SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1882

AUTOMATSKO UPRAVLJANJE UZ POMOĆ EVOLUCIJSKIH ALGORITAMA

Igor Horvatić

Zagreb, studeni 2010.

Mentor rada:
doc. dr. sc. Domagoj Jakobović

Voditelj rada:
doc.dr.sc Domagoj Jakobović

Sadržaj

Uvod....1

1. Genetski algoritam....2

1.1. Primjena genetskog algoritma na automatsko upravljanje....6

1.1.1. Generiranje početne populacije....8

1.1.2. Određivanje dobrote svakog kromosoma....9

1.1.3. Selekcija jedinki....11

2. Harmonijsko pretraživanje....14

2.1. Primjena harmonijskog pretraživanja na automatsko upravljanje....17

Diferencijalni operator mutacije 18

3. Grafičko sučelje programa....19

4. Rezultati izvođenja i odnos odabira parametara na kvalitetu rješenja i brzinu izračunavanja 22

4.1. Utjecaj broja pješaka na broj generacija....23

4.2. Kretanje prosječne dobrote u odnosu na broj generacija....25

4.3. Utjecaj diskretnog vremenskog intervala na broj generiranih novih jedinki....27

4.4. Utjecaj veličine populacije na broj izračunatih generacija i improvizacija....29

4.5. Utjecaj vjerojatnosti odabira iz memorije (HMCR) na kvalitetu rješenja....31

4.6. Utjecaj vjerojatnosti prilagodbe note (PAR) i vjerojatnosti selekcije na kvalitetu rješenja....32

4.7. Utjecaj mutacije/diferencijalnog operatora na kvalitetu rješenja....34

Zaključak....36

Literatura....37

Skraćenice....38

Popis stranih izraza....39

Dodatak....40

Uvod

Optimizacija procesa je sve zahtjevniji zadatak zbog sve složenijih struktura upravljanja, proizvodnje, raspodjele aktivnosti i višedimenzionalnih problema. Prirodom inspirirani algoritmi optimizacije su sve češći odabir za rješavanje problema koje ljudi ili klasične matematičke metode optimiranja ne uspijevaju riješiti. Radi svoje učinkovitosti i jednostavnosti, razvijeni su evolucijski algoritmi kao simulacija prirodnih procesa optimizacije. Evolucija je kao prirodna optimizacija traženje najsposobnijih jedinki za preživljavanje u postavljenim uvjetima nekog okoliša. Jedinke koje čine populaciju su izložene mutacijama, križanjima, selekciji i reprodukciji. Jedinke u evolucijskim algoritmima predstavljaju moguća rješenja zadanog problema. Svako rješenje, poput živih bića, ima svoju sposobnost prilagodbe na okoliš. Funkcija dobrote koja procjenjuje sposobnost opstanka jedinki u prirodi je život, broj potomaka i smrt. U evolucijskom algoritmu funkcija dobrote pridjeljuje svakoj jedinki vjerojatnost preživljavanja ovisno o problemu kojemu se traži rješenje.

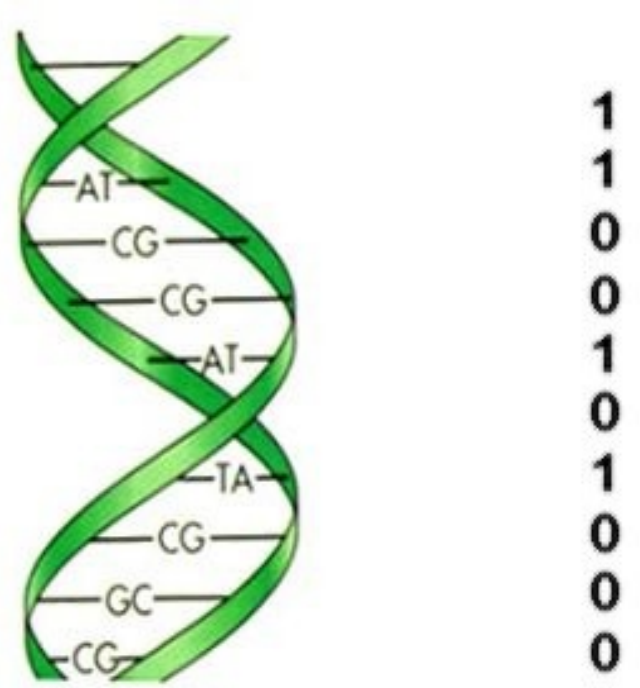
Genetski algoritam i *algoritam harmonijskog pretraživanja* su dva primjera evolucijskih algoritama. Genetski algoritam primjenjuje nad jedinkama genetske operatore dok ne razvije dovoljno dobru jedinku od koje ne može u razumnom vremenu razviti bolju. Algoritam harmonijskog pretraživanja imitira proces improvizacije melodije kod glazbenika koji traže savršen ustroj nota koji najbolje odgovara ukusu slušatelja.

U ovom radu oba algoritma su uspoređena u kvaliteti i brzini traženja dovoljno dobrog vektora kretanja u simuliranom vozilu. Automatsko upravljanje vozilom je ostvareno odabirom vektora smjera i akceleracije koji ne vodi u sudar sa preprekama. Rad je osmišljen kao usporedba algoritama na istom problemu i obradom rezultata iznosi razlike i prednosti pojedinog algoritma.

Algoritmi i njihova implementacija su detaljno opisani u prvom i drugom poglavlju, a uspoređeni u četvrtom. Treće poglavlje je fokusirano na grafičko i upravljačko sučelje programske izvedbe.

1. Genetski algoritam

Evolucijski algoritmi čine relativno novu skupinu optimizacijskih algoritama. Osnovna ideja je oponašanje evolucijskog procesa pronalaženja optimalnog rješenja kao prilagodbe na okolinu. Poput adaptacije živog svijeta na svoju okolinu, '70tih godina 20. stoljeća znanstvenici su opisali proces optimizacije po uzoru na evoluciju. Prve pokuse je izveo John Holland na Sveučilištu Michigan proučavajući stanične automate i 1975. godine postavio temelje područja primjene evolucije u rješavanju problema optimizacije. Prve konferencije o evolucijskim algoritmima su održane tek sredinom '80tih, a prvi komercijalni programi koji koriste genetske algoritme pojavili su se početkom '90tih.godina Nagli razvoj računala omogućio je razvoj masivnih numeričkih metoda s povećanjem količine računalnih operacija za više redova veličine što je omogućilo daljnji razvoj algoritama. Najčešće korišten evolucijski algoritam je genetski algoritam[3].



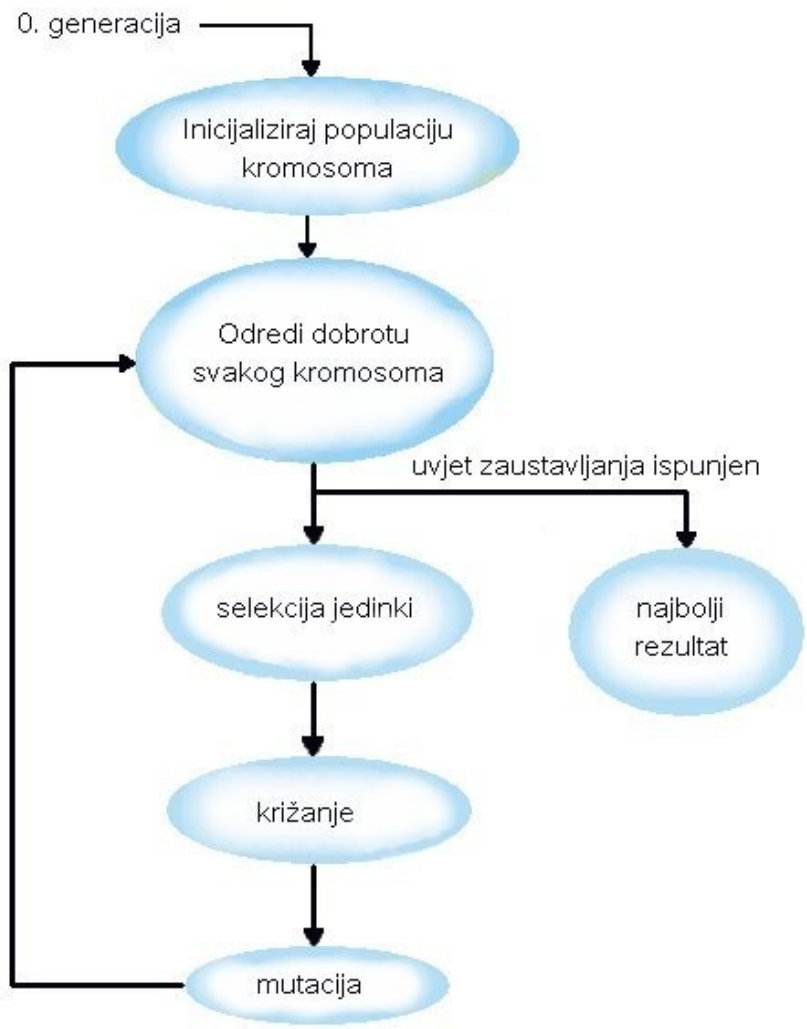
Slika 1-1 DNK reprezentirana binarnim znamenkama

Binarne znamenke dobro reprezentiraju kombinacije dušikovih baza u DNK lancima koje imaju samo dvije moguće kombinacije. Na slici 1-1 pokazana je interpretacija kombinacija baza pomoću binarnog niza. Izmjenom jedne kombinacije baza mjenjaju se svojstva lanca koji sadrži te baze. Niz tada sadrži neki drugi slijed baza koje u biološkom smislu prenose

drugačiju informaciju od početnog niza. Prenašanjem ideje u računalo moguće je binarnim nizom reprezentirati DNK molekulu i nad tim nizom primjenjivati simulirane genetske operacije. Ovisno o problemu, specifičnom interpretacijom binarnog niza moguće je modelirati informacije koje bi bile sadržane u nizu.

