SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1633

PODEŠAVANJE PARAMETARA GENETSKOG ALGORITMA

Vedran Lovrečić

Zagreb, listopad 2006.

Sažetak. U ovom su radu opisane osnove genetskih algoritama. Naglasak je stavljen na parametre genetskih algoritama. Zatim je opisan problem trgovačkog putnika. Opisano je nekoliko genetskih operatora za rješavanje problema trgovačkog putnika. Programski je ostvaren genetski algoritam za rješavanje problema trgovačkog putnika s 3-turnirskom eliminacijskom selekcijom, jednostavnom mutacijom i uniformnim križanjem. Eksperimentalno su određene ovisnosti između parametara veličine populacije, broja iteracija i vjerojatnosti mutacije.

Abstract. In this document are described basics of the Genetic Algorithms. Accent is placed on Genetic Algorithm parameters. Then Traveling Salesman Problem is described. Several genetic operators for solving Traveling Salesman Problem are described. Application for solving Traveling Salesman Problem using Genetic Algorithms is developed with 3-tournament elimination selection, simple mutation and uniform crossover. Correlations between population size, number of iterations and mutation probability are experimentaly deterimned.

Sadržaj

1. Uvod

Genetski algoritam je heuristička metoda koja služi za rješavanje optimizacijskih problema. Kombinira slučajno i usmjereno pretraživanje prostora rješenja. Princip rada genetskog algoritma analogan je evolucijskom procesu u prirodi. Iz početne se populacije korištenjem genetskih operatora evoluira optimalno rješenje. Za svaku vrstu problema potrebno je dizajnirati posebni genetski algoritam ili je potrebno prilagoditi problem nekom već postojećem genetskom algoritmu. Evoluciju rješenja korištenjem genetskog algoritma moguće je usmjeravati podešavanjem parametara koje neki genetski algoritam može imati. Izbor parametara važan je korak u dizajnu i korištenju nekog genetskog algoritma. O parametrima ovisi koliko će se vremena potrošiti na evoluciju rješenja, kojom će se brzinom algoritam usmjeravati prema potencijalnom optimumu, hoće li pretraživati veći ili manji dio prostora rješenja, itd. Set parametara koji za jedan genetski algoritam daje kvalitetne rezultate, za neki drugi genetski algoritam ne mora dati jednako kvalitetne rezultate. Genetski algoritam može postići jednako kvalitetna rješenja za više različith skupova parametara. Stoga se pokušava pronaći način na koji bi se mogli odrediti optimalni parametri za što veću grupu genetskih algoritama. Pokazalo se da je to jako teško ili nemoguće. Ipak postoje određene veze između parametara genetskog algoritma koje se u ovome diplomskom radu želi potvrditi na problemu trgovačkog putnika.

Veličina populacije, broj iteracija i vjerojatnost mutacije su tri parametra koje koriste svi genetski algoritmi prilikom evolucije rješenja. U ovome radu napravljena je analiza podešavanja tri navedena parametra korištenjem genetskog algoritma s 3-turnirskom selekcijom, PMX križanjem i mutacijom koja mijenja svaki gen unutar kromosoma s određenom vjerojatnošću.

Problem trgovačkog putnika je problem koji se danas koristi u raznim granama industrije kako bi se rješili optimizacijski problemi koji u svojoj srži sadržavaju najkraći obilazak točaka u grafu prema nekom kriteriju (to može biti najkraći put, najkraće vrijeme, ...). Kako se radi o problemu koji je NP težak, genetski algoritam se nameće kao metoda koja bi mogla biti uspješna u njegovu rješavanju. Zato je problem trgovačkog putnika izabran za problem na kojemu će se provjeriti povezanost i odnosi između parametara genetskog algoritma.

2. Genetski algoritmi

2.1. Osnove genetskog algoritma

Genetski algoritam je metoda optimizacije koja se temelji na prirodnoj evoluciji. Jedan od temelja genetskog algoritma je ideja o preživljavanju najboljih jedinki u populaciji. Preživljavanjem najboljih jedinki, genetski algoritam se usmjerava prema optimumu i pretražuje samo one dijelove prostora rješenja koji sadržavaju dobra rješenja. U prirodi preživljavaju bolje, jače i sposobnije jedinke, i veća je vjerojatnost da će se te jednike pariti kako bi nastali potomci koji će tvoriti slijedeću generaciju. Isto tako, potomci bi trebali biti bolji od roditelja baš zbog toga jer su nastali od sposobnijih jedniki.

Genetski algoritam radi na istom principu. Prva generacija se slučajno odabire. Svaka sljedeća generacija se popunjava križanjem najboljih jedniki iz trenutne populacije. Genetskim operatorima se stvaraju bolje jedinke sve dok se ne dođe do zadovoljavajućeg rezultata. Kako se ne bi dogodilo da se završi u nekom lokalnom optimumu, genetski algoritam koristi i operator mutacije koji simulira “nezgodu“ u stvarnoj evoluciji, jedinku koja se ne uklapa u populaciju. Razlog ovome je slučajan skok na neko još neistraženo područje prostora rješenja u kojem se možda nalazi bolje rješenje od onih trenutno sadržanih u populaciji. Ako se to ne dogodi, već se dogodi lošije rješenje, selekcija će se pobrinuti da se to rješenje brzo zamijeni s onim boljima.

Rad genetskog algoritma može se najbolje opisati sa slijedećih pet koraka [2]:

1. kodiranje jedinki
2. evaluacija dobrote jedinki
3. selekcija
4. križanje
5. mutacija
6. dekodiranje rješenja

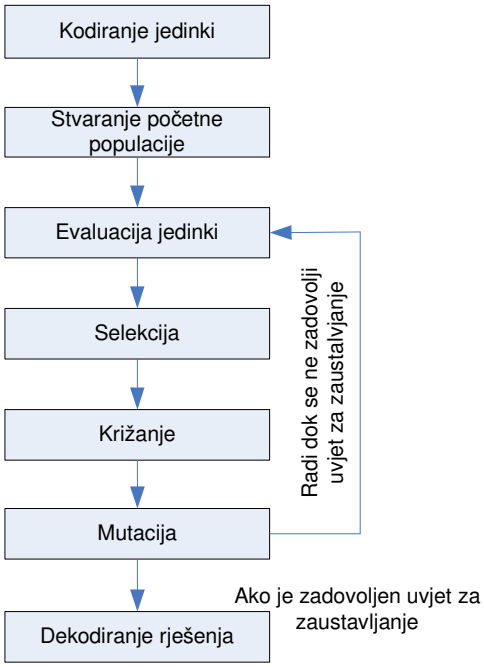
Kodiranje jedinki je bitan početni korak u dizajnu genetskog algoritma. Svaka jedinka mora biti jedinstveno predstavljena u populaciji. Praktična izvedba i načini rada genetskih operatora također ovise o kodiranju jedinki. Nakon što se odredi kako jedinke izgledaju, najčešće se slučajnim odabirom generira početna populacija u kojoj se za svaku jedinku računa (evaluira) dobrota.

Dobrota je mjera koja genetskom algoritmu govori je li neko rješenje bliže ili dalje od optimuma u usporedbi s ostalim članovima populacije. Oni članovi koji imaju bolju dobrotu su bliži optimumu i za njih je veća vjerojatnost križanja.

Križanje je postupak u kojem se odabiru dvije jedinke iz populacije te se njihovi dijelovi međusobno kombiniraju kako bi nastale nove jedinke. Novonastale jedinke se umeću u slijedeću generaciju umjesto onih lošijih.

Neke od novonastalih jedinki se još dodatno i mutiraju kako bi se proširio prostor rješenja koji se pretražuje. Na primjer, ako se uzme da je jedna jedinka kodirana kao niz znakova, mutacija je zamjena nekog slučajno odabranog znaka drugim slučajno odabranim znakom.

Ovaj postupak se ponavlja sve dok se ne dostigne neki unaprijed definirani kriterij zaustavljanja nakon kojeg se najbolje postignuto rješenje dekodira. To rješenje je najbliže optimumu ili je optimum. Na slici 2.1 se nalazi detaljniji opis rada genetskog algoritma.



SLIKA 2.1 – Princip rada genetskog algoritma

2.2. Primjer rada genetskog algoritma

Ovaj jednostavan primjer rada genetskog algoritma je ovdje samo za ilustraciju kako bi na jednostavan i razumljiv način prikazao osnovne principe rada genetskog algoritma. Za rješavanje ovog primjera postoje metode koje su znatno brže i efikasnije u pronalasku rješenja.

Cilj je pronaći minimum funkcije $f(x) = x^2 - 4x + 5$ na skupu cijelih brojeva od {0,...,15}. Traženjem minimuma klasičnim metodama pokazuje se da se minimum postiže za x = 2.

2.2.1. Kodiranje jedinki

Određivanje na koji će način jedinke biti kodirane je važan dio dizajna genetskog algoritma. Svako moguće rješenje iz prostora rješenja treba biti jedinstveno predstavljeno. S obzirom

na to da se ovdje koriste samo cijeli brojevi iz domene {0,...,15} najbolje je jedinku prikazati binarno, to jest kao niz jednica i nula. Za neki drugi problem može se iskoristiti i drugačiji način prikaza, kao što će se vidjeti na primjeru problema trgovačkog putnika. Broj 5 se može prikazati kao 101 ili 0101, broj 14 kao 1110. Razlog dodavanja 0 ispred broja 5 je postizanje jednake dužine svih jedinki. Broj ovako predstavljen naziva se kromosom, a svaka pojedina jedinica ili nula se naziva gen. U općenitom slučaju gen ne mora biti nula ili jedinica. U problemu trgovačkog putnika gen je jedan grad, a kromosom je skup gradova. Nakon određivanja kodiranja generira se početna populacija slučajnim odabirom brojeva iz domene problema.

2.2.2. Evaluacija jedinki

Evaluacija jedniki služi kako bi se odredilo koje su jedinke bolje od ostatka populacije kako bi se te jedinke mogle križati u nadi da će se dobiti bolja rješenja. S obzirom na to da se u funkciji *f* traži minimum funkcija dobrote je jednostavno *f(x)* sa slijedećim značenjem, što je vrijednost *f(x)* manja to je *x* bolja jedinka (kromosom). Ako se za *x* uzmu vrijednosti 5 i 14 tada su njihove vrijednosti *f(x)*

$$f(5) = 10$$

$$f(14) = 145$$

Očito je da je 5 bolja jedinka zato jer ima manju vrijednost *f(x).* Stoga se i kaže da 5 ima bolju dobrotu od 14. Dobrota se koristi kako bi se odredila vjerojatnost križanja za svaku pojedinu jedinku. Bolje jedinke imaju veću vjerojatnost križanja.

Detaljniji opisi kako se sve može definirati dobrota, kao i načini određivanja vjerojatnosti dostupni su u [1].

U postupku evaluacije gleda se je li zadovoljen kriterij zaustavljanja genetskog algoritma. Kriterij zaustavljanja se može definirati na više načina. Neki od najpopularnijih su: zaustavljanje algoritma nakon određenog broja iteracija, zaustavljanje kada se najbolje rješenje nije promijenilo unaprijed definiran broj generacija, zaustavljanje kada je prosječna dobrota generacije jako blizu dobroti najbolje jedinke.

Najčešće se koristi više kriterija za zaustavljanje, a ne samo jedan. Ovi kriteriji uvijek zaustavljaju algoritam nakon nekog vremena bez obzira na to je li pronađeno optimalno rješenje ili ne.

2.2.3. Križanje

U ovom primjeru, operacija križanja definirana je na sljedeći način: slučajno se odaberu dva kromosoma iz populacije, slučajno se odabere točka križanja, te se zamijene svi geni iza točke križanja u oba kromosoma. Geni iza točke križanja u prvom kromosomu prijeđu u drugi, a geni iz drugog kromosoma prijeđu u prvi.

Ako je prvi kromosom $k_1 = 5$, a drugi $k_2 = 14$ tada su njihove kodirane verzije

$$k_1 = 0101 \text{ i}$$

$$k_2 = 1110$$

a djeca koja nastaju njihovim križanjem, s točkom prekida postavljenom iza drugog gena

$$k_1 = 01|01 \text{ i}$$

$$k_2 = 11|10,$$

djeca su

$$k_1' = 0110 = 6 \text{ i}$$

$$k_2' = 1101 = 13$$

Ovako dobiveni kromosomi su spremni za dodavanje u novu populaciju, to jest u slijedeću generaciju.

U ovom primjeru koristi se samo jedna točka križanja, no moguće ih je koristiti i više, ovisno o duljini kromosoma. Slijedeći primjer pokazuje što bi se dobilo s dvije točke prekida.

Prvi kromosom $k_1 = 5$, drugi $k_2 = 14$ i točke prekida iza prvog i trećeg gena

$$k_1 = 0|10|1 \text{ i}$$

$$k_2 = 1|11|0$$

novonastala djeca

$$k_1' = 0111 = 7 \text{ i}$$

$$k_2' = 1100 = 12$$

Oni kromosomi koji imaju bolju dobrotu imaju i veću vjerojatnost ulaska u postupak križanja i ostavljanja svog genetskog materijala za sljedeće generacije. Križanjem kromosoma visoke vrijednosti funkcije dobrote postoji dobra vjerojatnost da će novonastali kromosomi imati bolju vrijednost funkcije dobrote od svojih roditelja. Time je postignut cilj stvaranja sve boljih i boljih rješenja u novijim generacijama.

Postupak dodavanja kromosoma može biti takav da se od dvoje dobivene djece bolje dodaje u novu populaciju na mjesto lošijeg roditelja ili da se nova populacija gradi od nule pa da se kromosomi iz stare populacije križaju sve dok ne popune sva mjesta u novoj populaciji sa svojom djecom.

Identična jedinka može nastati križanjem više različitih parova roditelja. To znači da se u populaciji može nalaziti više kopija iste jedinke. Genetski algoritam može provjeravati nalazi li se novonastala jedinka već otprije u populaciji i takvu jedinku ne dodati u populaciju. Ovakvim se pristupom dobiva na raznolikosti populacije. Ne može postojati niti jedan dupli kromosom u bilo kojoj generaciji. Na kraju će postojati samo jedno optimalno rješenje, sva ostala će biti lošija. Ovo predstavlja problem ako se za kriterij zaustavljanja uzme sličnost prosječne dobrote populacije i dobrote najbolje jedinke, zato jer te sličnosti nema. Genetski algoritam će imati problema s odlukom kada stati. Zato je dobro koristiti više uvjeta zaustavljanja ili one uvjete koji su neovisni od ovakvih problema.

Najbolju jedinku iz trenutne generacije je moguće direktno ubaciti u sljedeću generaciju. U svakoj generaciji će se uvijek nalaziti do tada najbolje dobiveno rješenje.

Prilikom definiranja metode križanja jako je bitno da se obrati dodatna pozornost na ispravnost novonastalih jedinki. Genetski algoritam treba biti siguran da se novonastale jedinke nalaze u prostoru rješenja. U ilustrativnom primjeru, ako se domena smanji na {0,...,10} tada se treba obaviti dodatna provjera zato jer kromosom 1110 = 14 nije u domeni, a 1010 = 10 jest. Oba koriste 4 bita za prikaz, ali 14 izlazi van prostora rješenja.

2.2.4. Mutacija

Prilikom križanja može se dogoditi da se algoritam zaglavi u lokalnom optimumu. To znači da se najbolja rješenja, ona koja se najviše križaju, nalaze u neposrednoj blizini nekog lokalnog optimuma. U novijim se generacijama generiraju samo ona moguća rješenja koja se isto nalaze u okolici tog lokalnog optimuma. Stoga postoji operator mutacije koji opet na slučajan način uzima neki kromosom iz novonastale generacije i njega mutira. Mutacija se radi tako da se odabere jedan slučajni gen kojemu će se zamijeniti vrijednost. Ako mu je vrijednost bila 0 tada će postati 1, i obrnuto.

Za kromosom k = 5 = 0101, slučajni gen koji će mutirati neka je drugi gen, novonastali kromosom ima vrijednost k = 0001 = 1.

Ovo je najjednostavniji mogući oblik mutacije, u kojemu se samo mijenja vrijednost slučajno odabranog gena. Mutacija može biti znatno složenija i operator mutacije se dizajnira u skladu s problemom i prikazom kromosoma za određeni problem.

