew Binstock, Jon Rex. *Practical Algorithms for Programmers.* Addison-Wesley Professional, 1995.

[5] Thomas Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford Univ. Press., 1996.

[6] Elliot Aronson, Timothy D. Wilson, Robin M. Akert. *Socijalna psihologija.* Gospodarska Misao, 2005.

[7] Marin Golub. *Genetski algoritam, skripta – 1. dio*, 1997. URL `http://www.zemris.fer.hr/~golub/ga/ga_skripta1.pdf`.

[8] Knapp, Mark L. *Social intercourse: from greeting to goodbye.* Allyn and Bacon, 1978.

[9] Maynard J. Smith. *The Theory of Evolution.* Cambridge University Press, 1993.

[10] Milena Karova, Vassil Smarkov, Stoyan Penev. Genetic operators crossover and mutation in solving the tsp problem. *CompSysTech'2005*, 2005.

[11] Chris Miles. *Symmetrical Traveling Salesperson Problem*, 2010. URL `http://www.cse.unr.edu/~miles/tsp/tsp.html`.

[12] Neven Elezović. *Statistika i procesi*. Element, 2007.

[13] Robert Sedgewick. *Algorithms in C*. Addison-Wesley, 1990.

[14] Ruth Snowden. *Teach Yourself Freud*. McGraw-Hill, 2006.

[15] Sanghamitra Bandyopadhyay & Sankar K. Pal. *Classification and Learning Using Genetic Algorithms*. Springer, 2007.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms, Second Edition*. MIT Press, McGraw-Hill, 2001.

[17] Wikipedia. *Ant colony optimization*, 2010. URL `http://en.wikipedia.org/wiki/Ant_colony_optimization`.

[18] Wikipedia. *Epistemology*, 2010. URL `http://en.wikipedia.org/wiki/Epistemology`.

[19] Wikipedia. *Particle swarm optimization*, 2010. URL `http://en.wikipedia.org/wiki/Particle_swarm_optimization`.

[20] Wikipedia. *Evolutionary algorithm*, 2010. URL `http://en.wikipedia.org/wiki/Evolutionary_algorithm`.

[21] Wikipedia. *Evolutionary computation*, 2010. URL `http://en.wikipedia.org/wiki/Evolutionary_computation`.

[22] Wikipedia. *Genetic algorithm*, 2010. URL `http://en.wikipedia.org/wiki/Genetic_algorithm`.

[23] Wikipedia. *Gradient descent*, 2010. URL `http://en.wikipedia.org/wiki/Gradient_descent`.

[24] Wikipedia. *Hill climbing*, 2010. URL `http://en.wikipedia.org/wiki/Hill_climbing`.

[25] Wikipedia. *No free lunch in search and optimization*, 2010. URL `http://en.wikipedia.org/wiki/No_free_lunch_in_search_and_optimization`.

[26] Wikipedia. *Swarm intelligence*, 2010. URL `http://en.wikipedia.org/wiki/Swarm_intelligence`.

[27] Wikipedia. *Speciation*, 2010. URL `http://en.wikipedia.org/wiki/Speciation`.

[28] Wikipedia. *Interval tree*, 2010. URL `http://en.wikipedia.org/wiki/Interval_tree`.

[29] Wikipedia. *Segment tree*, 2010. URL `http://en.wikipedia.org/wiki/Segment_tree`.

[30] Wolpert, D.H., Macready, W.G. *No Free Lunch Theorems for Optimization*, 1997.

[31] Predrag Zarevski. *Psihologija pamćenja i učenja*. Slap, 2001.

Evolucijski algoritmi inspirirani ljudskim psihosocijalnim ponašanjem

Sažetak

Autor: Domagoj Kusalić

Evolucijski algoritmi su metaheuristički optimizacijski algoritmi zasnovani na populaciji jedinki. Evolucijski algoritmi koriste neke mehanizme inspirirane biološkom evolucijom kako bi njenim oponašanjem pokušali usmjeriti pretragu domene problema u povoljnom smjeru. Postojeći evolucijski algoritmi su uglavnom zasnovani na oponašanju evolucije i raznih postupaka primjetnih u živom svijetu. Budući da čovjek predstavlja vrh evolucije, logično se postavlja pitanje može li se oponašanjem čovjekovih psiholoških i sociooloških postupaka razviti korisni optimizacijski algoritmi.