Genetski algoritam oponaša evoluciju tj. prilagodbu jedinki na svoju okolinu kroz generacije. Svakom generacijom nastaju bolje jedinke prilagođenije okolini. Okolina je modelirana problemom koji treba riješiti tj. funkcijom cilja koju treba optimirati, a ključni kriterij evaluacije je funkcija dobrote (engl. *fitness function*). Funkcija dobrote mora primjereno ocijeniti jedinke u generaciji kako bi ih gradirala od najboljih do najgorih. Najbolje jedinke, po teoriji evolucije, opstaju i nastavljaju svoj razvoj, dok najlošije odumiru tj. bivaju zamijenjene. Jedinka je standardno prikazana kao niz binarnih znamenki koje se preračunaju u vrijednosti domene. Mutacija se simulira izmjenom bita u nizu s određenom vjerojatnošću. Uz dovoljno dug niz i visoku vjerojatnost mutacije, moguće je i da više bitova bude promijenjeno u istoj jedinki. Križanje je proces kopiranja dijela niza bitova jedne jedinke i dijela niza bitova druge jedinke, kako bi se kombiniranjem ta dva dijela niza stvorio novi niz, koji je različit od početna dva. Bitno je da novi niz bude jednake duljine kao i odabrana dva 'roditelja' i uostalom kao duljine svih jedinki. Križanjem se stvara nova jedinka koja će zamijeniti neku od selektiranih za 'odumiranje'. Uobičajena metoda križanja za manje jedinke uključuje i slučajno generiranu jedinku koja se onda isključivo-ili operatorom kombinira s dvije odabrane jedinke. Time nastaje posve nova jedinka koja može unjeti neko bolje rješenje u populaciju.

Stvaranjem nove jedinke obavljena je jedna iteracija genetskog algoritma. Kada se stvori novih n jedinki, gdje je n veličina populacije, izvršena je jedna generacija genetskog algoritma izračunavanjem funkcije dobrote n puta [1].



Slika 1-2 Genetski algoritam

Tijekom izvođenja algoritma, u svakoj generaciji određeno je najbolje rješenje, koje se pohranjuje u zasebnu varijablu. Ukoliko kroz algoritam evoluira bolje rješenje, ono zamjenjuje do tada najbolje pronađeno. Najbolje do tog trenutka pronađeno je pohranjeno u zasebnu varijablu i ako evoluira bolje kroz algoritam, zamjenjuje najbolje pronađeno. Tako najbolje rješenje ostaje sačuvano čak i ako ne postoji u populaciji uslijed primjene genetskih operatora.

Odabir jedinki koje će ostati sačuvane moguće je ostvariti pomoću sljedećih metoda:

- Generacijski (engl. *generational*) odabir
- Eliminacijski (engl. *steady*-state) odabir

Generacijski odabir prenosi kopije jedinki iz jedne generacije u drugu s određenom vjerojatnošću. Nakon izračunavanja dobrote pojedinih jedinki, veća dobrota jedinke daje veću

vjerojatnost prijenosa ili 'preživljavanja' u sljedećoj generaciji. Dobrote se zbroje, zbroj se normalizira i položi na pravac sa vrijednostima od 0 do 1. Svakoj jedinki je pridružena vrijednost tj. udio na tom pravcu. Time se određuje vjerojatnost da ta jedinka bude odabrana za sljedeću generaciju. Primjerice od zbroja dobrote populacije koji je jednak 20, jedinka koja ima dobrotu 5 ima 25% vjerojatnost da će biti odabrana. Ako je druga u populaciji, a prva ima dobrotu 10, onda će biti odabrana ako je slučajni broj iz intervala [0.5 , 0.75). Prvoj jedinci naravno odgovara broj [0 , 0.5) Takav odabir se naziva generacijski jednostavni odabir (engl. *roullete-wheel selection*). Problem nastaje ako se stvori jako dobra jedinka u populaciji vrlo loših, jer je tada moguće da samo ta jedna jedinka ispuni sljedeću generaciju i time zaguši novu populaciju s mnogo istih rješenja koje imaju jednaku dobrotu.

Eliminacijski odabir podrazumijeva uklanjanje nepoželjnih jedinki i njihovo nadomještanje novima. Nepoželjne se određuju svojom dobrotom koja je preniska u odnosu na ostale. Moguće je odrediti fiksnu granicu dobrote ispod koje se eliminiraju sve jedinke, relativnu granicu u odnosu na prosječnu dobrotu populacije ili samo slučajno odabrati jedinku iz skupine lošijih dobrota. Empirijski je pokazano da eliminacija svih najlošijih po nekom kriteriju ne daje dobre rezultate[2]. Preporučena selekcija je slučajni odabir nekoliko jedinki, čak i ako nisu među najlošijima. Među njima se pronađe najlošija, koja ne mora ujedno biti i najlošija u populaciji, i eliminira. Time se zadržava mogućnost da lošije rješenje nekom mutacijom ili u kombinaciji križanja postane jako dobro rješenje.

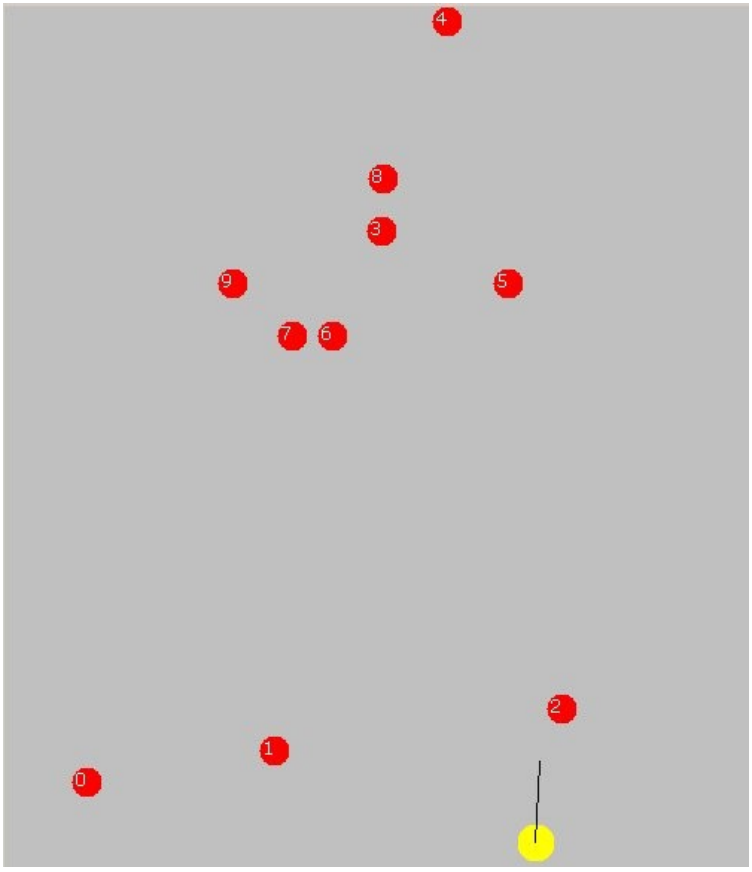
Nakon što je selekcija obavljena, odstranjene jedinke se nadomještaju novima, stvorene križanjem ili novim slučajnim generiranjem. Time završava iteracija genetskog algoritma i počinje nova.

Zaustavljanje algoritma definirano je kroz više uvjeta ovisno o problemu:

- ograničiti broj generacija koje će evoluirati,
- broj evaluacija funkcije cilja,
- postignuta je vrijednost funkcije cilja u određenim granicama točnosti,
- prosječna dobrota svih jedinki se nije značajno izmijenila u zadnjih nekoliko generacija,
- vremensko ograničenje je isteklo.

1.1. Primjena genetskog algoritma na automatsko upravljanje

Model na kojem se primjenjuje genetski algoritam u ovom radu je koncipiran na ideji biciklista koji izbjegava pješake u pješačkoj zoni. Biciklist se nalazi u dnu ekrana i procjenjuje smjer i brzinu kretanja tako da ne udari pješake.