Osim operatora mutacije moguće je koristiti i operator inverzije, što je zapravo jedna podvrsta mutacije. U kromosomu se izaberu dvije točke invertiranja i promijeni se poredak gena između tih točaka.

Na primjer, neka su točke invertiranja jedan i tri, a kromosom neka je k = 0101.

$$k = 0|10|1$$

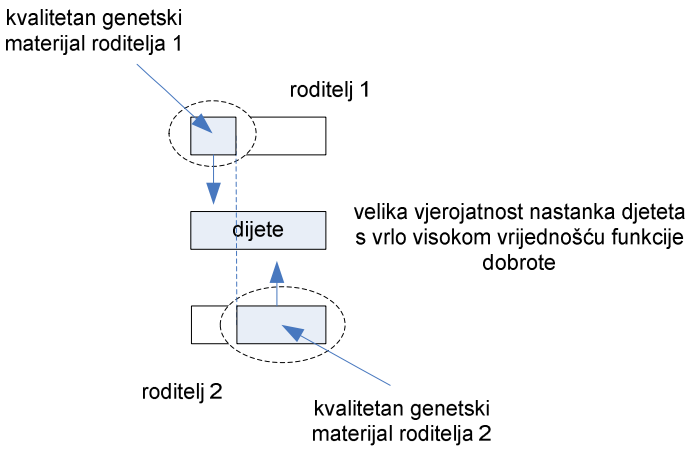
Nakon invertiranja kromosom k ima oblik k = 0011. Promijenjen je poredak gena između točaka invertiranja. Kako bi se pojasnio postupak neka se uzme kromosom k = 1001010111 s točkama invertiranja iza trećeg i sedmog gena.

$$k = 100|1110|111$$

Nakon invertiranja k = 100|0111|111 .

Ovaj jednostavni primjer prikazuje princip rada genetskog algoritma i logiku kojom se genetski algoritam koristi kako bi došao do optimuma. Kodiranje jedinki, operatori križanja i mutacije su posebno prilagođeni za problemsku domenu koja se obrađuje. U poglavlju 4 je opisano više genetskih operatora koji se koriste za rješavanje problema trgovačkog putnika, koji su znatno složeniji i konceptualno i za praktičnu izvedbu od operatora ovdje opisanih i korištenih za prezentaciju rada genetskog algoritma.

Križanjem se nastoji što je više moguće sačuvati genetski materijal roditelja. Mutacijom se želi unjeti „kaos“ u jedinku u nadi da će mutirana jedinka imati bolji genetski materijal nego nemutirana. Križanjem dvije jedinke visoke dobrote, ali s lošim pojedinim dijelovima kromosoma, moguće je dobiti bolju jedinku od roditelja. Križanje mora biti takvo da novonastala jedinka ima kvalitetne gene iz oba roditelja. Zato je bitno da se genetski materijal roditelja sačuva što je više moguće. Na slici 2.2 se nalazi primjer za navedeni slučaj.



SLIKA 2.2 – Važnost čuvanja genetskog materijala roditelja

2.2.5. Parametri genetskog algoritma

Parametri služe za upravljanje radom genetskog algoritma. Parametrima se može utjecati na vrijeme izvođenja genetskog algoritma, agresivnost kojom će algoritam pretraživati prostor rješenja (da li će biti više ili manje slučajnih skokova), brzinu kojom će algoritam konvergirati ka optimumu, itd.

Dizajn genetskog algoritama je bitna komponenta u životnom ciklusu (razvoju i korištenju) nekog genetskog algoritma, ali je puno bitnije znati kako pravilno iskoristiti taj algoritam kako bi se dobila rješenja koja su dovoljno dobra u nekom razumnom vremenu. Drugim riječima, genetski algoritam se može shvatiti kao alat s kojim se neki posao može kvalitetno napraviti, a parametri se mogu shvatiti kao kontrola kako se posao izvodi. Ako su parametri dobro podešeni tada će se posao kvalitetno izvesti.

Parametri koje je moguće namještati ovise o izvedbi genetskog algoritma. Postoje i neki standardni parametri koji se namještaju kod svakog genetskog algoritma i o kojima najviše ovisi kako će se genetski algoritam ponašati. Ti parametri su:

1. veličina populacije v_p ili N,
2. broj iteracija bl i
3. vjerojatnost mutacije v_m – najvažniji parametar genetskog algoritma.

U ovom su radu tri prethodno navedena parametra detaljno obrađena i analizirana na primjeru problema trgovačkog putnika.

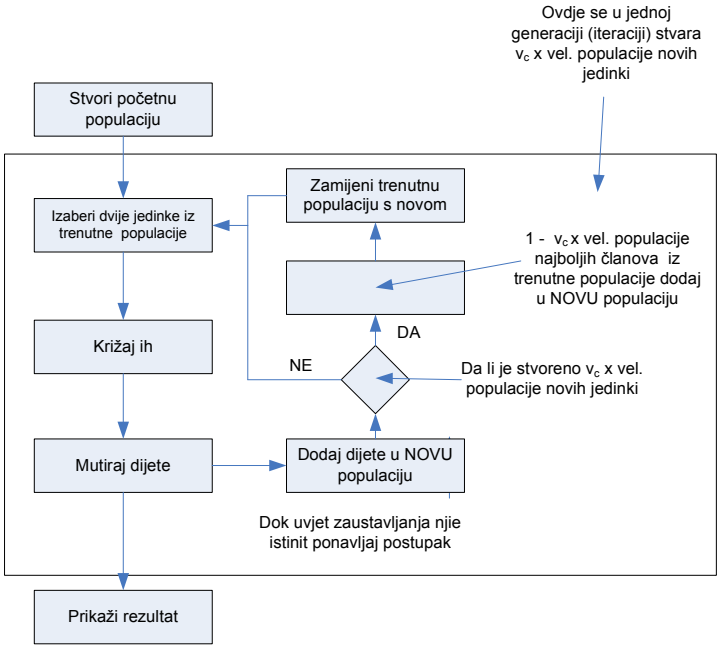
Ostali standaradni parametari koji se mogu podešavati su:

4. duljina kromosoma b,
5. vjerojatnost križanja v_c i
6. selekcijski pritisak.

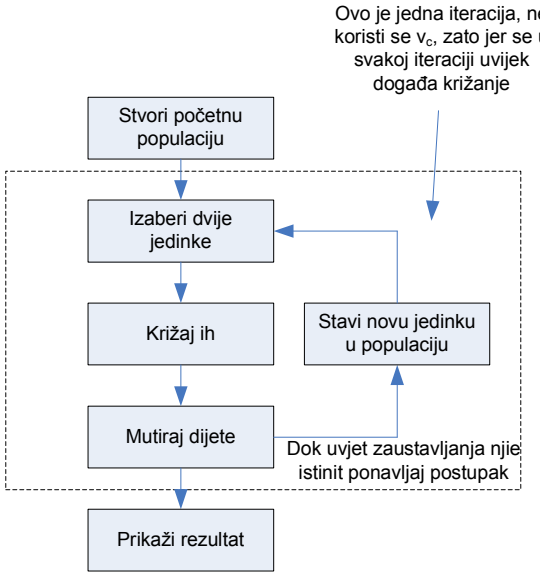
Neki genetski algoritmi mogu koristiti sve navedene parametre, neki mogu koristiti samo određene, ali svi koriste prva tri navedena parametra u svom radu. Osim parametra navedenih ovdje, koji su standardni za velik broj genetskih algoritama, postoje posebni parametri koje koriste genetski algoritmi specijaliziranog dizajna.

Duljina kromosoma (b) je parametar koji je gotovo uvijek zadan unaprijed i ne mijenja se. Taj se parametar izražava kao *dimenzija kromosoma x broj bitova* u binarnom prikazu. U ilustrativnom primjeru rada genetskog algoritma dimenzija kromosoma je 1, a broj bitova je 4. Drugi primjer je duljina kromosoma kod TSPa, koja ovisi o broju gradova koje je potrebno obići. Problem s 130 gradova ima duljinu kromosoma *130 x broj bitova* koji se koristi za prikaz pojedinog gena. Dimezija kromosoma je 130. Izgled kromosoma je definiran u poglavlju 4. S obzirom na to da je ovaj parametar predefiniran, potrebno je ispravno podesiti ostale parametre genetskog algoritma kako bi se u zadovoljavajućem vremenu postizala zadovoljavajuća rješenja [7].

Vjerojatnost križanja (v_c) je parametar koji se može, ali i ne mora pojaviti u genetskom algoritmu. Taj parametar direktno ovisi o dizajnu genetskog algoritma. Na slikama 2.3 i 2.4 se nalaze dvije varijante genetskog algoritma. Na slici 2.3 se nalazi skica genetskog algoritma koji koristi vjerojatnost križanja. Na slici 2.4 se nalazi skica genetskog algoritma koji ne koristi vjerojatnost križanja. Varijanta koja ne koristi vjerojatnost križanja (varijanta a) u svakoj iteraciji obavlja selekciju i reprodukciju. Varijanta koja koristi vjerojatnost križanja (varijanta b) u jednoj generaciji zamijeni *veličina populacije x* v_c jedinki. Očito je da se na drugačiji način novonastale jedinke ubacuju u slijedeću generaciju. U varijanti a nova populacija nastaje tako da se u svakoj iteraciji doda po jedan novi član. U varijanti b se u jednoj generaciji (iteraciji) zamijeni *veličina populacije x* v_c jedinki. Iako su možda naizgled slični, ova dva genetska algoritma su potpuno različita zbog načina na koji rade. Razlika je u definiciji pojma broj iteracija. U varijanti b broj iteracija (generacija) može biti puno manji nego u varijanti a. U jednoj iteraciji se mijenja *veličina populacije x* v_c jedinki, a ne samo jedna jedinka kao u varijanti a [7].



SLIKA 2.3 – Genetski algoritam s vjerojatnošću križanja



SLIKA 2.4 – Genetski algoritam bez vjerojatnosti križanja

Selekcijski pritisak je parametar pomoću kojega se može kontrolirati brzina konvergencije genetskog algoritma. Veći selekcijski pritisak znači bržu konvergenciju genetskog algoritma. Posljedica brže konvergencije može biti zaglavljivanje algoritma u nekom lokalnom optimumu, što nije poželjno. Za različite vrste selekcije se na različiti način definira selekcijski pritisak. Na primjeru turnirske selekcije, selekcijski pritisak se definira pomoću parametra *k* (veličine turnira). Što je *k* veći, veći je i selekcijski pritisak. Kako je pokazano da turnirska selekcija s *k = 2* ima veći selekcijski pritisak od proporcionalnih i rangirajućih selekcija, parametar *k* mora biti što je moguće manji. Na primjer, ako se u jednoj iteraciji obavlja i

selekcija i reprodukcija tada je potrebno da je najmanja vrijednost parametra *k = 3*. Razlog tome je slijedeći: slučajno se izaberu tri jedinke od kojih se bolje dvije križaju, a dijete ulazi u novu populaciju na mjesto one jedinke koja nije bila u procesu križanja [7].

Veličina populacije (N,v_p) je parametar koji direktno utječe na kvalitetu dobivenih rješenja. Ovisno o vrsti selekcije može utjecati i na vrijeme izvođenja algoritma. Korištenjem turnirske selekcije veličina populacije ne utječe direktno na vrijeme izvođenja genetskog algoritma. Manja populacija daje bolje rješenje za manji broj iteracija, povećanjem populacije potrebno je povećati broj iteracija kako bi se zadržala kvaliteta rješenja [7].

Vjerojatnost mutacije (v_m) je najvažniji parametar genetskog algoritma. Vjerojatnost mutacije radi slučajne skokove algoritma po prostoru rješenja, što omogućuje izbjegavanje zaglavljivanja algoritma u lokalnim optimumima i proširivanje područja pretrage na još neistražene dijelove prostora rješenja. Novonastala rješenja mogu biti lošija od postojećih, ali ta rješenja imaju manju vjerojatnost ulaska u daljne reprodukcije čime se ostavlja prostor za napredovanje u pravom smjeru. Zbog uloge koju ovaj parametar ima potrebno je jako pažljivo odabrati njegove vrijednosti. Genetski algoritam je jako osjetljiv i na najmanje promjene ovoga parametra.

Broj iteracija (bI) je parametar koji direktno utječe i na vrijeme izvođenja algoritma i na kvalitetu dobivenih rješenja. Što je veći broj iteracija rješenje je bolje. Ali, veći broj iteracija znači i više vremena koje će genetski algoritam potrošiti na svoje izvođenje. Kako je cilj postići što bolje rješenje u što kraćem vremenu, potrebno je odrediti broj iteracija u kojemu će algoritam pronaći zadovoljavajuće rješenje. U praksi nije uvijek potrebno pronaći optimum. Rješenje koje je dovoljno blizu optimuma, a za koje je potrebno puno manje vremena kako bi se do njega došlo, je zadovoljavajuće rješenje.

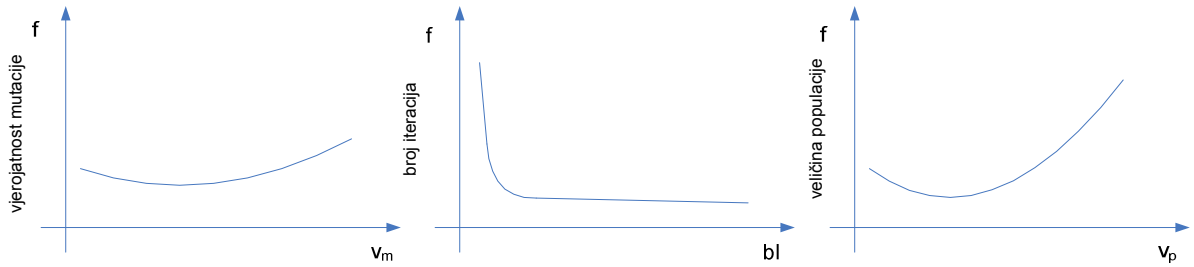
Parametri genetskog algoritma predstavljaju važnu komponentu rada algoritma. Mnogi su znanstvenici pokušavali odrediti skupove parametara koji bi bili optimalni za velik broj problema. No, nažalost se pokazalo da svaki problem ima svoj skup optimalnih parametara te da nije moguće odrediti parametre koji bi bili dobri za širok spektar problema. Ali se uspjelo doći do zaključaka o tome koji su parametri dobri za početak pretraživanja genetskog algoritma i od kuda treba krenuti kako bi se odredili optimalni parametri za pojedini genetski algoritam [7].

Postojali su i pokušaji da se odrede relacije pomoću kojih bi se mogla izračunati koja je poželjna vrijednost trećeg parametra ako su poznata prva dva [7].

Nemoguće je odrediti skupove parametara koji bi bili optimalni za rješavanje velikog skupa problema, ali postoje određene međuzavisnosti i pravilnosti između pojedinih parametara [7]:

1. veći broj iteracija u pravilu znači bolje rješenje
2. za veće populacije treba povećati broj iteracija i smanjiti vjerojatnost mutacije
3. jednako kvalitetna rješenja mogu se postići za različite skupove parametara

Na temelju eksperimenata su nacrtane krivulje na slici 2.5. Krivulje na slici 2.5 opisuju gore navedene međuzavisnosti koje je cilj potvrditi u ovome radu na problemu trgovačkog putnika. Krivulje se odnose na problem minimizacije. Ekvivalentne postoje i za problem maksimizacije višedimenzijske funkcije [7]. Ove krivulje pokazuju ovisnosti vjerojatnosti mutacije, broja iteracija i veličine populacije o vrijednosti funkcije dobrote.



SLIKA 2.5 – Skice međuzavisnosti parametara genetskog algoritma

Na slici 2.5 lijevo se nalazi krivulja koja opisuje ponašanje vjerojatnosti mutacije. U sredini je krivulja koja opisuje ponašanje broja iteracija. Desno se nalazi krivulja koja opisuje ponašanje veličine populacije u odnosu na promjenu ostalih parametara. Detaljna analiza krivulja se nalazi u poglavlju 5.4.