U ovom radu se pozitivno odgovara na dano pitanje istraživanjem i razvijanjem nekoliko algoritama osnovanih na ljudskom ponašanju. Osmišljen je generalizirani genetski algoritam uz pomoć kojeg se maksimizira količina informacija o stanju procesa pretrage, kako bi se pretraga usmjerila u što povoljnijem smjeru. Algoritam je programski ostvaren, ispitan, uspoređen sa postojećim genetskim algoritmom, te analiziran. Rezultati usporedbe sa postojećim algoritmom pokazuju da osmišljeni algoritam ima znatno bolje performanse. U radu je osmišljen pomoćni algoritam osobnog prostora koji omogućuje drugim algoritmima održavanje populacije raznolikom, kako ne bi previše rano konvergirali prema lokalnom optimumu. Osmišljen je i algoritam optimizacije procesiranja misli, koji upravlja populacijom jedinki na način kojim ljudski razum upravlja misaonim procesima. Algoritam je programski ostvaren, ispitan i analiziran. Rezultati analize performansi osmišljenog algoritma pokazuju da algoritam uspijeva održavati raznolikost populacije, te pritom neprestano poboljšavati kvalitetu rješenja. Osmišljena je prilagodba ostvarenih algoritama paralelnoj izvedbi, te su dane dodatne ideje za daljnja istraživanja.

Ključne riječi: evolucijski algoritmi, optimizacija, ljudsko ponašanje, misli

Evolutionary algorithms inspired by human psychosocial behavior

Abstract

Author: Domagoj Kusalić

Evolutionary algorithms are population-based metaheuristic optimization algorithms. Evolutionary algorithms try to mimic some mechanisms inspired by biological evolution, so they could direct the search in a favorable direction. Existing evolutionary algorithms are mainly based on imitating the evolution and various others procedures noticeable in the living world. Since humans are the top of evolution, it is logical to question whether it is possible to develop useful optimization algorithms by imitating human psychological and social behavior.

In this research a positive answer is given by developing several algorithms based on human behavior. Designed generalized genetic algorithm tries to maximize the amount of information given about the state of the search process, in order to direct the search in the favorable direction. The algorithm is implemented, tested, compared with the existing genetic algorithms, and analyzed. Comparing with the existing algorithm shows that the designed algorithm has better performance. Designed is supplementary personal space algorithm that allows other algorithms to maintain diverse populations in order to avoid too early convergation towards local optimum. Also designed is thoughts processing optimization algorithm, which manages a population of individuals modeled in a way that human mind controls thought processes. Algorithm was implemented, tested and analyzed. The results of performance analysis of designed algorithm shows that the algorithm is able to maintain diverse population, and simultaneously constantly improve the quality of solutions. Adjustments for paralel implementation are proposed, and additional ideas for further research are given.

Keywords: evolutionary algorithms, optimization, human behavior, thoughts

Životopis autora

Domagoj Kusalić je rođen 28.04.1988. u Osijeku. Kroz školovanje, pored nebrojenih zapaženih rezultatima na državnim natjecanjima iz robotike, logika, matematike i informatike, treba nadodati rezultate sa mnogih međunarodnih natjecanja iz informatike, među kojima je i brončana medalja sa Međunarodne informatičke olimpijade 2007. godine. Osim natjecanja, pored dvije plakete Zajednice tehničke kulture Osječko baranjske županije i dva priznanja izvrsnosti od Agencije za znanost i obrazovanje RH, ističu se i dva izravna upisa na FER, na kojemu trenutno studira na smjeru Računarska znanost. Domagoj u slobodno vrijeme izučava svakakve stvari iz struke, dovršava pisanje knjige o naprednim algoritmima i strukturama podataka, te se druži sa drugim studentima.