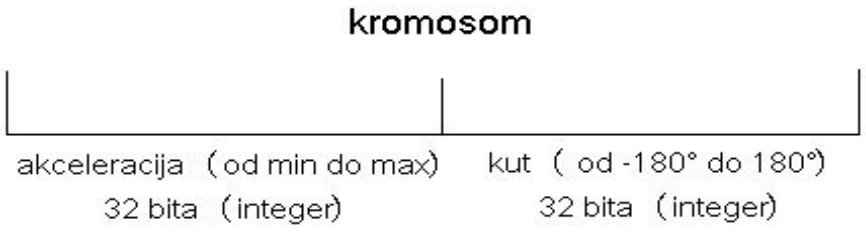
Slika 1-3 Prikaz pješačke zone, žuti biciklist i crveni pješaci

Biciklist se kreće pješačkom zonom i pokušava kretati se što vertikalnije, dakle stići prema vrhu ekrana što brže moguće. Njegova brzina, kao i u realnom svijetu, je veća od brzine kretanja pješaka tako da se čini kao da su pješaci statične prepreke. Njegova brzina se pridodaje pozicijama pješaka i tako se čini da se biciklist samo kreće lijevo ili desno, a pješaci prolaze pokraj njega. Koordinate pješaka su pohranjene u matrici. Broj pješaka je unaprijed određen, a time i dimenzije matrice. Kada pješak prođe uz biciklista i dođe do kraja zone, prividno se generira novi pješak na vrhu zone. Ustvari se samo po matrici vrši provjera je li koja koordinata prešla izvan dimenzije zone te se onda vrijednost y- koordinate izmijeni na

nulu tj. početnu koordinatu vrha zone, a za x-koordinatu se generira slučajni broj. Tako se dobiva konstantni izvor novih prepreka za izbjegavanje.

Algoritam ima diskretni vremenski interval u kojemu treba izračunati optimalni pravac kretanja. Interval je unaprijed zadan prije pokretanja simulacije i može se mijenjati. Tijekom tog vremena se evoluiraju i ispituju rješenja na osnovi kojih se računa kut i akceleracija. Biciklist bira smjer kretanja kao kut od -180° do 180° i može ubrzavati ili usporavati ovisno o tome koja akcija ne rezultira sudarom s preprekom. Maksimalna akceleracija je unaprijed određena, kao i maksimalna brzina. Za potrebe testiranja te granice su bile pomaknute na veće brojeve, kako bi se proučilo ponašanje algoritma na terenu većem od onog prikazanog grafičkim sučeljem i s većim brzinama.

Jedinka algoritma je kromosom od dvije cjelobrojne vrijednosti, ukupne duljine od 64 bita.



Slika 1-4 Struktura kromosoma

Prva 32 bita odgovaraju varijabli akceleracije koja se preračunava od minimalne do maksimalne cjelobrojne vrijednosti u minimalnu i maksimalnu granicu akceleracije. Minimalna cjelobrojna vrijednost odgovara minimalnoj akceleraciji tj. maksimalnoj negativnoj akceleraciji. Granična vrijednost je unaprijed određena.

Druga 32 bita predstavljaju kut tj. smjer kretanja. Broj se dijeli s maksimalnom vrijednošću cjelobrojnog tipa kako bi se dobio broj između -1 i 1 koji se množi s 90. Tom broju se pridodaje 90 kako bi se dobio puni raspon od 0 do 180 stupnjeva. U kombinaciji s negativnom akceleracijom, biciklist ima raspon kretanja od 360 stupnjeva.

Praćenje brzine biciklista je sadržano u zasebnoj varijabli kojoj je određena maksimalna vrijednost. Akceleracija se zbraja s dosadašnjom brzinom i ako prijeđe granicu maksimalne brzine, skalira se na iznos koji odgovara granici i pridodaje u brzinu. Ako je na maksimalnoj brzini, tada se u evaluaciji promatra samo smjer kretanja. Akceleracija i kut se preračunavaju u koordinate koje se procjenjuju na način da ne vode u sudar s pješakom, tj. da li je na putu

između trenutne pozicije biciklista i predloženih koordinata kretanja pješaka. Mogućnost sudara se određuje metodom koja uspoređuje blizinu koordinata biciklista i cijele matrice pješaka. Ako je udaljenost manja od veličine biciklista, označen je sudar i simulacija se zaustavlja. Površina biciklista i pješaka je određena kako bi predstavljali prostor koji zauzimaju.

Prije pokretanja programa potrebno je unijeti parametre u predviđena polja: veličina populacije, broj pješaka, duljina diskretnog vremenskog intervala i postotak mutacije.

Pokretanjem programa stvaraju se dvije dretve: glavna koja prati sve objekte i vrši pomicanje i sporedna s genetskim algoritmom. Usklađene su međusobnim isključivanjem (engl. *mutex*) i rukovanjem (engl. *handshake*). Međusobno isključivanje se primjenjuje u glavnoj dretvi gdje proces zaključava varijable najboljih koordinata dok ih čita i računa s njima. Nakon što pomakne sve objekte za određene koordinate i provjeri je li došlo do sudara, odgađa se za diskretni vremenski interval. Također postavlja zastavicu da je preuzeo i iskoristio koordinate, što dretva genetskog algoritma registrira i započinje novu evoluciju sa promijenjenim pozicijama pješaka i biciklista. Time počinje traženje rješenja za novi problem najboljeg vektora kretanja jer su izmijenjene međusobne pozicije objekata. Nakon što dretva genetskog algoritma generira i odredi dobrotu nove populacije, postavlja zastavicu da su koordinate slobodne za preuzimanje, za slučaj da glavni program traži najbolje rješenje već nakon prve generacije. Uz minimalni diskretni interval od 1 ms algoritam uspijeva izračunati barem nekoliko generacija prema testovima tako da je malo vjerojatno da je rješenje iz 'neevoluirane' populacije.

1.1.1. Generiranje početne populacije

Ovisno o unesenom broju veličine populacije n, stvara se matrica 3 x n. Iako je kromosom zamišljen kao 64-bitni niz, ipak se u izvedbi ostvaruje kao dva 32-bitna niza. Prvo mjesto u matrici je akceleracija, drugo kut, a treće sadrži dobrotu jedinke. Matrica se puni slučajno generiranim brojevima u rasponu od minimalne do maksimalne vrijednosti cjelobrojnog tipa. Generiranje osnovne populacije obavlja se zaključavanjem varijabli populacije i najbolje jedinke u prvoj generaciji dok nije cijela inicijalizirana. Time se glavni program štiti od preuzimanja starijeg rješenja koje nije odgovarajuće trenutnom stanju. Nakon generirane početne populacije i njene evaluacije funkcijom dobrote, postavlja se zastavica koja

obilježava rješenje slobodno za preuzimanje. U narednim generacijama nema međusobnog isključivanja, jer su rješenja već prilagođena trenutnom problemu, a dodatni posao zaključavanja objekata (engl. *overhead*) degradira efikasnost programa. Kada glavna dretva preuzima najbolje rješenje, sama zaključava varijable i postavlja zastavicu kojom označava da su preuzete. Dretva algoritma ispituje postavljanje zastavice te, u slučaju da je postavljena, generira novu početnu populaciju.

1.1.2. Određivanje dobrote svakog kromosoma

Funkcija dobrote procjenjuje svaki kromosom na način da mu pridodaje iznos koji predstavlja podobnost rješenja. Svaka jedinka sadrži akceleraciju i kut tj. promjenu brzine i smjer kretanja u prostoru. Cilj algoritma je navigirati biciklista kroz pješake, stoga rješenje koje ne rezultira sudarom treba biti nagrađeno, a pravac koji vodi u sudar treba biti sankcioniran. Kao i kod svakog kretanja, bolji put je onaj koji brže vodi do cilja. Zato je više nagrađen onaj smjer koji bez sudara ima veću promjenu po y-osi tj. prema vrhu ekrana. Preračunavanje akceleracije i smjera kretanja u predložene koordinate se izvodi preko slijedećih formula:

akceleracija = prva polovina kromosoma / maxIntValue * max akceleracija

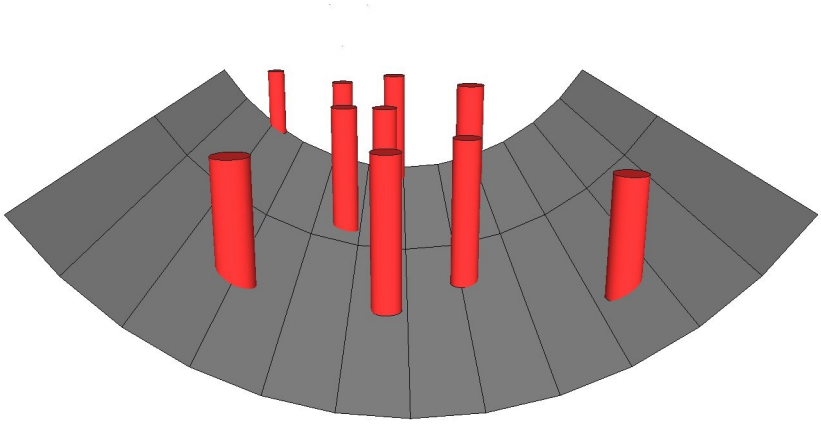
kut = ((druga polovina kromosoma /maxIntValue) * 90 + 90) * pi / 180

predlozeni x = (akceleracija + brzina biciklista) * cos(kut)

predlozeni y = (akceleracija + brzina biciklista) * sin(kut)