Genetski algoritam se može dizajnirati i tako da se tokom rada algoritma parametri mogu sami podešavati, bez uplitanja korisnika, kako bi se promijenilo ponašanje algoritma. Na primjer, ako se najbolja jedinka nije promijenila kroz 100 iteracija, tada algoritam mijenja veličinu populacije i vjerojatnost mutacije kako bi izbjegao mogućnost zaglavljivanja u lokalnom optimumu. U genetskom algoritmu koji koristi turnirsku selekciju moguće je u početku postaviti *k* na veću vrijednost (npr. k = 5) kako bi algoritam brže konvergirao. Nakon određenog broja iteracija algoritam sam smanjuje selekcijski pritisak na manju vrijednost (k = 3) kako izbjegao zaglavljivanje u lokalnim optimumima.

3. Problem trgovačkog putnika

3.1. Definicija problema

Problem trgovačkog putnika (*engl. Traveling Salesman Problem, TSP*) je naziv za širok spektar problema. Problemi imaju i teorijsko i praktično značenje.

Problem je po prvi puta postavio Euler 1759. godine. Eulerov problem bio je pomicati kralja na šahovskoj ploči tako da kralj svako polje obiđe samo jednom i da obiđe sva polja.

Praktičniji opis problema se pojavljuje 1832. godine u knjizi njemačkog trgovca BF Voigta. Knjiga opisuje kako biti uspješan trgovački putnik. Uspješan trgovački putnik mora obići što je moguće više lokacija, a da se pri tome niti jedna ne obiđe dva puta. To je jedan od najbitnijih aspekata prilikom rađenja rasporeda obilaska kupaca [2].

Datum službene definicije problema u matematici nije poznat. Poznato je da se to dogodilo negdje oko 1931. godine.

Definicija koja jako dobro opisuje dani problem u njegovoj osnovnoj i najjednostavnijoj varijanti je slijedeća definicija preuzeta iz [6]:

Za dan skup gradova i cijenu putovanja između svakog para gradova, problem trgovačkog putnika mora pronaći najjeftiniji put koji će obići sve gradove točno jednom i na kraju se vratiti u početni grad.

U ovom diplomskom radu razmatra se osnovna definicija problema. Dodatni uvjet je da je cijena putovanja od grada A do grada B jednaka kao i cijena putovanja od grada B do grada A. Iz definicije problema se vidi kako je ovo problem u kojem se traži minimum.

Matematička definicija problema je sljedeća[2]:

Na zadanom težinskom grafu $G = (V,E)$, gdje je c_{ij} težina (cijena) brida koji spaja vrhove i i j i ima nenegativnu vrijednost, potrebno je pronaći stazu koja obilazi sve čvorove i ima najmanju cijenu.

Iako je definicija jednostavna i jasna problem je težak, čak štoviše spada u klasu NP - teških problema. Točnije, problem je faktorijalne složenosti. Za 10 gradova broj mogućih rješenja iznosi 10! = 3,628,800. Za veći broj gradova, što su u današnje vrijeme realne ture (100 – tinjak gradova) , broj mogućih tura se penje na približno 9.3e157. Vrijeme potrebno da se metodom grube sile (brute force) riješi problem sa sto gradova iznosi 3e144 godina. Pretpostavka je da je broj ruta koje današnje računalo može obraditi po sekundi jednak 1,000,000 [5].

3.2. Primjeri problema trgovačkog putnika

Praktični primjer ove najjednostavnije definicije je doslovno trgovački putnik kojemu je bitno obići sve svoje kupce uz najmanji prijeđeni broj kilometara. Ovdje je cijena broj prijeđenih kilometara. Cijena može biti i najmanje vrijeme koje je potrebno kako bi se obišli svi kupci.

Druga razina problema trgovačkog putnika u cijenu uključuje više varijabli. Npr. gledaju se i minimalni put i minimalno vrijeme koje je potrebno kako bi se obišli svi gradovi (točke koje je potrebno obići). Kako se u praksi dosta često događa da se ne poklapaju minimalni put i minimalno vrijeme, uvode se i težine kojima se opisuje je li bitniji put ili je bitnije vrijeme. Težine se dodaju u funkciju dobrote i određuju ponašanje genetskog algoritma koji rješava problem.

Broj uvjeta u cijenama nije ograničen. Bitno je napomenuti kako nije preporučljivo staviti previše uvjeta pred algoritam. Veći broj uvjeta produljuje vrijeme izvođenja algoritma i algoritam koristi veću količinu resursa.

Treća razina problema uz više faktora cijene obuhvaća i određena ograničenja. Npr. ograničenja mogu biti slijedeća: neki resurs je dostupan samo u određenom vremenskom periodu, resurs je uvijek dostupan, ali za njega postoji red čekanja koji varira u vremenu itd.

Ova ograničenja se najbolje mogu opisati na dva sljedeća problema. Prvi od njih spada u takozvani vremenski ovisan problem trgovačkog putnika (*engl. Time Dependent Travelnig Salesman Problem, TDTSP*). Uz minimalni prijeđeni put u obzir se uzima i vrijeme koje je potrebno da se put obavi kao i vremenski periodi u kojima se jedan dio posla mora napraviti.

Potrebno je napraviti optimalan obilazak korisnika kroz zabavni park. Korisnik ujutro dolazi na ulaz zabavnog parka i bira koje će sve atrakcije u parku obići taj dan. Do atrakcije je potrebno doći, pričekati u redu, provesti neko vrijeme te nakon toga ići dalje. Također, neke atrakcije rade samo određeno vrijeme. Ovdje se vidi kako su i vrijeme i put uračunati u cijenu. Čekanje u redu i vrijeme potrebno za dolazak do neke atrakcije predstavljaju ograničenja [9].

Drugi primjer spada u grupu problema trgovačkog putnika koja se zove problem raspoređivanja vozila (engl. *Vehicle Routing Problem, VRP*). Ovaj problem se može najbolje opisati na primjeru određivanja optimalne rute za neko dostavno vozilo u gradu. Vozilo mora do određenog vremena proizvod dostaviti do prodavaonica, uz najmanji prijeđeni broj kilometara. U vožnju se moraju uračunati gužve u prometu, eventualni radovi i razni drugi faktori koji se mogu pojaviti u vožnji. Cijena je najmanji pređeni broj kilometara. Ograničenja su vremenski rok za dostavu, gužve u prometu, radovi, itd.

Još jedan primjer problema trgovačkog putnika je i problem više trgovačkih putnika (*engl. multisalesman TSP*). Problem je jednak osnovnom problemu. Umjesto jednog putnika ima ih više. Osim ovdje opisanih problema postoji još velik broj primjena TSP problema u konstrukciji aviona, robotici, rudarstvu, računarstvu, itd.

3.3. Metode za rješavanje problema trgovačkog putnika

Problem trgovačkog putnika spada u grupu NP-teških problema, što znači da se ne može riješiti u polinomijalnom vremenu. Polinomijalno vrijeme je vrijeme potrebno za izračunavanje nekog problema. Uvjet je da vrijeme izračunavanja ne smije biti veće od polinoma koji opisuje problem. Stroga matematička definicija je sljedeća:

$$m(n) = O(n^k)$$

gdje je $m(n)$ vrijeme izračunavanja, $O(n^k)$ složenost problema. K je konstantna vrijednost i ovisi o problemu. Vrijeme se izražava kao broj koraka koji je potreban da bi se došlo do rješenja. Problem trgovačkog putnika je faktorijalne složenosti. Vrijeme potrebno za njegovo izračunavanje se može napisati sljedećom definicijom:

$$m(n) = O(n!)$$

Za velike $n, n!$ raste puno brže nego , n^k, k je konstatnta.

Jedan način rješavanja problema, koji sigurno daje optimalno rješenje je pronaći sve moguće staze i izračunati njihove duljine. Staza koja ima najmanju duljinu se odabire kao najbolja. Drugim riječima to je optimalna staza. Npr. za problem od 25 gradova bilo bi potrebno deset milijuna godina da se sve staze odrede i izračunaju (broj mogućih staza iznosi $3{,}2 \times 10^{23}$). Pretpostavka je da se u jednoj nanosekundi može odrediti i izračunati duljinu jedne staze.

Iz tog su se razloga u praksi pojavile druge metode za rješavanje problema trgovačkog putnika. Korištenjem tih metoda dolazi se do optimalnog ili rješenja dovoljno blizu optimumu u znatno kraćem vremenu. Za praktičnu primjenu najčešće nije potrebno pronaći optimalno rješenje. Rješenje blizu optimuma je dovoljno dobro rješenje, ako se za njegovo izračunavanje znatno skraćuje vrijeme izvođenja algoritma.

Pohlepni algoritmi (*engl. Greedy Algorithms*) su jedna metoda koja pronalazi zadovoljavajuće rješenje za problem trgovačkog putnika. Algoritam stvara listu svih bridova u grafu i slaže ih od najmanje cijene prema najvećoj. Tada odabire bridove tako da prvo uzima one s manjom cijenom pazeći da ne zatvori ciklus. Bitno je napomenuti kako ova metoda ne daje uvijek zadovoljavajuća rješenja [2].

Metoda najbližeg susjeda je još jedna jednostavna metoda koja može dati dobra rješenja u problemu trgovačkog putnika. Ona radi tako da se odabere jedan grad i ostali gradovi se obilaze tako da se uvijek posjećuje onaj grad koji ima najmanju udaljenost do onog grada u kojem se trenutno nalazimo. Potrebno je paziti da se na zatvori ciklus dok se ne obiđu svi gradovi [2].

Osim ovih jednostavnih metoda postoje i one kompliciranije koje daju bolja rješenja, ali i zahtjevaju više vremena i resura kako bi došle do rješenja. To je na primjer metoda najmanjeg razapinjućeg stabla ili genetski algoritmi.

4. Rješavanje problema trgovačkog putnika uz pomoć genetskih algoritama

Pretraga cijelokupnog prostora rješenja je dugotrajan postupak. Metode poput pohlepnog algoritma ili najbližeg susjeda ne daju uvijek dobre rezultate. Zato se genetski algoritmi, kao metoda slučajnog i usmjerenog pretraživanja prostora rješenja, nameću za metodu koja bi u relativno kratkom vremenu mogla pronaći zadovoljavajuće rješenje.

Za problem trgovačkog putnika postoji više različitih načina kodiranja jedinki, više različitih vrsta selekcije, križanja i mutacije od kojih su one najčešće ovdje opisane. Dodatni plus genetskim algoritmima je i njihova mogućnost kombiniranja s metodama traženja lokalnog optimuma. Kombinacijom genetskog algoritma i metode traženja lokalnog optimuma se postiže brža konvergencija algoritma. Takva vrsta genetskih algoritama se naziva hibridnim genetskim algoritmima. Ovdje je opisana jedna metoda traženja lokalnog optimuma, *2opt* metoda, koja se može koristiti umjesto operatora mutacije.

4.1. Prikaz kromosoma

Iako postoji više mogućih načina prikaza, najintuitivniji prikaz, koji se najčešće i koristi može se prikazati na slijedeći način:

(0 , 1 , 3 , 5 , 4 , 6 , 2 , 7)

U ovom prikazu postoji 8 gradova koje treba obići. Redosljed obilaska je slijedeći: kreće se iz grada 0, pa u grad 1, nakon njega 3 itd. Dakle, mjesto u osmorki označava kada će taj grad biti posjećen. A broj koji je na tom mjestu označava sam grad. Ukratko, gradovi su poredani u smjeru u kojem se posjećuju. Ovaj način prikaza je uobičajen u TSP implementacijama [2,5].

Drugi mogući način prikaza je također niz znakova, ali sa sljedećim značenjem:

Ako je $v = a_1a_2\ldots a_n$, jedna moguća tura, tada se ona obilazi tako da se iz grada *i* ide u grad a_i. Na primjer, ako je v = 3421, tada se iz grada 1 ide u grad 3, iz grada 3 se ide u grad 2, iz grada 2 se ide u grad 4, a iz grada 4 se ide u grad 1. To je takozvani ciklički prikaz. Problem sa ovakvim načinom prikaza je mogućnost pojave ilegalne ture što se vidi na primjeru v = 3412 [2].

Još jedan način prikaza koji se koristi je matrični prikaz. Matrica je tipa *n * n*. To je zapravo matrica susjedstva u grafu koji se dobiva povezivanjem svih gradova. Na mjestu *M[A,B]* se nalazi jedinica ako postoji put iz grada *A* u grad *B*, inače je na mjestu *M[A,B]* nula. Zbog kompleksnosti on nije ovdje korišten, iako je po brzini izvođenja bolji prikaz od ovdje

korištenog. Danas se ide u smjeru razvoja operatora za ovakav način prikaza kromosoma [2].

4.2. Funkcija cilja

Funkcija cilja je jednostavna formula koja zbraja udaljenosti između gradova, dajući tako za svaki kromosom ukupnu dužinu puta. Što je dužina puta kraća, kromosom je bolji i ima veću šansu za opstanak.

4.3. Operatori selekcije

4.3.1. Prirodna selekcija

Iz početne se populacije eliminira $R = M \times p_e / 100$ jedinki (M je veličina populacije, a p_e je vjerojatnost križanja). Jedinke se eliminiraju tako da se sačuva različitost populacije, odnosno eliminiraju se slične jednike. Na početku se cijela populacija sortira prema dobroti. Nakon toga se uspoređuje sličnost dobrota susjednih jedniki. Ukoliko je razlika dobrota susjednih jedinki manja od predefiniranog malog realnog pozitivnog broja ε, eliminra se jedna od n – torki. To se ponavlja dok je broj eliminiranih jedniki manji od R. Ako je nakon ovog postupka broj eliminiranih jedinki i dalje manji od R eliminiraju se jedinke s lošijom vrijednošću funkcije dobrote [4].

4.3.2. Turnirska selekcija

Ideja turnirske selekcije je jednostavna. Iz cijelokupne populacije odabere se k jedinki (k je veličina turnira, obično između 3 i 7) te se najlošija od njih izbaci. Od ostatka jedniki slučajnim se odabirom odabiru dvije koje se onda križaju i daju novu jedniku koja ulazi na mjesto izbačene jedinke u populaciju [1,3].

4.4. Operatori križanja

4.4.1. Partially matched crossover (PMX)

PMX križanje radi na sljedeći način:

U oba roditelja označe se 2 točke prekida (na slici 4.1 označene s |). Geni između tih točaka se zamijene u oba roditelja, što daje slijedeće: 3 zamjenjuje 2, 8 zamjenjuje 4, 4 zamjenjuje 1 i obratno.

Sada se popunjava ostatak kromosoma tako da se gradovi izvan točaka prekida vraćaju na svoje mjesto ukoliko već ne postoje kao rezultat zamjene. Na mjesto grada koji već postoji upisuje se onaj grad kojeg mijenja novopridošli grad. Npr. kada se nakon zamjene želi u prvo dijete staviti grad 1 on već postoji (novija jedinica pridošla iz drugog kromosoma). Sada na

mjesto jedinice dolazi 8 jer 4 zamjenjuje 1, a kako i 4 već postoji u kromosomu, 8 zamjenjuje 4. Prilikom izgradnje ovog operatora posebnu pažnju treba posvetiti dijelu koji će raditi postupak traženja gena opisan u prethodnoj rečenici kako ne bi došlo do izgradnje ilegalnih tura.

(2 5 1 | 3 8 4 | 7 6)

(8 6 7 | 2 4 1 | 3 5)

postaje

(3 5 8 | 2 4 1 | 7 6)

(1 6 7 | 3 8 4 | 2 5)

SLIKA 4.1 – PMX križanje

PMX križanje odvlači populaciju u lokalni optimum brže od drugih operacija križanja ovdje opisanih [2,5].