Varijabla maxIntValue sadrži najveću vrijednost cjelobrojnog podatka. Vrijednosti sadržane u kromosomu se djele s tom varijablom kako bi se dobio razlomak koji je pomnožen s maksimalnom vrijednošću koju varjabla može poprimiti. Tako se dobiva vrijednost iz raspona od minimalne do maksimalne vrijednosti te varijable. Primjerice za definirano maksimalno ubrzanje, rješenje može sadržavati bilo koju vrijednost od negativnog do pozitivnog maksimalnog ubrzanja. Kut je na sličan način definiran da poprima vrijednosti od 0° do 180°. Zbog implementacije ugrađenih funkcija sinus i kosinus koje koriste radijane pri računanju, kut je množen sa pi/180 radi preračunavanja u radijane. Izračunavaju se predložene koordinate preko transformacije iz polarnog u Kartezijev sustav uzimajući u obzir i dosadašnju brzinu biciklista iz prethodnih izračuna. Provjera puta od trenutne pozicije biciklista do predloženih koordinata obavlja se izračunom vektora u segmentima. Koordinate se uvećavaju za segmente dok ne dosegnu vrijednost predloženih koordinata. Segmenti ovise

o veličini pješaka jer je bitno uzeti u obzir da početna pozicija biciklista nije u sudaru, a nije ni završna predložena. Moguće je da je pješak negdje između početne i predložene i biciklist jednostavno 'prođe' kroz njega. Zato se pri provjeri trenutnim koordinatama biciklista dodaju postupno segmenti vektora i provjerava je li došlo do sudara u bilo kojem koraku. U slučaju sudara, kromosomu se pridjeljuje dvostruka negativna akceleracija, vrijednost koju je nemoguće drugačije poprimiti. U slučaju da nema sudara, jedinka biva evaluirana na osnovi pomaka prema pretopstavljenom kraju zone.



Slika 1-5 Funkcija dobrote pješačke zone

Na taj način se dobivaju pozitivno ocijenjene jedinke, koje pomiču biciklista naprijed i negativno ocijenjene, koje ga sudaraju. Ugrađena je i eksponencijalna funkcija koja postupno kažnjava preveliko odstupanje od sredine zone.

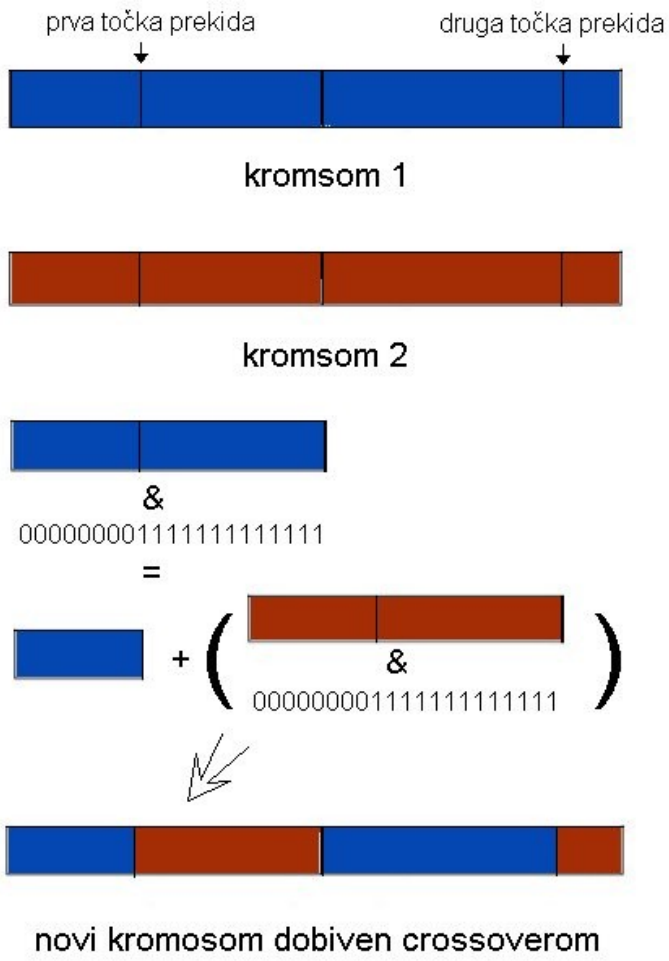
$$Penal = e^{\frac{x\_biciklista - sirina\_ulice/2}{0.8 * sirina\_ulice}} - 1$$

Kazna za približavanje lijevom ili desnom rubu zone je gotovo ravan penalu sudara. Konstanta od 0.8 je određena empirijski na osnovu izračuna kazne. Na slici 1-5 je prikazana prilagođena funkcija dobrote zone kako bi je vidio biciklist. Intuitivno bi težio sredini zone i izbjegavao crvene cilindre. Sa stvarnim iznosima gdje je sudar negativna vrijednost, a pozitivna prolaz, prikaz bi bio okrenut obrnuto. Svi faktori se zbrajaju u varijablu 'dobrota', koja se potom vraća u algoritam i pridružuje na treće mjesto u matrici populacije. Time je olakšano navigiranje kroz jedinke i odstranjivanje u daljnjim koracima algoritma. Nakon procjene dobrote, u posebnu varijablu je zabilježena najbolja do sada evoluirana dobrota, te se uspoređuje sa svim dobrotama u populaciji. Ako postoji viša vrijednost, ta se upisuje kao

najbolja nađena. Najbolje nađene predložene koordinate su pohranjene u zasebne varijable koje glavni program koristi i preuzima.

1.1.3. Selekcija jedinki

Nakon evaluacije populacije potrebno je odvojiti one jedinke koje su nedovoljno dobre za 'preživljavanje'. Jedinke ispod određene razine dobrote bivaju zamjenjene križanjem nekih od preostalih jedinki. Granična dobrota je određena kao polovina deceleracije, ali je dodana i vjerojatnost slučajnog odabira za selekciju, kako se ne bi deterministički odstranile sve lošije jedinke, već tek slučajno određen postotak. Vjerojatnost selekcije je usporediva sa varijablom PAR u harmonijskom pretraživanju. Nakon što je odabran kromosom koji treba zamijeniti, odabiru se dva slučajna kromosoma koji će križanjem stvoriti novi. Križanje je izvedeno u dvije točke prekida, jer je kromosom dvodimenzionalan.



Slika 1-6 Izvedba križanja, prvi dio kromosoma se maskom oduzima od samog sebe dok je pohranjen u drugu varijablu kako bi se dobio dio prije prekida. Tada zbrajanjem nastaje prvi dio novog kromosoma. Isti proces se ponavlja za drugi dio novog kromosoma, ali s drugom točkom prekida.

Odabrani kromosomi mogu biti i neki koji će u istoj generaciji biti odstranjeni, ali su bili slučajno odabrani prije nego su došli na red za selekciju, tako križati i stvoriti nove jedinke. Opisani postupak je implementacija eliminacijskog odabira s elitizmom.

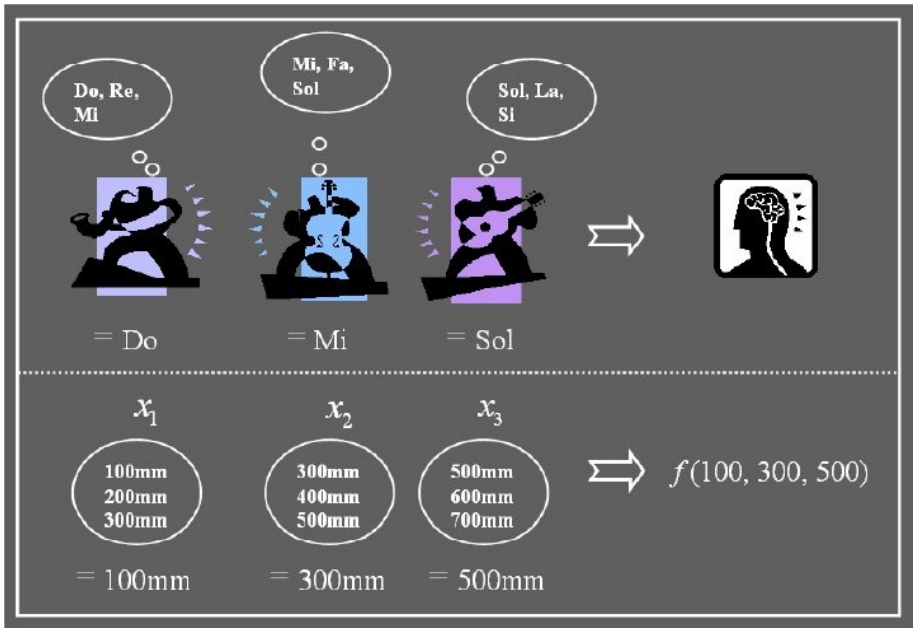
Mutacija je promjena jednog bita uz neku vjerojatnost. Vjerojatnost je zadana u polju na sučelju programa i istražen je njezin utjecaj na dobrotu rješenja u poglavlju 4. Generira se slučajni broj između 0 i 63 i tada je generirana maska sa jedinicom na tom mjestu. U slučaju da je broj veći od 31, oduzima se 32 od broja i generira maska cjelobrojnog tipa. Izvodi se logička operacija 'isključivo-ili' nad maskom i kromosomom. Za svaki kromosom računa se vjerojatnost odabira pomnožena s brojem bitova. Prolazi se cijelom populacijom i ako je

odabrana određena jedinka određuje se bit koji će mutirati. Tako nisu samo novogenerirane jedinke izložene mutaciji već i preostale u populaciji.