4.4.2. Greedy crossover

Definicja Greedy Crossover križanja je slijedeća:

„Greedy Crossover uzima prvi grad iz jednog roditelja, uspoređuje gradove u koje se dolazi iz tog grada u oba roditelja te uzima onog čiji je put kraći. Ako se je jedan grad već pojavio u djetetu tada se uzima drugi. Ako su oba u djetetu tada se slučajnim odabirom odabire jedan neodabrani grad.“ [1]

4.4.3. Greedy subtour crossover (GSX)

Ovo križanje radi tako da iz oba roditelja uzima što je moguće dulji podskup gradova iz oba roditelja na način koji je prikazan na slici 3.2. Na taj je način najbolje sačuvan genetski materijal roditelja. To zapravo znači sljedeće, ako postoje dva kromosma koja oba sadržavaju podskupove optimalne ture, ovim križanjem se može vrlo brzo doći do spajanja tih dijelova što naravno dovodi do brže konvergencije samog problema.

```
ulaz: kromosom ka = (a0,a1,..., an-1) i kb = (b0,b1,..., bn-1)
            izlaz: dijete k
            procedura krizaj (ka,kb){
                  fa <- true
                  fb <- true
                  izaberi jedan slucajan grad g
                  izaberi x gdje je ax = t
                  izaberi y gdje je by = t
                  k <- t
                  čini{
                        x = (x - 1) mod n
                        y = (y - 1) mod n
                        ako fa = true onda{
                              ako ax nije u k onda
                                    k <- ax * k
                              inače
                                    fa <- false
                        }
                        ako fb = true onda{
                              ako by nije u k onda
                                    k <- k * by
                              inače
                                    fb <- false
                        }
                  } dok fa = true ili fa = true
                  ako staza nije potpuna onda
                        dodaj ostale gradove slucajnim redosljedom u k
                  vrati k
            }
```

SLIKA 4.2 – Pseudokod GSX algoritma

Na slici 4.2 *n* je broj gradova, a * je znak konkatenacije nizova. GSX križanje je najefikasnija vrsta križanja u odnosu na vrste križanja ovdje opisane [4].

4.4.4. Poredano križanje (OX)

Poredano križanje je slično PMX križanju zbog toga što se odabiru dvije točke prekida i zamjenjuju se geni u kromosomima na tim mjestima. Razlika je u načinu popunjavanja ostatka kromosoma. Kod poredanog križanja se samo ciklički promijeni poredak ostalih gena u kromosomu tako da se dobije legalna tura. Ako su kromosomi v_1 i v_2 roditelji tada postupak dobivanja djece izgleda (točke križanja su označene s |):

$$\begin{aligned} v_1 &= 213|854|76 \\ v_2 &= 326|571|84 \end{aligned} \qquad (1)$$

točke križanja se nalaze iza trećeg i šestog gena. Za početak se u djeci zamjene geni koji se nalaze unutar točaka križanja

$$\begin{aligned} v_1 &= \_\ \_\ \_\ |571|\ \_\ \_ \\ v_2 &= \_\ \_\ \_\ |854|\ \_\ \_ \end{aligned} \qquad (2)$$

Nakon toga se geni roditelja zapišu u redosljedu počevši od gena iz druge točke križanja

$$\begin{aligned} v_1 &\ 76213854 \\ v_2 &\ 84326571 \end{aligned} \qquad (3)$$

iz ovako zapisanih gena se izbace novopridošli geni iz drugog roditelja. Dakle, iz v_1 se izbace geni 5,7 i 1, a iz v_2 se izbace geni 8,5 i 4. Sada se dobiva slijedeće:

$$\begin{aligned} v_1 &\ 62384 \\ v_2 &\ 32671 \end{aligned} \qquad (4)$$

Gene dobivene u koraku 4 je potrebno umetnuti u pripadajuću djecu počevši od druge točke križanja.

$$\begin{aligned} v_1\text{'} &= 384\ |571|\ 62 \\ v_2\text{'} &= 671\ |854|\ 32 \end{aligned} \qquad (5)$$

Na ovaj su način uvijek dobivene legalne ture [2].

4.4.5. Matrično križanje (MX)

Matrično križanje se koristi kod prikaza kromosoma matrično. Matrica je tipa *n x n*, elementi matrice su ili 1 ili 0, jedan ako se između grada *i* i grada *j* nalazi brid, a nula inače. Po načinu na koji se radi križanje slično je križanju s jednom ili dvije točke prekida.

Matrice *A* i *B* su roditelji, križanjem s dvije točke prekida nakon prvog i drugog stupca dobivaju se djeca prikazana na slici 4.3.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \qquad A' = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad B' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

a) b)

SLIKA 4.3 – Matrično križanje

Operacijom križanja na slici 4.3 došlo je do zamjene drugog stupca. Na a) dijelu slike se nalaze roditelji, a na b) dijelu slike se nalaze dijeca nakon zamjene drugog sutpca. Prilikom križanja došlo je do pojave redova u kojima ima više od jedne jedinice i redova u kojima

nema niti jedne jedince. Taj se problem rješava tako da se jedna jedinica iz reda u kojemu ih ima više prebaci u red u kojemu nema niti jedne jedinice. Koja će se jedinica prebaciti određuje se slučajnim odabirom. Kao rezultat promjene dobiva se sljedeće:

$$A'' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B'' = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

SLIKA 4.4 – Ispravljena djeca nakon matričnog križanja

Djeca na slici 4.4 u svakom redu imaju jednu jedinicu. Nakon ove promjene potrebno je još izgraditi samo jedan ciklus u svakoj matrici. Na slici 4.4 se vidi da u *A''* postoje dva ciklusa. Iz *a* se ide u *a*, a iz *b* se ide u *c* pa natrag u *b*. Stvaranje jednog ciklusa se radi tako da se pokuša što je više moguće zadržati genetski materijal roditelja [2].

Ovo su samo neka od križanja koja se koriste za rješavanje problema trgovačkog putnika. Osim križanja ovdje opisanih postoje i druge vrste križanja koja se u praksi manje koriste. Operacije selekcije, križanja i mutacije koje su ovdje opisane mogu se koristiti u rješavanju svih problema, ne samo problema trgovačkog putnika, koji za kodiranje kromosoma koriste jednaki način kao i problem trgovačkog putnika (npr. problem n kraljica).

4.5. Operatori mutacije

4.5.1. Zamjena gradova

Kao što joj i ime kaže ova mutacija radi tako da uzme dva slučajno odabrana grada u kromosomu i zamjeni njihova mjesta. Na slici 4.5 je prikazana zamjena 5 i 1 [1].

(3 5 4 2 6 1 7 8)

postaje

(3 1 4 2 6 5 7 8)

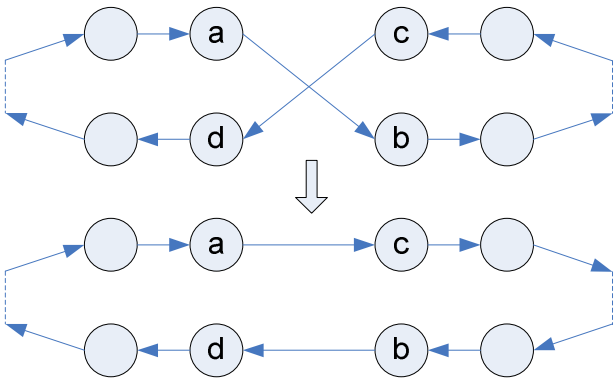
SLIKA 4.5 – Zamjena gradova

4.5.2. Greedy swap mutacija

Ova mutacija je u potpunosti ista kao i jednostavna no uz jedan dodatni uvjet. Do zamjene kromosoma dolazi samo ako je tura dobivena mutacijom kraća od one prije mutacije [1].

4.5.3. 2opt metoda

2opt metoda je jedna od najpoznatijih metoda lokalnog pretraživanja u algoritmima koji rješavaju problem trgovačkog putnika. Unapređuje turu brid po brid okrećući poredak gradova u podturi. Algoritam je prikazan na slici 4.6. Detaljniji rad algoritma opisuje slijedeći primjer: neka postoji put od grada *A* do grada *B* i put od grada *C* do grada *D*. Uspoređuje se da li je *AB + CD > AC + BD*. Ako je, dolazi do zamjene kao što je to prikazano na slici 4.6. Postupak se ponavlja dok je god moguće skratiti turu. Korištenje ovog operatora unosi hibridnost u implementaciju s obzirom na to da je ovo klasična metoda pretraživanja. Velik problem *2opt metode* je zapinjanje u lokalnom optimumu iz kojeg se ne može izvući zbog načina usporedbe (obrnuta situacija nego inače). Zato su tu operatori križanja koji unose dovoljnu raznolikost kako bi *2opt metoda* konvergirala do globalnog optimuma. No usprkos ovog ograničenja ovo je daleko najbolji operator mutacije od svih ovdje opisanih [4].



SLIKA 4.6 – 2opt metoda

4.5.4. Zamjena svakog grada

U kromosomu se gleda svaki gen (grad) i svaki gen može mutirati s vjerojatnošću v_m. Postupak je slijedeći:

1. algoritam se nalazi na genu *i* u kromosomu
2. gen *i* može mutirati s vjerojatnošću v_m
3. ako dolazi do mutacije na mjesto gena i dolazi neki drugi slučajno odabrani gen, a gen *i* ide na mjesto tog slučajno odabranog gena
4. postupak se ponavlja za sve gene unutar kromosoma

Parametar v_m se ne odnosi na kromosom, već se odnosi na svaki gen unutar kromosoma. Moguće je da mutacija bude takva da novonastalo dijete bude jednako roditelju.

Značenje parametra v_m se određuje na temelju rada genetskog algoritma. Značenje definirano u metodi zamjene svakog grada se koristi kod algoritma koji u jednoj iteraciji radi i selekciju i reprodukciju. Kod algoritma koji izgleda kao algoritam sa slike 2.3 b), gdje postoji parametar vjerojatnosti križanja, v_m se može interpretirati kao broj novonastale djece koja će mutirati (*veličina populacije x* v_c *x* v_m).

Osim ovih operatora mutacije postoji još i metoda kod koje se uzme jedan slučajan grad i umetne se na slučajno odabrano mjesto ili isti princip, ali s nekom slučajno odabranom podturom.

5. Eksperimentalno podešavanje parametara genetskog algoritma

5.1. Vrsta genetskog algoritma

Za rješavanje problema trgovačkog putnika genetskim algoritmima postoji mnogo operatora križanja i mutacije. Operatori se mogu međusobno kombinirati kako bi se izgradio genetski algoritam. Na samom početku izgradnje genetskog algoritma potrebno je odabrati na koji će se način kodirati jedinke, a tek nakon toga koji će se operatori koristiti. Neki operatori omogućuju bržu konvergenciju algoritma, čime se povećava mogućnost zapinjanja u nekom lokalnom optimumu. Drugi operatori sporije konvergiraju, ali zato omogućuju širu pretragu prostora rješenja i daju algoritmu veću vjerojatnost da će pronaći optimum. Primjer operatora koji imaju strašno brzu konvergenciju su *GSX* križanje i *2opt* metoda mutacije. Operatori koji sporije konvergiraju, ali omogućuju širu pretragu su PMX križanje i metoda zamjene svakog grada kao operator mutacije. Usporedba brzina rada i kvalitete dobivenih rješenja za pojedine operatore su dostupne u [8].

Cilj ovog diplomskog rada je odrediti kako se ponaša genetski algoritam za različite kombinacije vrijednosti parametara veličine populacije, vjerojatnosti mutacije i broja iteracija. Operatori koji su korišteni su upravo oni koji omogućuju algoritmu širu pretragu prostora stanja. Isto tako, kako bi se zaštitio najbolji kromosom u populaciji algoritam ima ugrađen elitizam.

Algoritam koristi kodiranje jedinki u kojemu su gradovi su poredani u smjeru u kojem se posjećuju. Dakle, kromosom oblika

(0,1,5,3,2,4,6,7)

predstavlja sljedeću turu: kreće iz grada 0, ide se u grad 1, pa u grad 5, nakon njega 3 i tako do 7. Kod računanja duljine puta zatvara se ciklus. Iz zadnjeg posjećenog grada trgovački putnik se vraća natrag u početni grad. Ovaj oblik kodiranja se koristi zato jer je intuitivan, iz njega se lako može isčitati svaka staza. Svaka se jedinka može jedinstveno zakodirati. Nedostatak ovog načina prikaza je činjenica da kromosomi

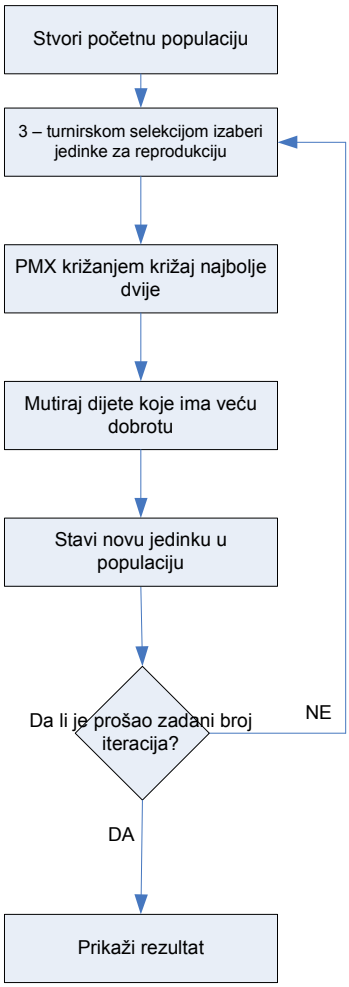
(0,1,5,3,2,4,6,7) i (4,6,7,0,1,5,3,2)

predstavljaju istu jedinku.

Operator selekcije je 3-turnirska selekcija. 3-turnirska selekcija u svakom trenutku čuva dva najbolja člana u populaciji. Time je ostvaren elitizam i time algoritam osigurava da će najbolja pronađena jedinka uvijek biti u populaciji.

Operator križanja je PMX operator s dvije točke prekida. Operator mutacije je metoda zamjene svakog grada zato jer unosi najveću raznolikost od svih opisanih operatora mutacije.

U svakoj se iteraciji obavlja slučajni odabir tri jedinke. Najbolje dvije jedinke se križaju. Bolje dijete iz operacije križanja ulazi u operator mutacije. Mutirano dijete se umeće u populaciju umjesto one jedinke koja nije poslužila za križanje, to jest najlošije jedinke od tri koje su slučajno odabrane u *i-toj* iteraciji. Postupak se ponavlja za unaprijed zadan broj iteracija. Na kraju algoritam staje i daje rješenje. Način rada algoritma je prikazan na slici 5.1.



SLIKA 5.1 – Dijagram toka genetskog algoritma korištenog za eksperimente

Mjerenja su ponavljna 7 puta za svaki pojedini set parametara. Rezultat za jedan set parametara je srednja vrijednost svih 7 dobivenih rješenja. Bitna stvar kod rada ovog

genetskog algoritma je činjenica da se duplikati u populaciji ne brišu već su dozvoljeni i mogu se pojaviti tokom rada algoritma.

5.2. Skupovi podataka nad kojima su se obavljala mjerenja

Problem trgovačkog putnika ima veliku biblioteku poznatih problema koji se mogu koristiti za testiranje rada algoritma. Biblioteka sadrži velik broj problema koji imaju različite složenosti što se tiče veličine, broja točaka koje se trebaju obići, i odnosa među tim točkama, odnosno udaljenosti. Dimenzije problema se kreću od 20ak točaka pa sve do par tisuća točaka. Najveći broj problema je napravljen s nekoliko stotina točaka. Problemi dolaze iz različitih domena. Neki predstavljaju obilazak gradova u Njemačkoj ili Sjedinjenim Američkim Državama, drugi pak obilaske raznih dijelova gradova, a treći dolaze iz rudarske industrije.

Za eksperimente koji su napravljeni u ovom diplomskom korištena su dva problema, prvi s 130 točaka i drugi s 264 točke. Oba problema predstavljaju obilaske gradova. Veći broj mjerenja je napravljen na problemu s 130 gradova. Analiza koja je ovdje provedena temeljena je na rezultatima koji su dobiveni za problem s 130 točaka. Podaci mjerenja za problem s 264 točke se nalaze na priloženom CDu.

5.3. Vrijednosti parametara za koje su se radila mjerenja

U tablici 5.1 se nalaze početni parametri za koje su rađena mjerenja. U prilogu diplomskom radu se nalaze .xml datoteke sa svim mjerenjima. U pojedinoj datoteci se nalaze parametri za koje se mjerenje radi i svih sedam razultata koji su korišteni za računanje srednje vrijednosti.

TABLICA 5.1 – Početni set parametara

broj iteracija (bI)	veličina populacije (v_p)	vjerojatnost mutacije (v_m)
10000	100	0,02
20000	200	0,01
50000	500	0,005
70000	700	0,002
100000	1000	0,001

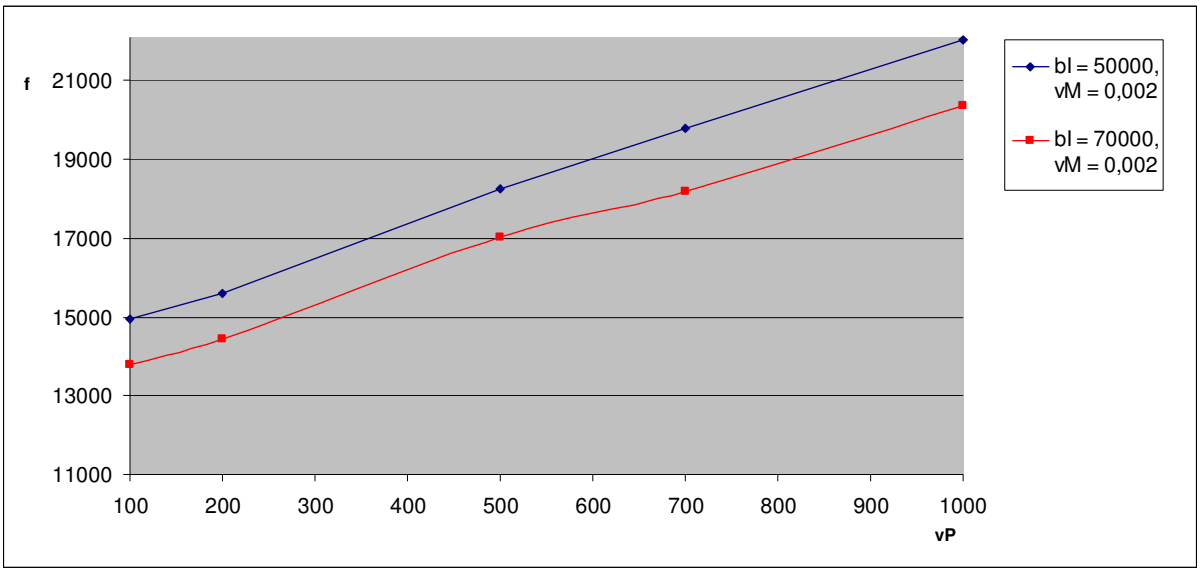
```
<rezultat>
      <bI>10000</bI>
      <vP>100</vP>
      <vM>1p</vM>
      <najboljI0>24651,1349581757</najboljI0>
      <najboljI1>23953,0706226864</najboljI1>
      <najboljI2>21915,8097964561</najboljI2>
      <najboljI3>22842,7608322578</najboljI3>
      <najboljI4>22125,8632660828</najboljI4>
      <najboljI5>23341,8336716638</najboljI5>
      <najboljI6>24686,0961646857</najboljI6>
</rezultat>
```

SLIKA 5.2 – Primjer izlazne datoteke s rezultatima

Na slici 5.2 se nalazi primjer jedne izlazne datoteke za parametre broj iteracija = 10000, veličina populacije = 100 i vjerojatnost mutacije = 1%. Ispod parametara za koje su napravljena mjerenja nalazi se sedam razultata traženja najboljeg puta čija srednja vrijednost predstavlja rješenje za taj set parametara.

Analiza rezultata mjerenja s početnim setom parametara pokazala je kako je potrebno napraviti dodatna mjerenja s drugačijim vrijednostima parametara. Pokazalo se kako rješenja konstantno postaju bolja sa smanjenjem veličine populacije. Bilo je potrebno smanjiti parametar veličine populacije kako bi se provjerilo do kuda ide to ponašanje genetskog algoritma. Manje vrijednosti veličine populacije znače i manji potreban broj iteracija kako bi se postiglo jednako dobro rješenje. Za vjerojatnost mutacije su napravljena dodatna mjerenja unutar granica 1% i 2‰ zato jer se pokazalo da se unutar tih granica postižu najbolja rješenja. Slike 5.3 i 5.4 prikazuju neke rezultate mjerenja dobivene se početnim setom parametara. Na svim se grafovima na *y* osi nalazi vrijednost funkcije *f*, najkraći izračunati put, a na *x* osi se nalazi parametar koji se promatra.

Slika 5.3 prikazuje promjenu funkcije dobrote u ovisnosti o veličini populacije. Na slici se vidi konstantan pad funkcije dobrote s manjim vrijednostima veličine populacije. To je dobro ponašanje zato jer se ispituje minimizacijski problem. Potrebno je vidjeti kako se algoritam ponaša za vrijednosti veličine populacije koje su manje od 100. Cilj je odrediti za koju vrijednost veličine populacije se postiže minimum. Na slici se također vidi odnos između krivulja za 50.000 i 70.000 iteracija. Krivulja za 70.000 iteracija se nalazi niže na grafu. To potvrđuje pretpostavku o radu genetskog algoritma. Veći broj iteracija, u pravilu, znači i bolje rješenje.



SLIKA 5.3 – Promjena funkcije dobrote u ovisnosti o veličini populacije za početni set parametara

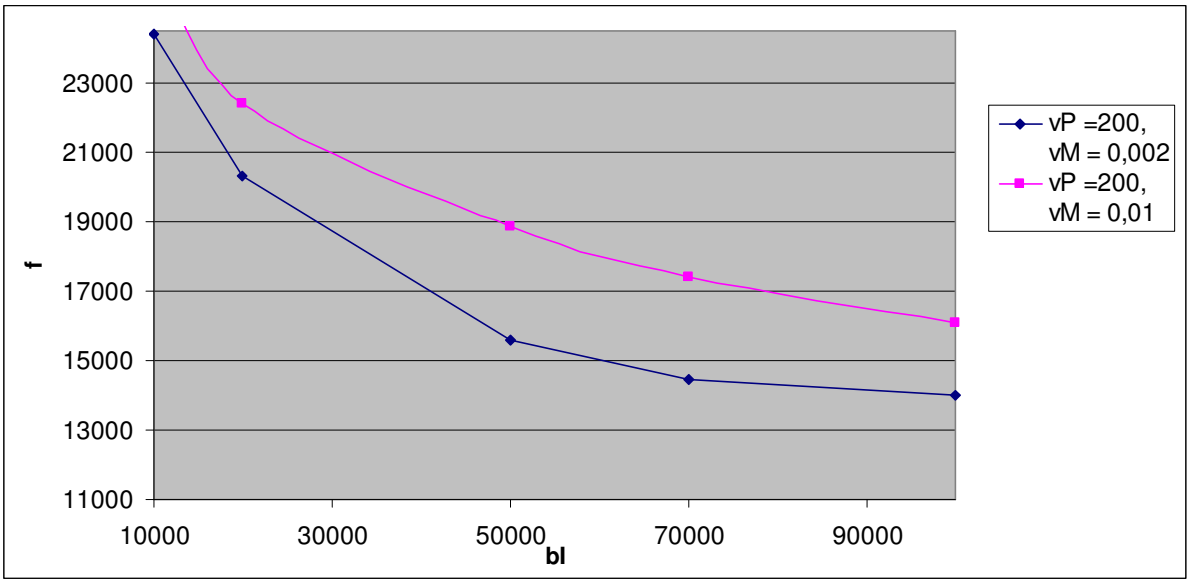
TABLICA 5.2 – Eksperimentalni podaci za sliku 5.3

		bl = 50000, vM = 0,002	bl = 70000, vM = 0,002
vP	100	14952	13778
	200	15614	14444
	500	18248	17029
	700	19772	18201
	1000	22043	20359

Slika 5.4 prikazuje ovisnost funkcije dobrote o promjeni broja iteracija. Bolja rješenja se postižu za veći broj iteracija algoritma. Slika također prikazuje i promjenu rješenja za različite vrijednosti parametra mutacije. Veličina populacije je konstanta i iznosi 200. Vjerojatnost mutacije ima vrijednosti 1% i 2‰. Bolja rješenja se postižu za manju vjerojatnost mutacije. Povećanjem populacije potrebno je smanjiti vjerojatnost mutacije kako bi se dobila kvalitetna rješenja.

TABLICA 5.3 – Eksperimentalni podaci za sliku 5.4

		vP = 200, vM = 0,002	vP = 200, vM = 0,01
vP	10000	24420	26638
	20000	20305	22416
	50000	15614	18841
	70000	14444	17430
	100000	14016	16071



SLIKA 5.4 – Promjena funkcije dobrote u ovisnosti o broju iteracija za početni set parametara

Dodatni set parametara nad kojim su napravljena mjerenja se nalazi u tablici 5.2.

TABLICA 5.4 – Dodatni parametri za koja su izvršena mjerenja

broj iteracija (bI)	veličina populacije (v_p)	vjerojatnost mutacije (v_m)
10000	20	0,01
20000	30	0,007
50000	40	0,005
	60	0,003
	80	0,002

Korištenjem ovih parametra postignuto je znatno bolje ponašanje genetskog algoritma što se tiče rezultata i vremena izvođenja. Zaključak o duljini vremena izvođenja se može odmah izvesti s obzirom na to da su mjerenja rađena za manji broj iteracija. Detaljna analiza rezultata dobivenih ovim mjerenjima se nalazi u poglavlju 5.4.

Kao što se vidi u tablici 5.4, najviše se promijenio parametar veličine populacije. Veličina populacije je smanjena do vrijednosti od 20 jedinki. Broj iteracija je prilagođen manjim vrijednostima veličine populacije (manja veličina populacije, manji broj iteracija za isti rezultat). Vrijednosti za parametar vjerojatnosti mutacije su suženi na one vrijednosti za koje je genetski algoritam davao bolja rješenja u početnom setu parametara.

Također, bitno je reći da su i nakon ovih mjerenja napravljena još neka dodatna mjerenja na temelju dobivenih grafova. Dodatna mjerenja su bila potrebna kako bi se postignuti rezultati jasnije vidjeli na grafovima i kako bi se lakše moglo uočiti kretanje rada genetskog algoritma.

Tako su napravljena dodatna mjerenja za 500.000, 700.000 i milijun iteracija. Rezultati ovih mjerenja jasnije prikazuju konvegenciju genetskog algoritma s porastom broj iteracija. Osim toga su napravljena još neka dodatna mjerenja s veličinom populacije 15 kako bi se jasnije vidjelo kretanje genetskog algoritma kada se taj parametar mijenja. Sva ta mjerenja, sa priloženim grafovima su detaljno analizirana u poglavlju 5.4.

Brojčane vrijednosti parametara koje su korištene u ovom diplomskom radu vrijede samo za ovaj genetski algoritam. Za neki drugi genetski algoritam ovdje korišteni parametri možda neće dati kvalitetne rezultate. No, kretanja parametara koja će biti objašnjena u poglavlju 5.4 su ono što se može iskoristiti za razne vrste genetskih algoritama kao smjernice gdje treba tražiti optimalni set parametara za neki genetski algoritam.

Sveukupno je napravljeno oko 1500 mjerenja. Ponavljanja od 7 puta za isti set parametara su uračunata u 1500 mjerenja. Na temelju dobivenih mjerenja izvedeni su zaključci o ponašanju ovdje opisanog genetskog algoritma za rješavanje problema trgovačkog putnika.

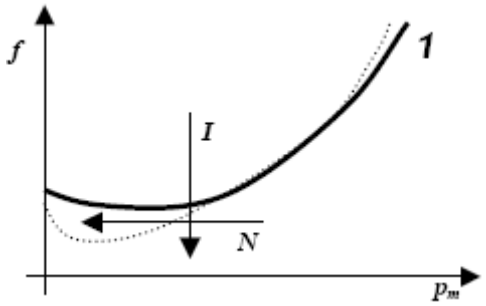
5.4. Analiza rezultata mjerenja

Analiza rada genetskog algoritma napravljena je za tri najvažnija parametra genetskog algoritma: veličinu populacije, broj iteracija i vjerojatnost mutacije. Veličina populacije i broj iteracija direktno utječu na kvalitetu rješenja i vrijeme izvođenja algoritma. Vjerojatnost

mutacije je parametar koji utječe na osjetljivost genetskog algoritma i omogućuje mu širinu pretraživanja prostora stanja. Sve analize su rađene na *ch130* problemu iz standardne biblioteke TSP problema.

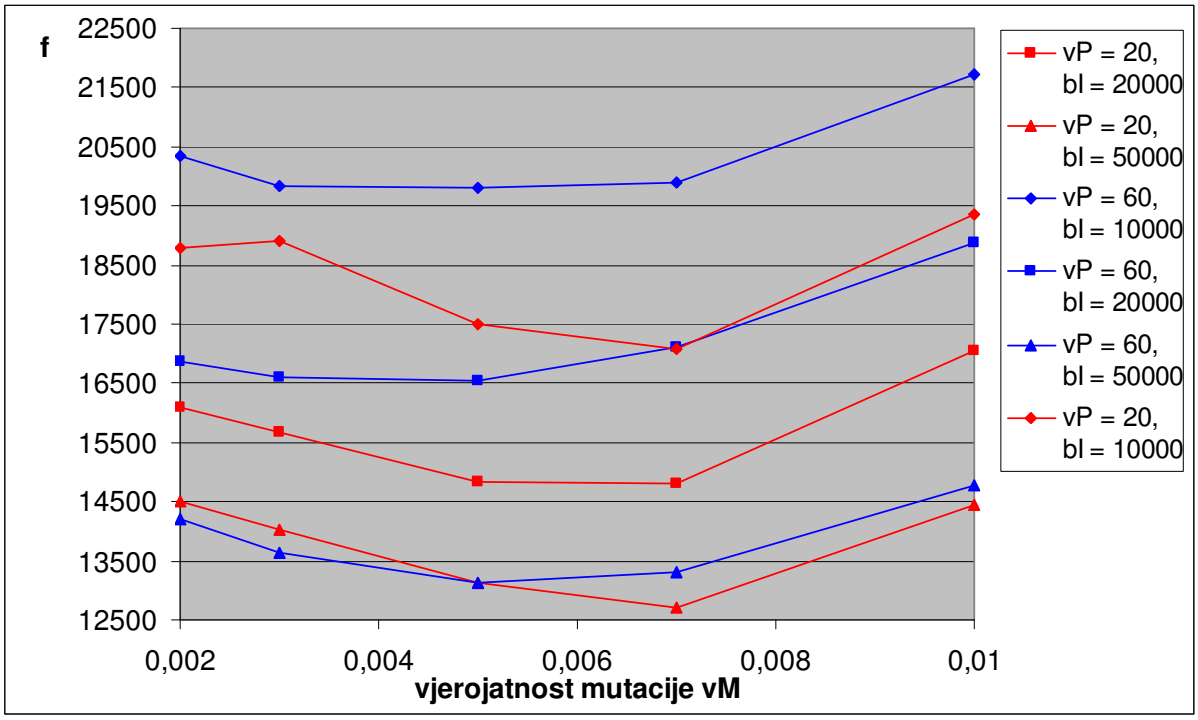
5.4.1. Analiza parametra vjerojatnost mutacije (v_m)

Na slici 5.5 se nalazi predviđeno kretanje krivulje ovisnosti dobrote o promjeni parametra vjerojatnosti mutacije.

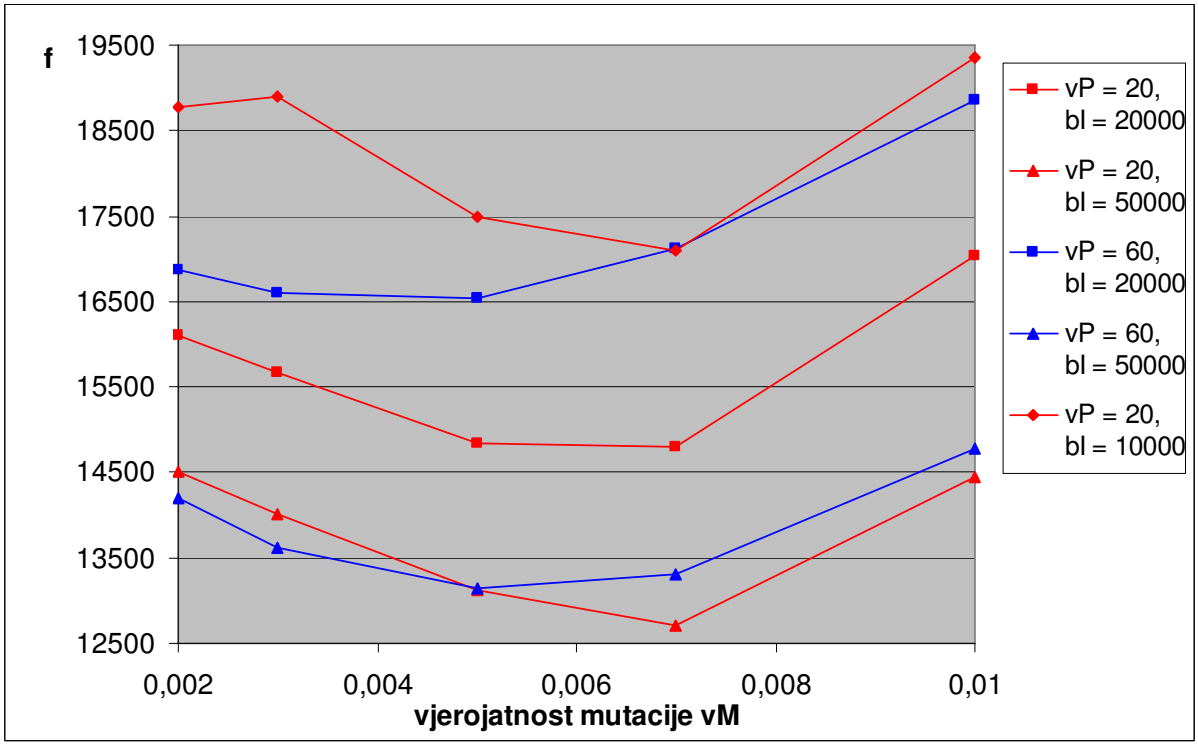


SLIKA 5.5 – Očekivano kretanje krivulje za vjerojatnost mutacije

Mjerenjima su dobiveni rezultati koji su prikazani na slici 5.6.