Nakon obavljene selekcije, križanja i mutacije, populacija se ponovo evaluira i kreće nova generacija evolucije algoritma. Kada istekne diskretni vremenski interval, glavni program preuzima najbolja nađena rješenja i mijenja stanje zone, koordinate pješaka i biciklista.

2. Harmonijsko pretraživanje

Harmonijsko pretraživanje je novi pristup metaheurističkom optimiranju. Zasniva se na ideji improvizacije melodije glazbenika, specifično jazz svirača, koji osluškuju jedni druge i ukomponiraju vlastite note da cjelina zvuči uhu ugodno. Osnovnu ideju je predložio Zong Woo Geem 2001. godine, ali prvi radovi su objavljeni tek 2005. godine. Do danas je objavljeno svega stotinjak radova, a prve knjige su izdane 2009. godine. Neki radovi pokazuju da je harmonijsko pretraživanje brže, u smislu da u manje iteracija dolazi do optimalnog rješenja, nego genetski algoritam [5]. Drugi radovi tvrde da harmonijsko pretraživanje ima problem prerane konvergencije i zna zapinjati u lokalnim optimumima [7]. Predložena su poboljšanja u izvedbi, a u ovom radu će implementacija obuhvatiti neka poboljšanja i kasnije biti uspoređena s već opisanom izvedbom genetskog algoritma.



Slika 2-7 Improvizacija glazbenika i harmonijski vektori

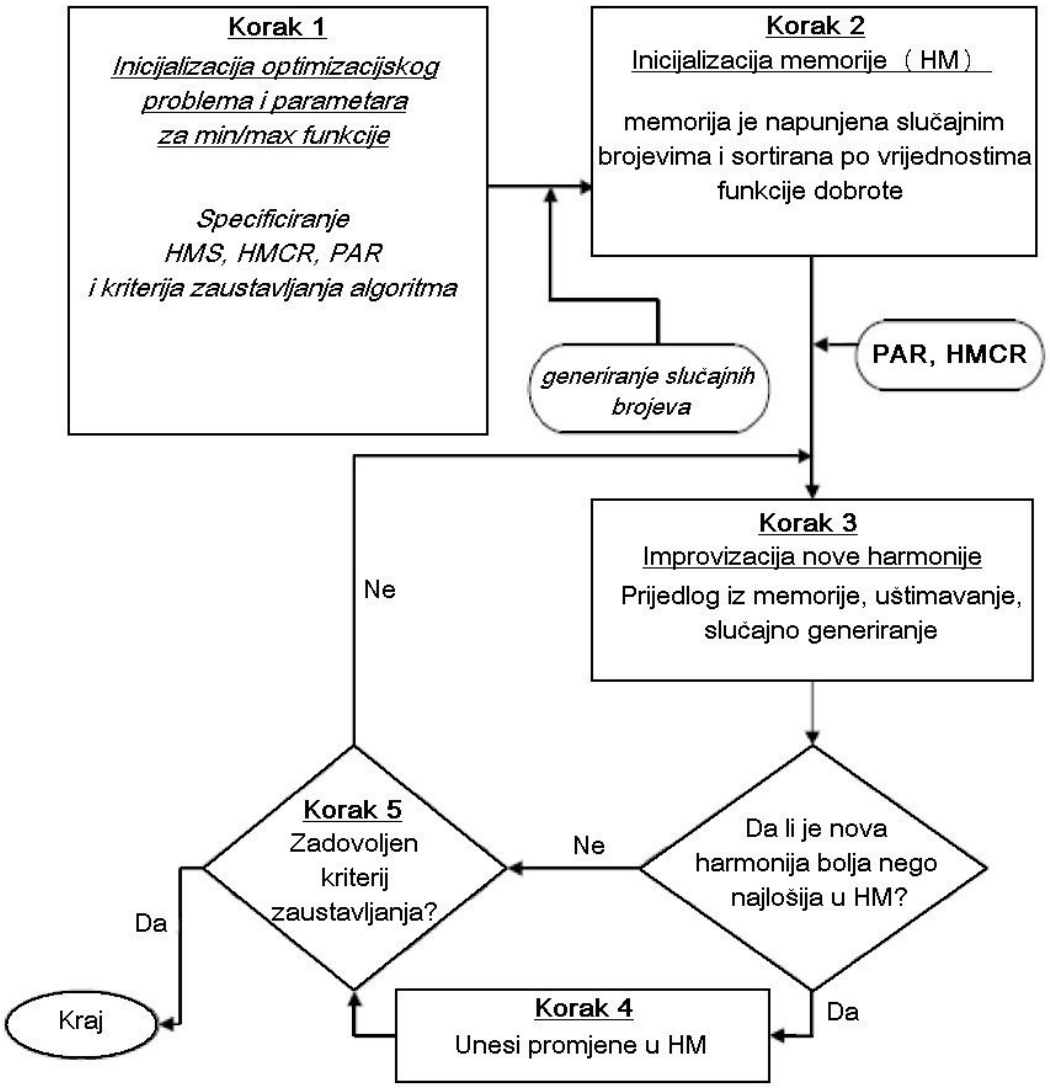
Harmonijsko pretraživanje je algoritam oponašanja fenomena, za razliku od algoritama koji oponašaju neku prirodnu pojavu. Fenomen se odnosi na pojavu osjećaja kod glazbenika u zajedničkom stvaranju harmonije ili skupa zvukova koji slušatelju i samim izvođačima zvuči dobro. Upravo ta procjena 'dobroga' je zapravo funkcija dobrote u algoritmu. Glazbenik generira notu svojim insturmentom. Glazbenik predstavlja izvor jednog vektora, a nota

rješenje. Muzička grupa je zapravo skup vektora koji zajedno pronalaze harmoniju, analogno populaciji koja evoluira u genetskom algoritmu.

Ključni parametri algoritma su:

- Veličina harmonijske memorije (*HMS – Harmony memory size*)
- Vjerojatnost odabira iz memorije (*HMCR – Harmony memory consideration rate*)
- Vjerojatnost 'uštimavanja' note (*PAR - pitch ajustment rate*)
- Broj improvizacija (*NI - number of improvizations*)

Broj glazbenika je predstavljen vrijednošću parametra HMS i predstavlja istovremeni broj vektora u memoriji. Glazbenici u samom algoritmu nisu specifična varijabla, već samo predstavljaju virtualni izvor pojedinog vektora i improvizacija. Note ili akordi, ako je rješenje višedimenzionalno, se nalaze u harmonijskoj memoriji (engl. *Harmony Memory*) kao vektori. Nakon inicijalizacije vektora slučajnim odabirom iz cijele domene rješenja, napunjen je HM. Tada se generira novi 'prazni' vektor **x** koji je nova improvizacija glazbenika. Taj vektor se puni ili vrijednostima iz HM ili generirajući nove vrijednosti. Tako se modelira odluka glazbenika da odsvira nešto što su već odsvirali drugi ili da improvizira novu notu. Ta vjerojatnost je iskazana HMCR varijablom. Ako glazbenik odluči odsvirati nešto iz memorije, generira se slučajni broj kako bi se odredilo od kojeg glazbenika će preuzeti notu. Tako se uzima komponenta vektora i gdje je i = rand(0,1) * HMS. Postupak se ponavlja za svaku komponentu. Ako je komponenta preuzeta iz memorije, tada se još prilagođava sa vjerojatnošću PAR. Prilagođavanje je snižavanje ili povećavanje note, što odgovara dodavanju ili snižavanju vrijednosti za određen iznos toj komponenti vektora **x**. U ovom radu je korišteno diferencijalno harmonijsko pretraživanje, gdje je razlika od osnovnog harmonijskog pretraživanja upravo u tom koraku 'uštimavanja'. Umjesto promjene za određen iznos, odabrani vektor mijenja se za razliku neke dve slučajno odabrane komponente vektora pomnožene s određenim koeficijentom. U slučaju da se nota tj. komponenta vektora improvizira, što se događa s vjerojatnošću 1-HMCR, vrijednost je tada slučajno generirani broj iz domene rješenja. Broj improvizacija je usporediv s brojem generacija genetskog algoritma. U ovom radu nije korišten broj improvizacija, jer je izvođenje omeđeno diskretnim vremenskim intervalom umjesto brojem izračunavanja algoritma.



Slika 2-8 Algoritam harmonijskog pretraživanja

2.1. Primjena harmonijskog pretraživanja na automatsko upravljanje

Automatsko upravljanje je ostvareno algoritmom harmonijskog pretraživanja na sličan način kao i genetskim algoritmom. Algoritam traži najbolje koordinate pomaka, vanjske varijable i dretve su iste, a razlika je u pokretanju dretve algoritma harmonijskog pretraživanja umjesto dretve genetskog algoritma.

Harmonijska memorija je matrica 3 x veličina harmonijske memorije (HMS), analogno populaciji genetskog algoritma. Veličina HMS-a je uzeta kao i kod genetskog algoritma od 20 do 500 jedinki te je ispitan utjecaj na kvalitetu rješenja. Harmonije, istovjetne jedinkama u genetskom algoritmu, su jednako podijeljene u dva 32-bitna cjelobrojna tipa uz još jedno 32-bitno mjesto za dobrotu rješenja. Prva varijabla je akceleracija, druga kut kretanja. Parametar preuzimanja iz memorije (HMCR) je varijabilan u iznosima od 0.2- 0.9. tj. 20% do 90% vjerojatnosti odabira iz memorije. Odabir iz memorije uzima jednu nasumičnu vrijednost iz stupca akceleracija i jednu nasumičnu iz stupca kuteva. U slučaju izabiranja iz memorije primjenjuje se diferencijalni operator mutacije umjesto fiksnog mjenjanja (PAR) s vjerojatnostima od 0.1 do 0.5. Fiksno 'uštimavanje' bi uvećalo ili umanjilo vrijednost za frakciju iznosa ukupnog raspona vrijednosti. Diferencijalni operator mutacije slučajno odabire dvije vrijednosti iz stupca HM-a i oduzme ih. Time se dobiva pozitivni ili negativni pomak, ovisno o odabranim vrijednostima.

Zaustavljanje algoritma je definirano istovjetno kao i za genetski algoritam. Nakon što istekne diskretni vremenski interval, generira se nova populacija slučajnih rješenja i proces improvizacije započinje ispočetka. Promjenjene su pozicije biciklista i pješaka i novo pokretanje algoritma radi s tim podacima. Sudar zaustavlja sve procese.

Diferencijalni operator mutacije

Izračunavanje varijance populacije iz generacije u generaciju prema [7] pokazuje da diferencijalni operator mutacije daje bolje rezultate brže od standardnog harmonijskog pretraživanja. Standardni algoritam koristi fiksne pomake u jednu ili drugu stranu, dok diferencijalni ovisi o slučajnom odabiru vektora iz populacije. Time se širi područje pretraživanja nedeterministički i brže nego u standardnom algoritmu.

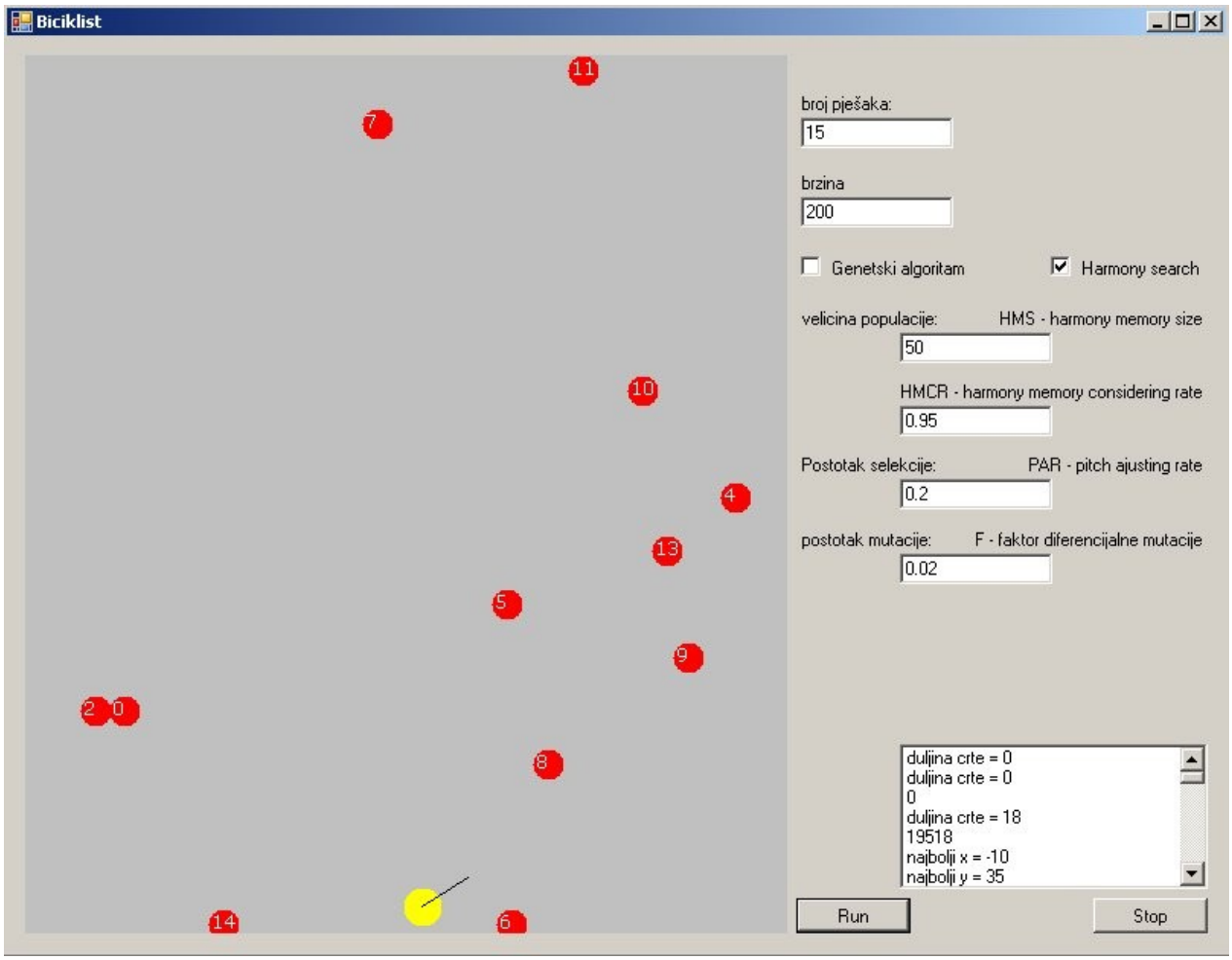
$$x_i = x_i + F(x_{r1} - x_{r2})$$

Odabranom vektoru *xi* se pridodaje razliku dva slučajno odabrana vektora pomnožena faktorom skaliranja F.

Faktor skaliranja F poprima vrijednosti između 0 i 1. Prema [7] dobar iznos varijable F je 0.8, ali je njegov iznos proširen u postupku ispitivanja. U slučaju prekoračenja maksimalne vrijednosti cjelobrojnog tipa (engl. *integer overflow*) C# jezik u ovoj implementaciji ne izvodi provjeru i ne izbacuje iznimku. Program se nastavlja izvoditi dok se odbacuju bitovi najvećih težina. Time je moguće dobiti vrijednosti u sasvim drugom dijelu domene rješenja i moguće je dobiti bolje rješenje. Funkcionalnost je slična operatoru mutacije u genetskom algoritmu, kao preventiva zapinjanju u lokalnim optimumima.

3. Grafičko sučelje programa

Grafičko sučelje za prikaz izvođenja i kontrole programa:



Slika 3-9 Grafičko sučelje

Sučelje je podijeljeno u prikaz izvođenja algoritama na lijevoj strani aplikacije i polja za unos parametara i kontrolu izvođenja programa na desnoj strani. Pješake predstavljaju crvene točke i numerirani su kako bi se pri izvođenju programa nadzirala korektnost izračunavanja. Žuti krug predstavlja biciklista, a crna crta je trenutni vektor kretanja, tj. najbolji nađeni smjer kroz algoritme.

Na desnom panelu nalaze se polja za broj pješaka i brzinu. Redom polja od vrha prema dnu imaju slijedeće funkcije:

Broj pješaka utječe na brzinu izvođenja programa prilikom izračunavanja funkcije dobrote, jer je matrica koordinata pješaka s kojom je potrebno usporediti blizinu koordinata biciklista veća.

Brzina je pauza u milisekundama u kojoj algoritmi izračunavaju najbolje koordinate.

Nakon brzine su dva međusobno isključiva polja za označivanje (engl. *radio button*) kojima se bira algoritam za izračunavanje. Lijevo je genetski algoritam, desno harmonijski. U slučaju ne odabiranja ili odabiranja istovremeno oba, izvodi se genetski algoritam.