SLIKA 5.6 a) Eksperimentalna promjena funkcije dobrote u ovisnosti o promjeni vjerojatnosti mutacije



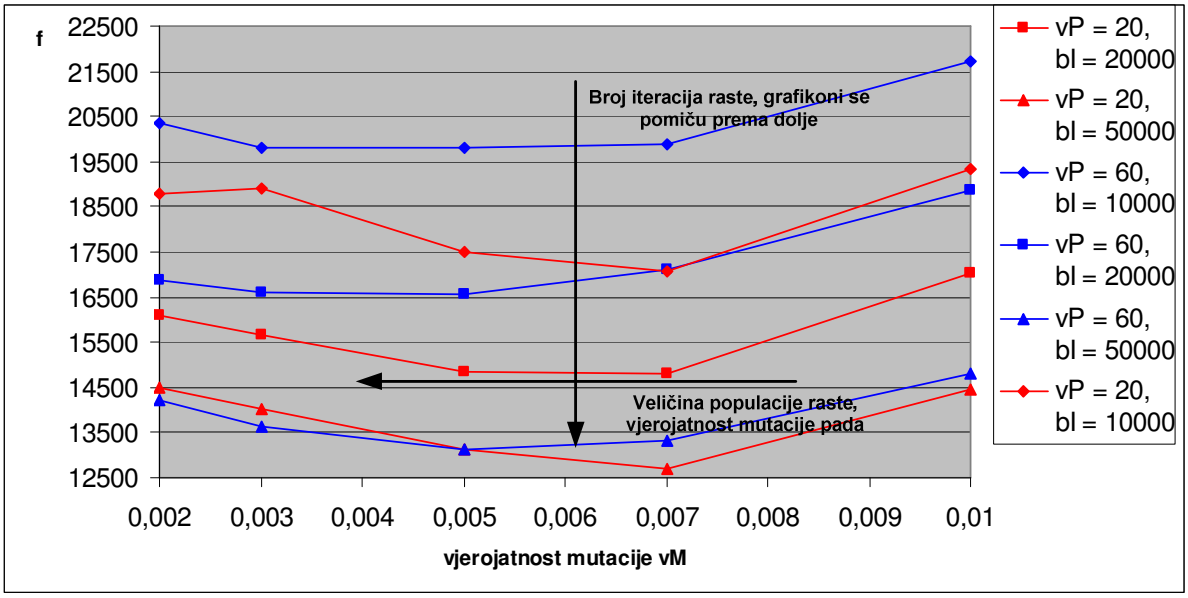
SLIKA 5.6 b) Detaljniji prikaz promjene sa slike 5.6 a)

Slika 5.6 predstavlja usporedbu rada genetskog algoritma za veličine populacije 20 i 60 jedinki i za broj iteracija 10.000, 20.000 i 50.000.

Na slici 5.6 crvenom su bojom označeni grafikoni koji predstavljaju rezultate dobivene za veličinu populacije 20. Plavom bojom označeni grafikoni koji se odnose na veličinu populacije 60. Broj iteracija se razlikuje prema oznakama točaka, za 10.000 iteracija oznaka je romb, za 20.000 oznaka je kvadat, a za 50.000 oznaka je trokut.

Iz slike je vidljivo kako je najbolji rezultat postignut za veličinu populacije od samo 20 jedinki i za broj iteracija od 50.000. Taj rezultat je postignut za vjerojatnost mutacije od 7 ‰.

Na slici se također lijepo vidi da se s porastom broja iteracija postižu i bolja rješenja. Grafikon se spušta prema dolje. Osim toga na slici se vidi da su i minimumi za različite vrijednosti veličine populacije postignuti za različite vrijednosti vjerojatnosti mutacije. Za veličinu populacije od 20 jedinki minimum postiže za 7‰, dok se za veličinu populacije 60 taj isti minimum postiže za 5‰. Slika 5.7 pokazuje kako se kreću grafikonu u ovisnosti o promjeni parametara veličine populacije i broja iteracija.



SLIKA 5.7 – Kretanje grafikona s promjenama promatranih parametara

Usporedbom slika 5.5 i 5.7 jasno je vidljivo, da iako grafikoni nemaju isti oblik, ponašanje genetskog algoritma je jednako. Povećanjem broja iteracija postižu se bolji rezultati. Za veće populacije potrebno je smanjiti vjerojatnost mutacije kako bi se postigla ista kvaliteta rješenja.

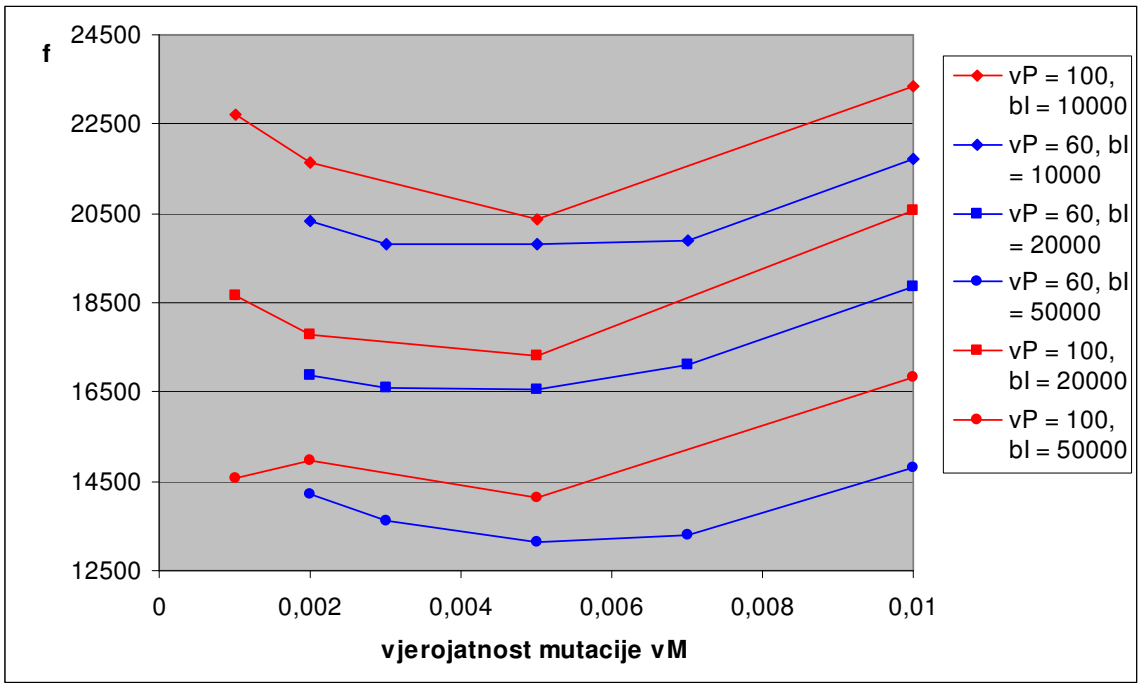
U tablici 5.5 se nalaze i vrijednosti funckije dobrote postignuti za pojedini set parametara.

TABLICA 5.5 – Eksperimentalni podaci za slike 5.6 i 5.7

vP = 20		**bI**		
vM		10.000	20.000	50.000
	0,002	18776	16099	14510
	0,003	18893	15668	14015
	0,005	17490	14832	13122
	0,007	17090	14808	12708
	0,01	19352	17037	14446

vP = 60		**bI**		
vM		10.000	20.000	50.000
	0,002	20344	16863	14205
	0,003	19823	16610	13624
	0,005	19808	16543	13138
	0,007	19890	17121	13314
	0,01	21714	18868	14788

Na slici 5.8 se nalazi usporedba rada algoritma za vrijednosti parametra veličine populacije 60 i 100 jedinki.



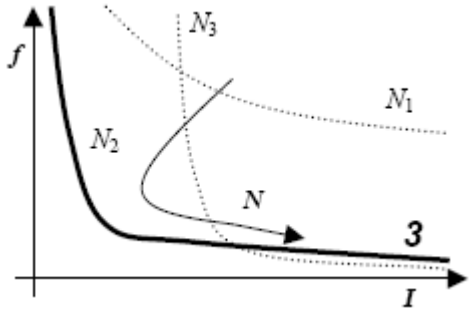
SLIKA 5.8 - Usporedba rada algoritma za veličinu populacije 60 i 100 jedinki

TABLICA 5.6 – Podaci za sliku 5.8

vP = 100		bI		
vM		10.000	20.000	50.000
	0,002	23360	20556	16844
	0,005	20380	17319	14124
	0,007	21643	17792	14952
	0,01	22696	18666	14566

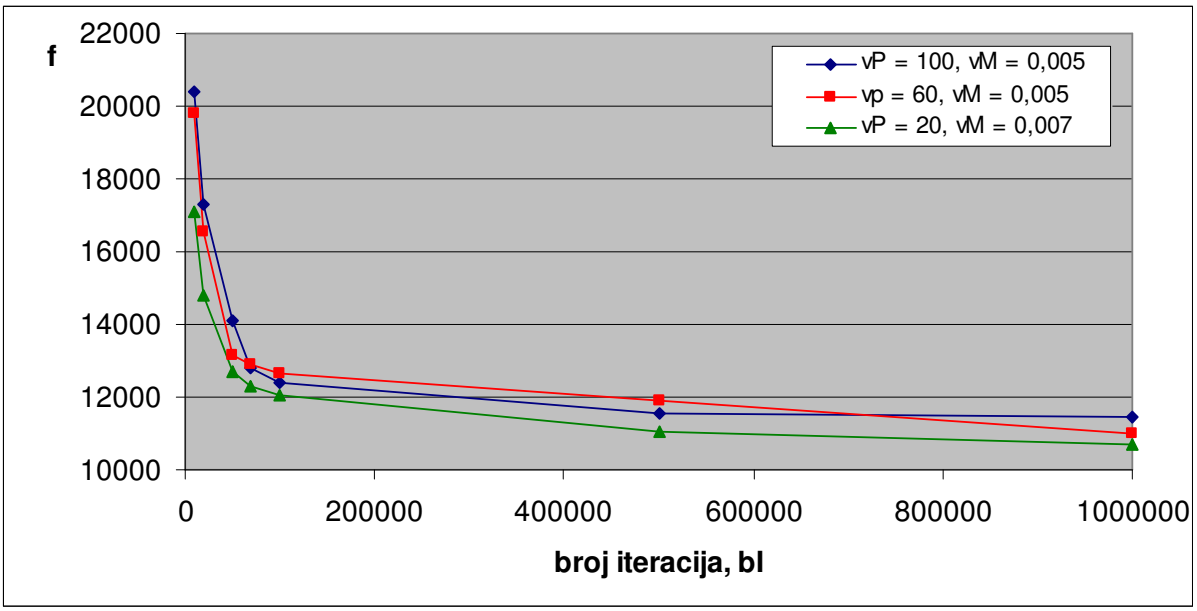
Na slici 5.8 se vidi kako je ponašanje algoritma slično kao i za analizirani slučaj. Vjerojatnosti mutacije za koje se postiže minimum u ovome slučaju su jednake i iznose 5‰ . Stoga bi bilo potrebno napraviti dodatna mjerenja sa finijim vrijednostima vjerojatnosti mutacije oko vrijednosti 5‰. Time bi se dobile optimalne vrijednosti rješenja za veličine populacije 60 i 100 jedinki. I dalje vrijedi ponašanje algoritma da veći broj iteracija znači bolje rješenje. Također, za vrijednost veličine populacije 100 i broj iteracija 50.000 za 1‰ postiže se bolji rezultat nego za 2‰. To je događaj koji se može dogoditi zato jer genetski algoritam barata se slučajnim vrijednostima pa uvijek postoji mala vjerojatnost da će genetski algoritam pronaći bolje rješenje negdje izvan predviđenih kretanja krivulja.

5.4.2. Analiza parametra broj iteracija (bl)



SLIKA 5.9 – Očekivano ponašanje za broj iteracija

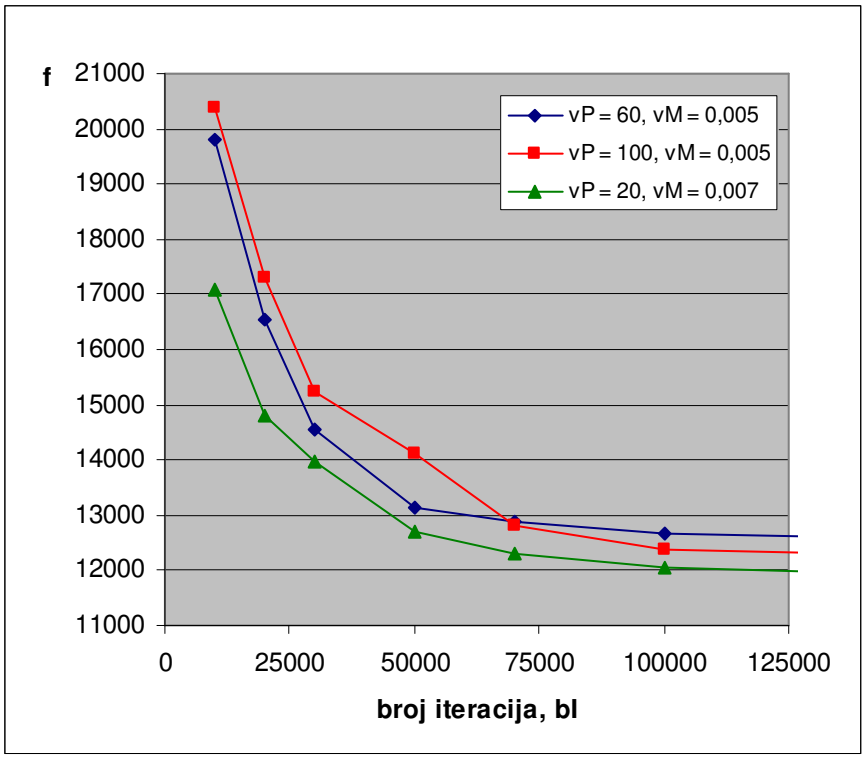
Mjerenjima su postignuti rezultati prikazani na slici 5.10.



SLIKA 5.10 – Evolucijski proces

TABLICA 5.7 – Podaci za sliku 5.10

		$v_P = 100, v_M = 0,005$	$v_P = 60, v_M = 0,005$	$v_P = 20, v_M = 0,007$
bl	**10000**	20380	19808	17090
	20000	17319	16543	14808
	50000	14124	13138	12708
	70000	12811	12896	12322
	100000	12389	12654	12056
	500000	11565	11924	11029
	1000000	11437	10993	10697



SLIKA 5.11 – Detaljniji prikaz evolucijskog procesa u području najbrže konvergencije

Tablica 5.8 prikazuje za koje su sve brojeve iteracija napravljena mjerenja.