Polje za *veličinu populacije* određuje broj jedinki nad kojom se vrši evolucija odnosno improvizacija. Ima utjecaj na brzinu izvođenja, jer je potrebno više jedinki evaluirati funkcijom dobrote i više puta ispitati uvjete operatora zbog toga što se vjerojatnosti provjeravaju neovisno za svaku jedinku.

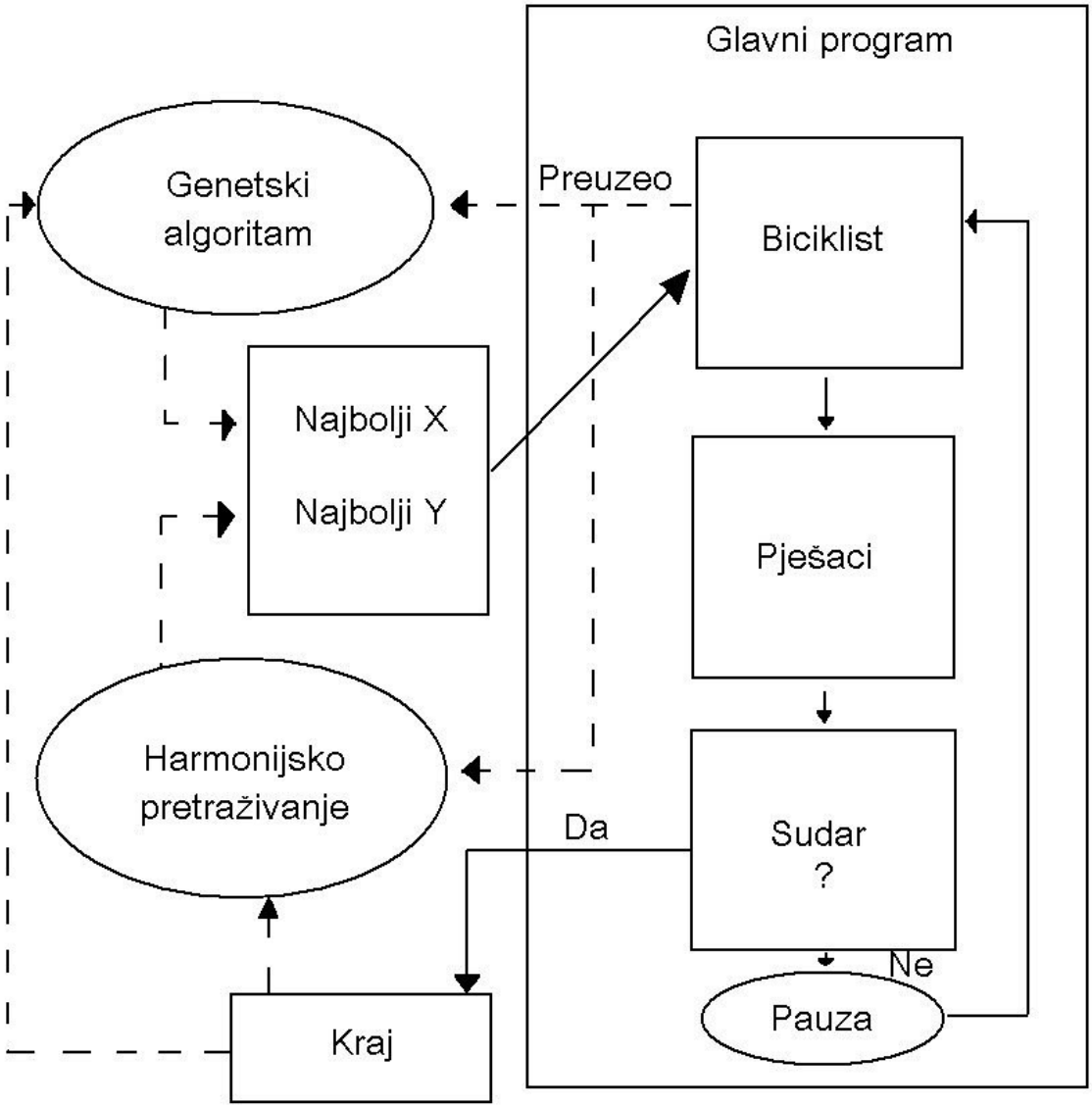
Daljnji parametri zavise o kontekstu odabira algoritma, jer se drugačije pristupa selekciji. Polje za odabiranje iz memorije i polje za korekciju varijabli imaju standardne vrijednosti od 0.2 do 0.9 i kao takve se koriste u programu. Genetski algoritam to polje koristi kao vjerojatnost selekcije ako jedinka ima prenisku dobrotu.

Kao dodatno polje za harmonijsko pretraživanje je i faktor diferencijalne mutacije. Taj parametar poprima realne vrijednosti od 0 do 1. Genetski algoritam ima vjerojatnost mutacije koja obavlja istovjetnu zadaću u tom algoritmu, ali vjerojatnost mutacije jedinke ne bi trebala prelaziti 0.015 (tj 1.5%), jer bi u tom slučaju svaka jedinka u populaciji imala vjerojatnost mutacije zbog duljine od 64 bita.

Ispod polja parametara je polje ispisa u kojem se pojavljuju vrijednosti i kratke obavijesti o izvođenju programa, a primarna zadaća je praćenje toka programa.

Na dnu su dvije funkcionalne tipke 'Run' i 'Stop' odnosno u prijevodu 'Pokreni' i 'Zaustavi'.

'Run' pokreće simulaciju i dretvu odabranog algoritma, dok 'Stop' zaustavlja sve dretve.



Slika 3-10 Prikaz programa i toka podataka

4. Rezultati izvođenja i odnos odabira parametara na kvalitetu rješenja i brzinu izračunavanja

U ovom poglavlju ispitani su utjecaji varijacije parametara na dobrotu rješenja i potreban broj generacija za evoluciju dovoljno dobrog rješenja. Ispitivanje domene je provedeno automatskim unosima parametara koji su obrađeni grafički.

Parametri koji su istraživani su:

- broj pješaka *n* (1 do 100)
- diskretni vremenski interval *t* (1 ms do 1000 ms)
- veličina populacije oba algoritma pop. (10 do 500)
- vjerojatnost odabira iz memorije (HMCR) za harmonijsko pretraživanje (0 do 1)
- vjerojatnost selekcije za genetski algoritam sel. (0 do 1)
- vjerojatnost uštimavanja (PAR) harmonijskog pretraživanja (0 do 1)
- vjerojatnost mutacije za genetski algoritam (0 do 0.015)
- težina faktora diferencijalne mutacije (F) za harmonijsko pretraživanje (0 do 1)
- relativna stabilizacija prosječne dobrote rješenja nakon određenog broja generacija kako bi se dobio uvid u brzinu konvergencije ka dovoljno dobrim rješenjima

Pri ispitivanju parametara mijenjanjem jednog, ostali su bili fiksirani na interval 50ms, populacija 50, HMCR 0.95, mutacija 0.005, par/selekcija 0.3, F 0.6, 20 pješaka, maksimalna akceleracija 30. Za izračunavanje prosječnih vrijednosti provedeno je 20 mjerenja i pokretanja programa i izračunata aritmetička sredina rezultata.

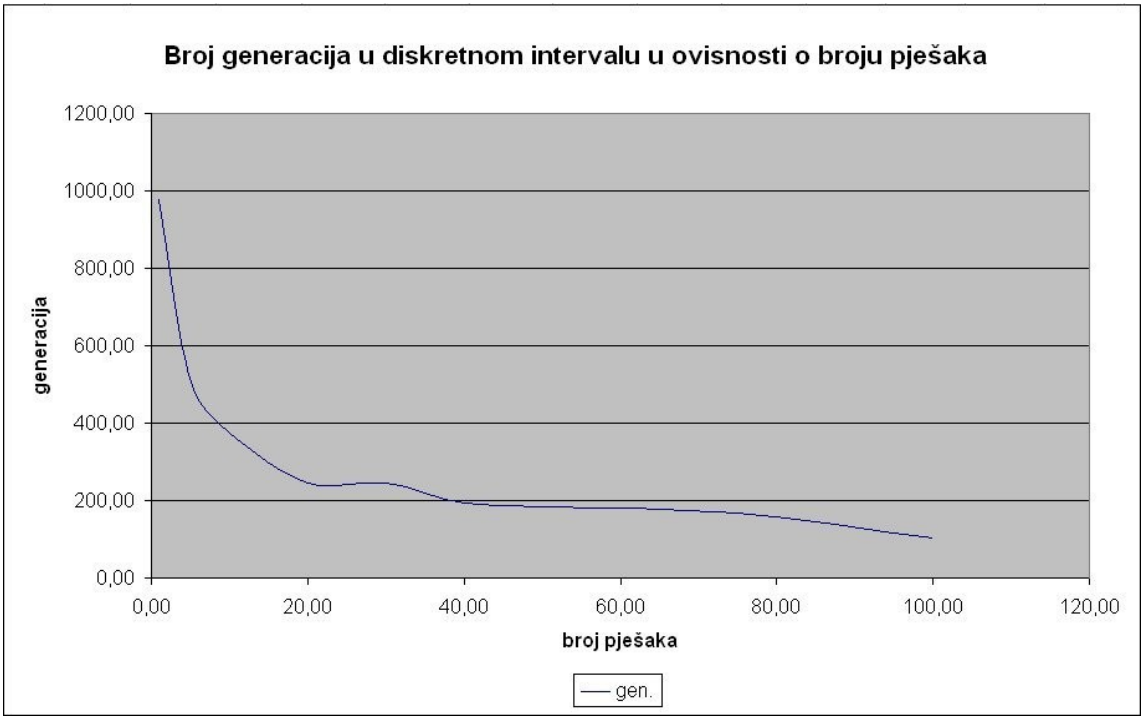
4.1. Utjecaj broja pješaka na broj generacija

U tablici 4-1 prikazano je opadanje broja generacija s povećanjem broja pješaka u konstantnom diskretnom vremenskom intervalu. Generacija je definirana kao generiranje *n* novih jedinki gdje je *n* veličina populacije. Jednako je definirana i u harmonijskom pretraživanju. Za uzimanje prosjeka po pješaku mjerenja su ponovljena 20 puta.