TABLICA 5.8 – brojevi iteracija

Broj iteracija, bl
10.000
20.000
50.000
70.000
100.000
500.000
1.000.000

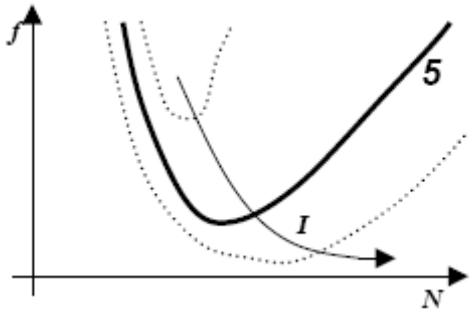
Na slici 5.10 se vidi kako s povećanjem broja iteracija genetski algoritam pronalazi sve bolja rješenja. Za veličinu populacije od 20 članova i vjerojatnost mutacije od 7‰ pronalaze se bolja rješenja od onih za veće populacije. Razlog tome je vjerojatnost mutacije koju bi trebalo smanjiti za veće populacije kako bi se postigla jednako kvalitetna rješenja.

Već za 200.000 iteracija postižu se rješenja koja su malo lošija od rješenja koja se postižu za 1.000.000 iteracija. To samo govori o tome da je bitno na početku odrediti koliko se je potrebno približiti optimumu zato jer je vrijeme izvođenja algoritma za 200.000 iteracija znatno kraće od vremena koje je potrebno algoritmu da napravi 1.000.000 iteracija.

Treća stvar koja se može primjetiti je da se postižu bolja rješenja za populaciju od 20 članova nego rješenja za 60 i 100 članova. Kako broj iteracija raste to se izjednačava. Broj iteracija postaje dovoljno velik da veće populacije mogu konvergirati. Za dovoljno velik broj iteracija potrebno je kvalitetno odrediti vjerojatnost mutacije kako bi se postizala zadovoljavajuća rješenja.

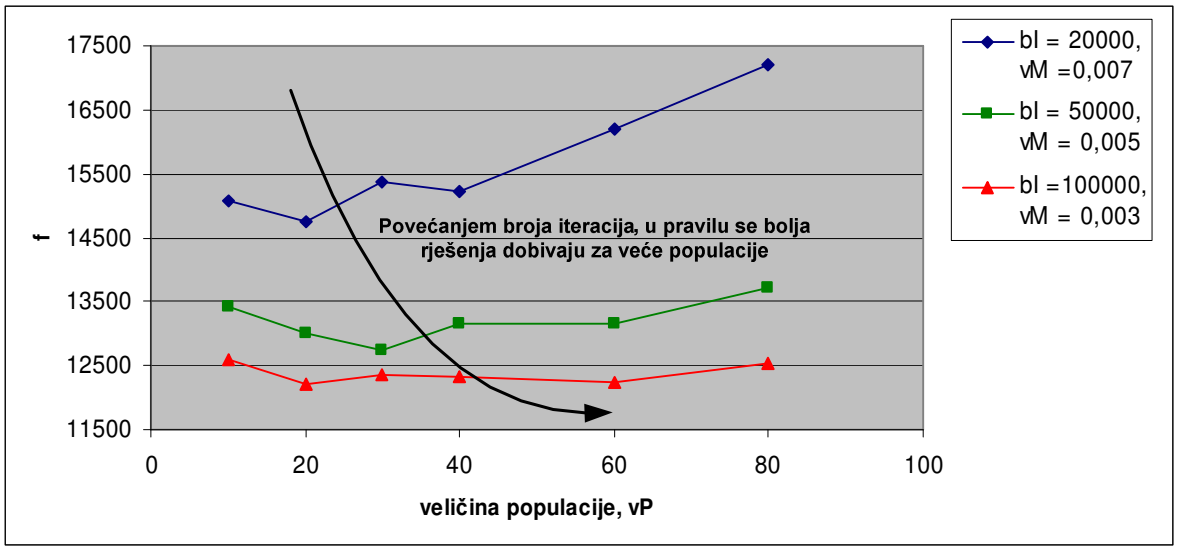
Slika 5.11 prikazuje područje najbrže konvergencije genetskog algoritma. Za veličinu populacije 20 jedinki postižu se najbolja rješenja. Veće populacije počinju davati rješenja jednake kvalitete za 75.000 i više iteracija. Nagib i brzina konvergencije algoritma ovisi o veličini populacije što se vidi iz nagiba priloženih krivulja. Za veći broj iteracija nagib je blaži nego za manji broj iteracija (75..000 i više).

5.4.3. Analiza parametra veličina populacije (vP)



SLIKA 5.12 – Očekivano ponašanje za veličinu populacije

Eksperimentalni rezultati su prikazani na slici 5.13.

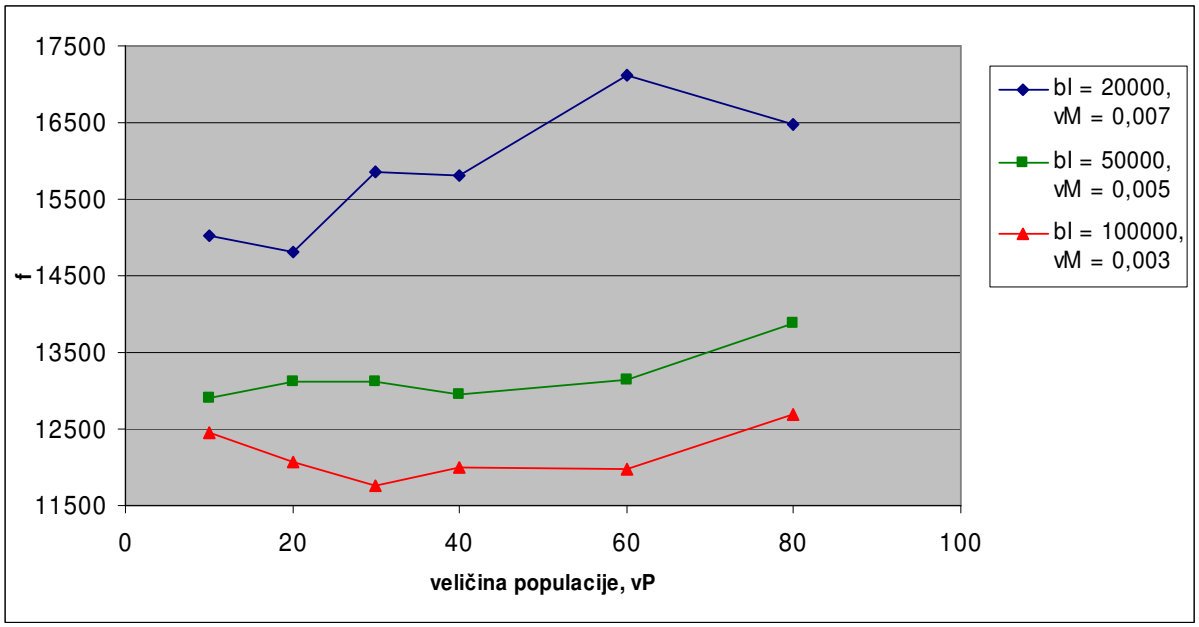


SLIKA 5.13 – Odnos veličine populacije i kvalitete rješenja

TABLICA 5.9 – Podaci za sliku 5.13

		bI = 20000, vM = 0,007	bI = 50000, vM = 0,005	bI = 100000, vM = 0,003
vP	**10**	15083	13415	12581
	20	14765	13017	12211
	30	15361	12730	12362
	40	15226	13166	12316
	60	16201	13150	12225
	80	17204	13727	12524

Slika 5.13 također pokazuje kako se za veći broj iteracija postiže bolje rješenje. Crna strelica na slici prikazuje kako se kreću rješenja genetskog algoritma ako se povećava broj iteracija. Dakle, rješenja padaju, teže optimumu, a postižu se za veće vrijednosti veličine populacije. Za 20 i 50 tisuća iteracija ovo se ponašanje dosljedno prati. Za 100 tisuća iteracija događa se slučaj u kojemu se jednako kvalitetno rješenje postiže i za veličinu populacije od 20 jedinki i od 60 jedinki. Na slici 5.14 se nalaze grafikoni za iste vrijednosti parametara, ali za mjerenja koja su dodatno napravljena. Na slici 5.14 se može zamjetiti kako je ponašanje algoritma vrlo slično ponašanju koje se vidi na slici 5.13. Postoje određene razlike u kvaliteti rješenja i u izgledu krivulja. Na slici 5.13 algoritam za 100.000 iteracija najbolje rješenje postiže za veličinu populacije 30. Za veličinu populacije 60 rješenje je nešto slabije. Razlika je uočljiva i za 20.000 iteracija za veće populacije gdje na slici 5.14 algoritam postiže bolje rješenje za veličinu populacije 80 nego za veličinu populacije 60. To je samo pokazatelj kako je jako bitno da se za isti set parametara uvijek mora napraviti više mjerenja i srednja vrijednost uzeti za rješenje zbog visoke razine stohasitke kojom se genetski algoritam koristi pri pronalasku optimalnog rješenja.



SLIKA 5.14 – Ponovljena mjerenja za isti set parametara kao i na slici 5.13

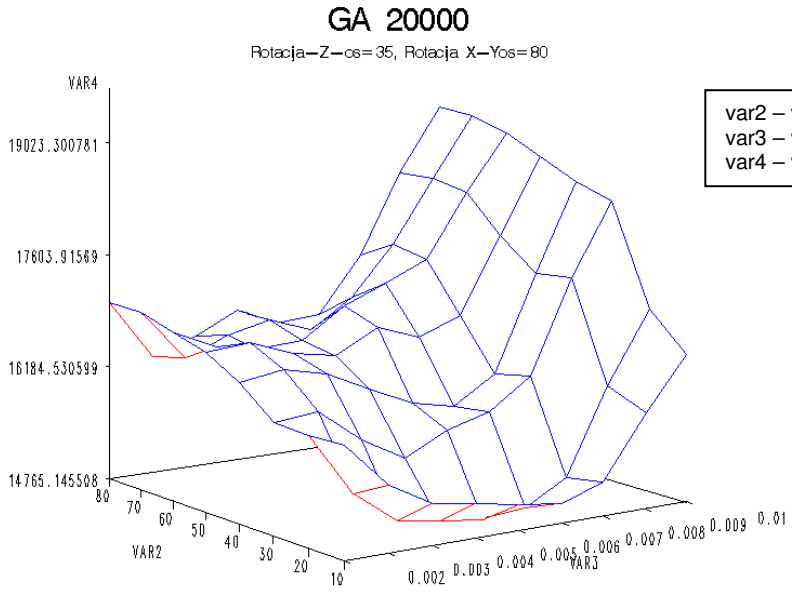
TABLICA 5.10 – Podaci za sliku 5.14

		bI = 20000, vM = 0,007	bI = 50000, vM = 0,005	bI = 100000, vM = 0,003
vP	**10**	15014	12916	12459
	20	14808	13122	12073
	30	15857	13125	11757
	40	15804	12947	11995
	60	17121	13138	11978
	80	16485	13891	12685

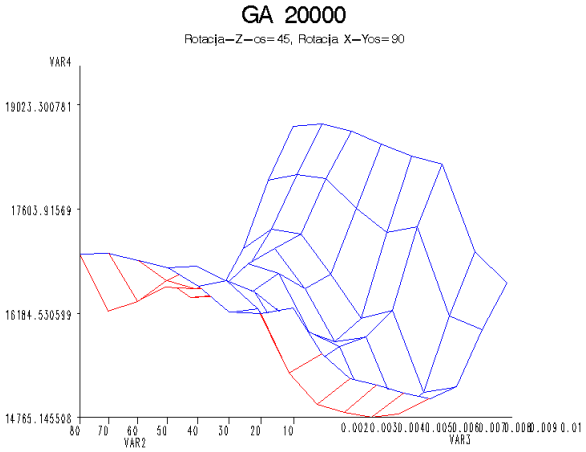
5.4.4. 3D prikazi analiziranih rješenja

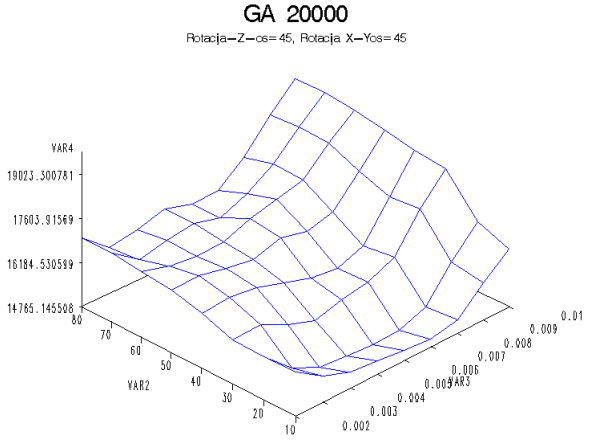
U analizi rada genetskog algoritma promatrana su tri parametra, vjerojatnost mutacije, broj iteracija i veličina populacije. Sve dosadašnje analize su se odnosile na promatranje jednog parametra uz držanje ostalih parametara fiksnim. Moguće je pogledati kako izgleda ploha međusobne ovisnosti dva parametra dok se treći parameter drži fiksnim. Ovdje je to napravljeno tako da se broj iteracija drži fiksnim dok se u xy ravnini nalaze vrijednosti za vjerojatnost mutacije i veličinu populacije. Analiza je napravljena za 20.000, 50.000 i 100.000 iteracija. Moguće je da neke od vrijednosti prikazane na 3D grafovima ne odgovaraju u potpunosti pripadajućim vrijednostima u prethodnoj analizi. To je zato jer su neka mjerenja ponovno napravljena kako bi se na grafu bolje istaknula kretanja rješenja u ovisnosti o promjenama parametara.

Prvi set slika prikazuje rezultate postignute za 20.000 iteracija. Drugi set slika prikazuje rezultate postignute za 50.000 iteracija. Treći set slika prikazuje rezultate postignute za 100.000 iteracija.



a)





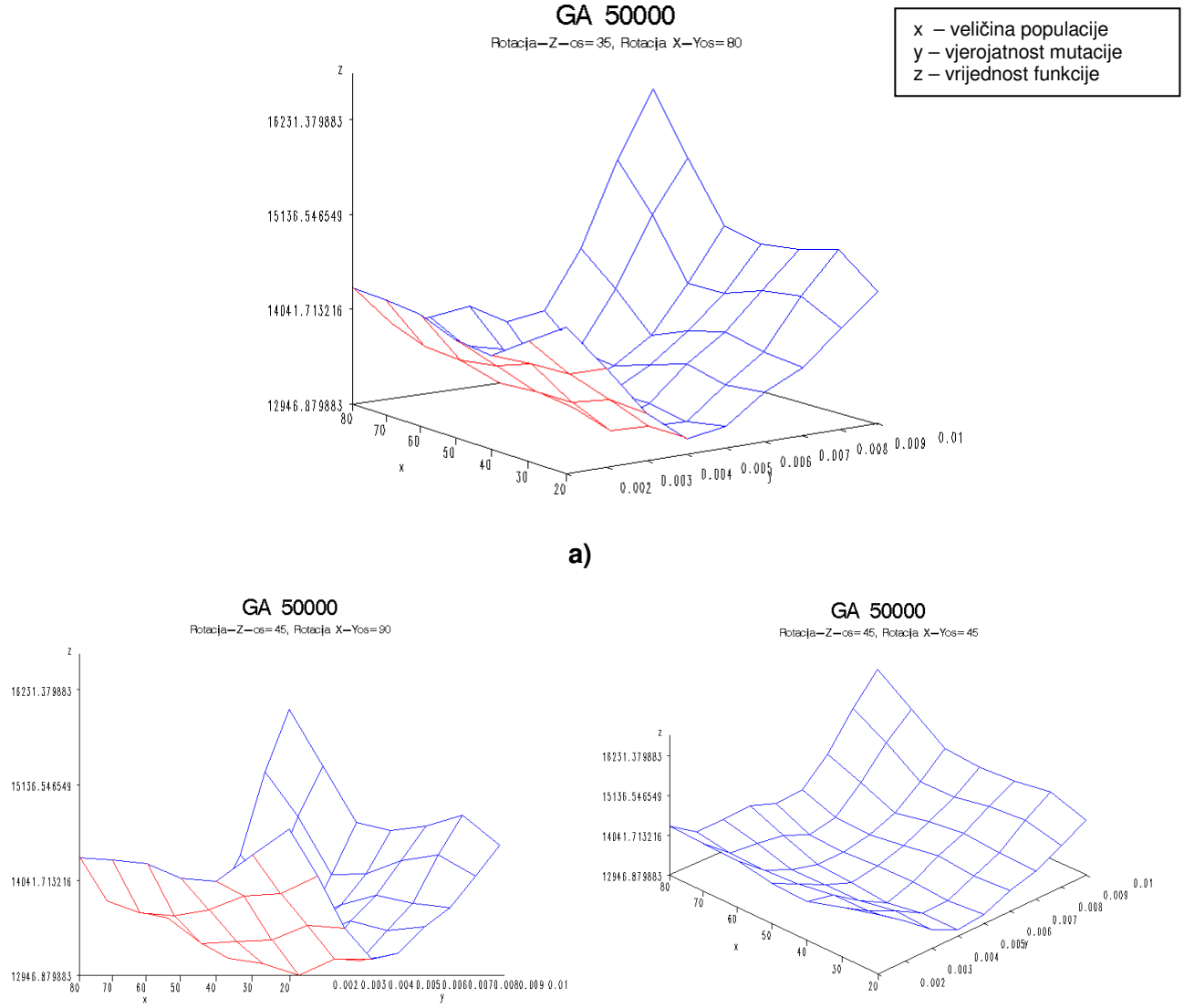
b) **c)**

SLIKA 5.15 – 3D prikaz rezultata za 20.000 itreacija

Slika 5.15. prikazuje rezultate dobivene za 20.000 iteracija. Veličina populacije se kreće od 10 do 80 jedinki, a vjerojatnost mutacije od 2‰ do 1%. Ploha je prikazana iz tri različita kuta na slikama 5.15 a), b) i c).

Na slici 5.15 se može jasno vidjeti kretanje ka najboljem rješenju kako se mijenjaju parametri veličina populacije i vjerojatnost mutacije. Za veće populacije rješenje je lošije i pada prema najboljem za manje vrijednosti veličine populacije. Najbolje rješenje se postiže za veličinu populacije 15 i za vjerojatnost mutacije od 7‰. Za veće populacije se dobivaju lošija rješenja. 20.000 iteracija je premalo za konvergenciju velikih populacija. Za najmanje vrijednosti

veličine populacije postoji lagani rast prema lošijim rješenjima. Najmanje populacije zaglave u lokalnom optimumu iz kojeg se ne mogu izvući.



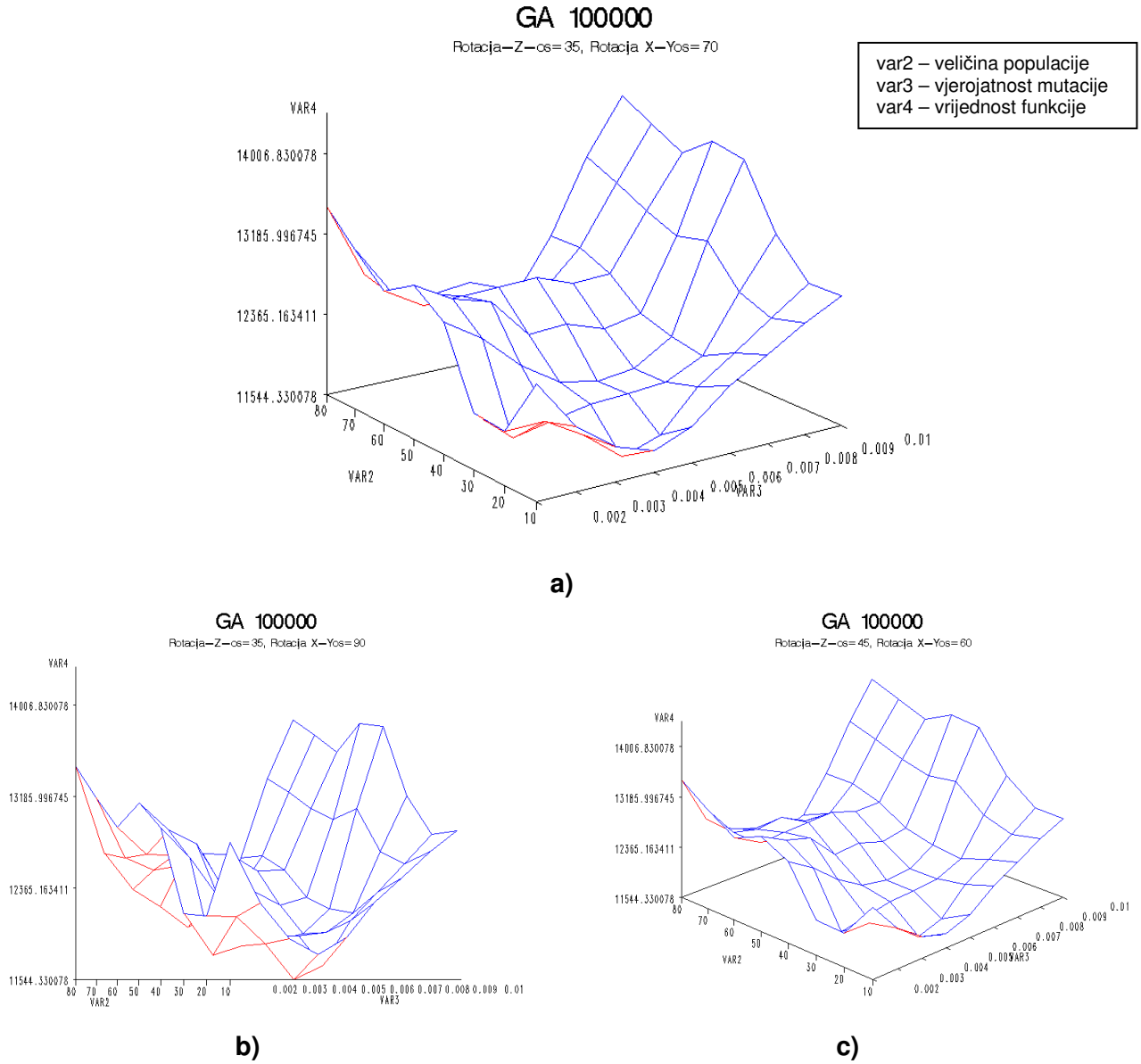
SLIKA 5.16 – 3D prikaz rezultata za 50.000 iteracija

Slika 5.16. prikazuje rezultate dobivene za 50.000 iteracija. Veličina populacije se kreće od 20 do 80 jedinki, a vjerojatnost mutacije od od 2‰ do 1%. Ploha je prikazana iz tri različita kuta na slikama 5.16 a), b) i c).

Za 50.000 iteracija izgled plohe je drugačiji nego za 20.000 iteracija. Generalno kretanje rješenja je jednako. Na rubovima se nalaze lošija rješenja, dok se negdje unutar plohe

nalaze bolja rješenja. U ovome slučaju najbolje postignuto rješenje ima vrijednost 12947 i postignuto je za veličinu populacije od 40 članova i vjerojatnost mutacije 5‰.

U analizama svakog parametra zasebno za 50.000 iteracija je postignuto najbolje rješnje 12707, za veličinu populacije 20 i vjerojatnost mutacije 7‰. Do ove razlike je došlo zbog toga što su za 3D analizu napravljena dodatna ponovljena mjerenja kako bi se bolje istaknula generalna kretanja po plohi.



SLIKA 5.17 – 3D prikaz rezultata za 100.000 iteracija

Slika 5.17. prikazuje rezultate dobivene za 100.000 iteracija. Veličina populacije se kreće od 10 do 80 jedinki, a vjerojatnost mutacije od od 2‰ do 1%. Ploha je prikazana iz tri različita kuta na slikama 5.17 a), b) i c).

Ploha na slici 5.17 oblikom izgleda drugačije nego prethodne dvije. Kretanja po plohi su jednaka kao i za 20.000 i 50.000 iteracija. Najbolje postignuto rješenje je 11544 i postignuto je za veličinu populacije 20 i vjerojatnost mutacije 5‰. Očekivana vrijednost možda nije bila tako mala populacija, ali u genetskom algoritmu vrijedi da se za različite vrijednosti parametara mogu dobiti isti ili slični rezultati. Najbolje postignuto rješenje za veličinu populacije 60 i vjerojatnost mutacije 3‰ iznosi 11978. 11978 je i dalje znatno bolje od rješenja postignutih za 20 i 50 tisuća iteracija.

Najbolja rješenja se postižu za vrijednosti parametra vjerojatnosti mutacije koje se se nalaze između 3‰ i 7‰. Unutar ovih vrijednosti vjerojatnosti mutacije bi trebalo tražiti optimalno rješenje

Veličina populacije ovisi o broju iteracija. Većim populacijama je potrebno više iteracija kako bi konvergirale ka kvalitetnim rješenjima. Ova dva prametra je potrebno međusobno usklađivati u potrazi za optimalnim rješenjem.

Provedena 3D analiza potvrđuje pretpostavke poznate o ponašanju genetskog algoritma. Za veći broj iteracija postiže se bolje rješenje. Većim populacijama potreban je veći broj iteracija kako bi one konvergirale prema optimumu. Za veće populacije potrebna je manja vrijednost parametra mutacije kako bi se postigla kvalitetna rješenja. U tablici 5.11 se nalazi usporedni prikaz najboljih rješenja postignutih za svaki set grafova u kojemu se jasno vide upravo izvedeni zaključci.

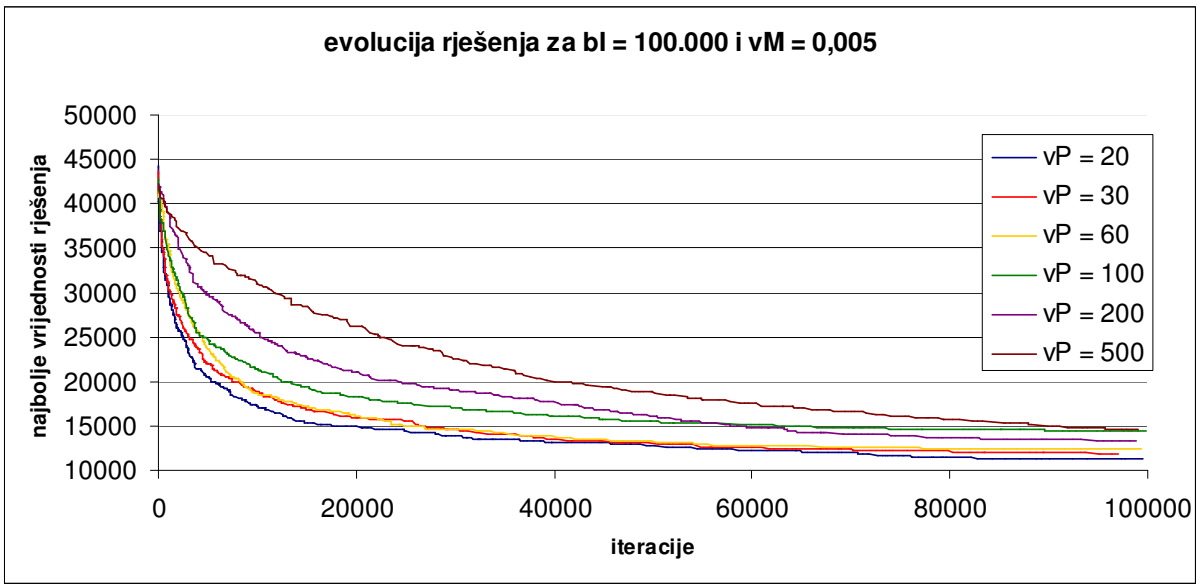
TABLICA 5.11 – Usporedni prikaz najboljih rješenja prikazanih na 3D grafovima

Broj iteracija	20.000	50.000	100.000
Rješenje	14689	12947	11978
Veličina populacije	15	40	60
Vjerojatnost mutacije	7‰	5‰	3‰

Sve vrijednosti za koje su postignuti ovi rezultati se nalaze na prliženom CDu.

5.4.5. Evolucija rješenja u jednom izvođenju genetskog algoritma

Genetski algoritam kreće sa generiranjem slučajne populacije. Ne vodi se računa o tome kakve su jedinke, već je cilj generirati broj jedinki zadan veličinom populacije. Uvijek postoji mala vjerojatnost da će se prilikom tog generiranja stvoriti i jedna ili više jedinki vrlo visoke dobrote. No to je rijetkost i njačešće nastanu jedinke koje su daleko od optimuma. Na slici 5.18 se nalazi usporedni prikaz konvergencije genetskog algoritma za 100.000 iteracija i za veličine populacije 20, 30, 60 ,100, 200 i 500 jedinki.



SLIKA 5.18 – Evolucijski proces za konstante vrijednosti bl i v_m

Iz slike 5.18 se može vidjeti kako za manje vrijednosti veličine populacije algoritam brže konvergira. U manjem broju iteracija postižu se bolji rezultati. Za veće populacije je potreban veći broj iteracija kako bi algoritam došao do rješenja koje je jednako kvalitetno kao i rješenje koje je postignuto za manje populacije. To se najbolje vidi na nagibima krivulja koje iste imaju kako broj iteracija raste. Krivulja koja pokazuje veličinu populacije od 200 jedinki ima puno blaži nagib nego krivulja koja predstavlja populaciju od 60 jedinki. Za dovoljno velik broj iteracija, veće populacije bi u konačnici postigle bolja rješenja.

6. Zaključak

U ovome diplomskom radu napravljena je analiza podešavanja parametara veličine populacije, broja iteracija i vjerojatnosti mutacije na problemu trgovačkog putnika. Praktični dio rada pokazao je ono što se očekivalo: da su parametri međusobno zavisni. Promjena jednog parametra najčešće povlači za sobom i promjenu ostalih parametara ako se želi zadržati ista kvaliteta rješenja.

Potvrdile su se otprije poznate činjenice o radu genetskih algoritama, a to su:

- veći broj iteracija, u pravilu znači bolje rješenje;
- za veće populacije treba povećati broj iteracija i smanjiti vjerojatnost mutacije;
- jednako kvalitetna rješenja se mogu postići za različite skupove parametara.

Genetski algoritmi su dobar i kvalitetan alat za rješavanje raznih vrsta problema, ali su alat kojeg treba znati dobro iskoristiti ako se s njime žele dobiti kvalitetni rezultati. Također je bitno odrediti točnost rješenja jer se time može znatno skratiti vrijeme traženja skupa parametara koji daju traženu točnost. Brojčane vrijednosti koje su postignute u ovome radu se odnose na genetski algoritam opisan u poglavlju 5. Smjernice koje su ovdje potvrđene su ono što treba slijediti prilikom traženja optimalnih parametara u drugim primjenama genetskih algoritama.

7. Literatura

[1] Konstantin Boukreev. “Genetic Algorithm and Traveling Salesman Problem”, http://www.generation5.org/content/2001/tspapp.asp, 2001.

[2] Kylie Bryant, Arthur Benjamin. *“Genetic Algorithms and the Traveling Salesman Problem”*, http://www.math.hmc.edu/seniorthesis/archives/2001/kbryant/kbryant-2001-thesis.pdf, 2000.

[3] Marin Golub, doc. dr.sc. *“Genetski algoritam”*, http://www.zemirs.fer.hr/~golub/ga.html, 2004.

[4] Hiroaki Sengoku, Ikua Yoshihara. *“A Fast TSP Solver Using GA on Java”* http://www.gcd.org/sengoku/docs/arob98.pdf, 1998.

[5] Andy Thomas. “*Solving The Travelling Sales Man Problem Using Genetic Algorithm”,* http://www.generation5.org/content/2001/ga_tsp.asp, 2001.

[6] http://www.tsp.gatech.edu/problem/index.html

[7] Marin Golub, doc. dr.sc. *“Paralelni genetski algoritmi”*, http://www.zemirs.fer.hr/~golub/ga.html, 2004.

[8] Vedran Lovrečić, seminarski rad, *”Genetski algoritam u primjeni”,* http://www.zemirs.fer.hr/~golub/ga.html, 2005.

[9] Loenard J. Testa, Albert C. Esterline, Gerry V. Dozier, Abdollah Homaifar: A Comparison of Operators for Solving Time dependent Traveling Salesman Problems Using Genetic Algorithms. GECCO 2000: 995-1102