Tablica 4-1 Utjecaj broja pješaka na brzinu izvođenja genetskog algoritma

n	1	5	10	20	30	40	50	75	100
gen.	977,7	517,0	373,1	246,7	243,7	194,1	182,3	166,9	102,1

Povećanje broja pješaka očekivano usporava izvođenje algoritma. Pri izračunavanju funkcije dobrote algoritam iterira kroz matricu koordinata pješaka. Funkcija dobrote uspoređuje blizinu biciklista sa svakim pješakom u matrici kroz korak provjere. Potrebno je, ovisno o maksimalnoj akceleraciji i veličini pješaka, proći matricnom barem par puta. Mjerenje je obavljeno s diskretnim intervalom od 50ms. Uz činjenicu da je oko 200 generacija potrebno za postizanje dovoljno dobrog rješenja [9], prema slici 4-1 program može podnijeti oko 50 pješaka i u razumno kratkom vremenu evoluirati dovoljan broj rješenja.

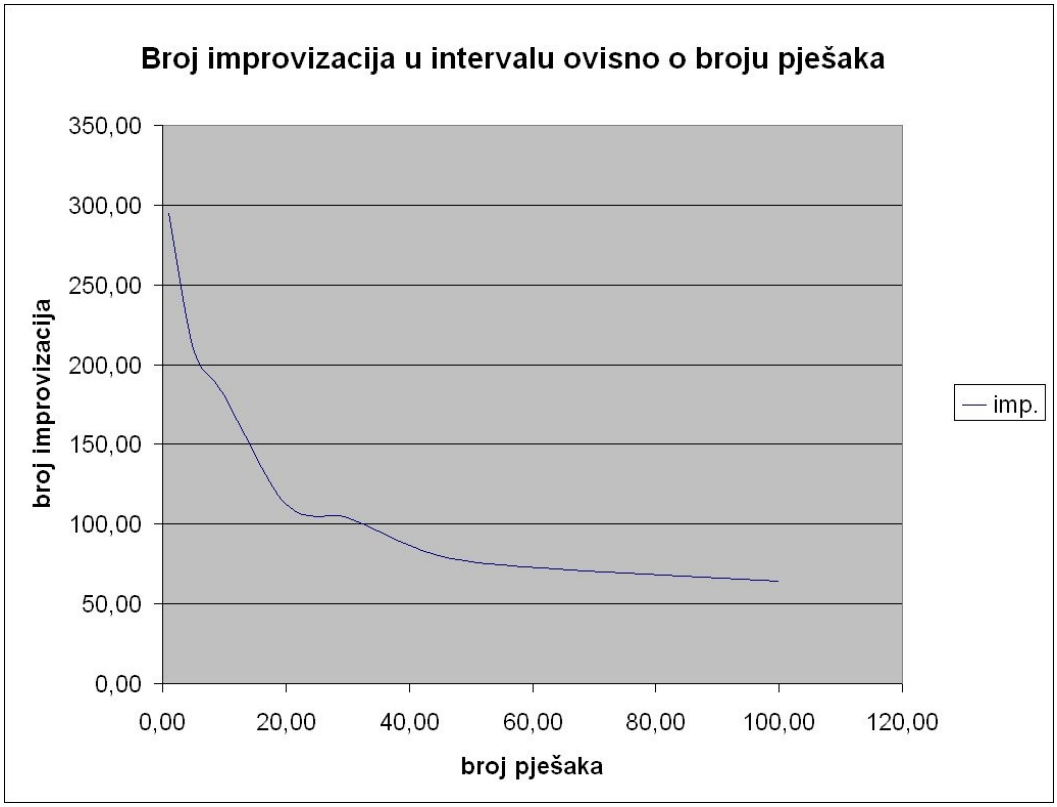


Slika 4-11 Broj generacija u genetskom algoritmu

Tablica 4-2 Utjecaj broja pješaka na brzinu u harmonijskom pretraživanju

n	1	5	10	20	30	40	50	75	100
imp.	295,8	210,9	181,8	112,3	104,4	86,2	76,2	68,7	64,7

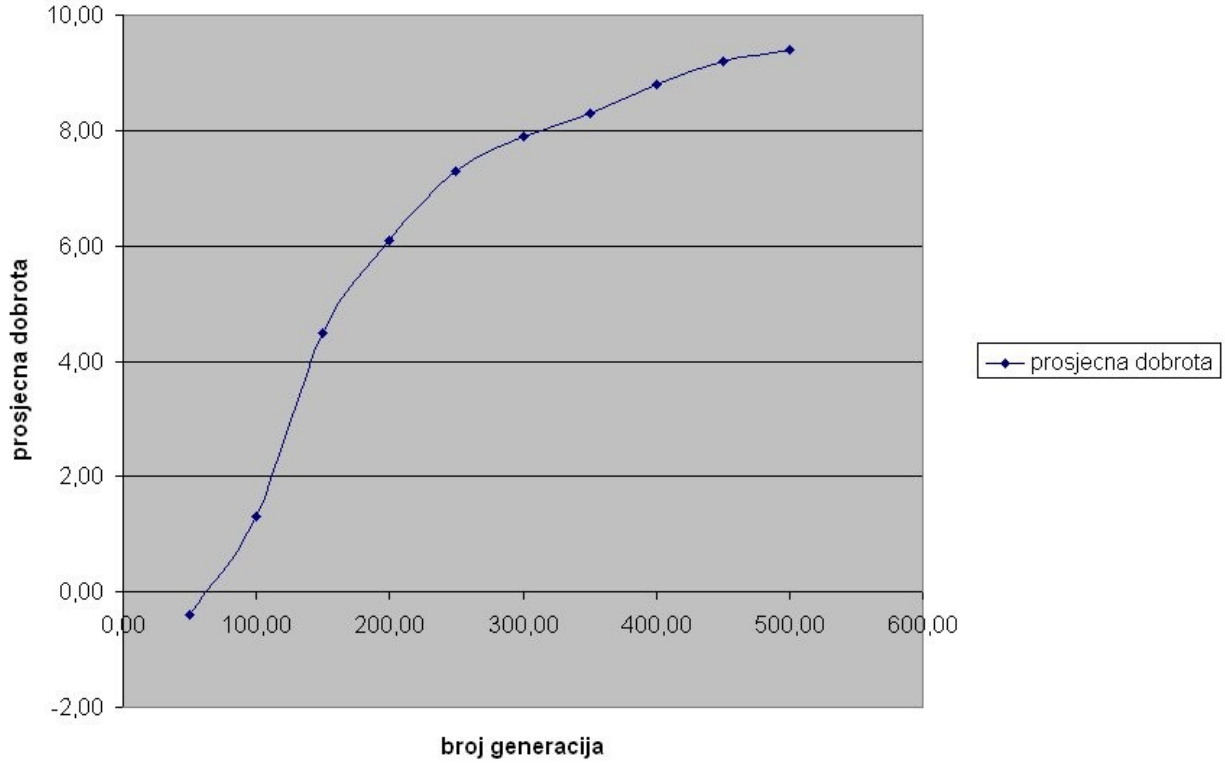
Harmonijsko pretraživanje je u testovima pokazalo oko 3 puta manju brzinu, nego genetski algoritam u izračunavanju generacija. Razlog tome je u implementaciji algoritma, gdje se u jednoj improvizaciji dodaje točno jedno novo rješenje, ali potrebno je obaviti *n* dodavanja novih rješenja kako bi se to smatralo improvizacijom usporedivom sa generacijom genetskog algoritma. Genetski algoritam samom mutacijom neke jedinke generira novu, dok je kod harmonijskog pretraživanja proces složeniji. Korišten je interval od 50ms, a broj pješaka inverzno proporcionalno smanjuje broj improvizacija po intervalu. Prema slici 4-2 vidljivo je da se zadovoljavajućih dvije stotine improvizacija dobiva pri manjem broju pješaka nego kod genetskog algoritma.



Slika 4-12 Opadanje broja improvizacija zbog povećanja broja pješaka

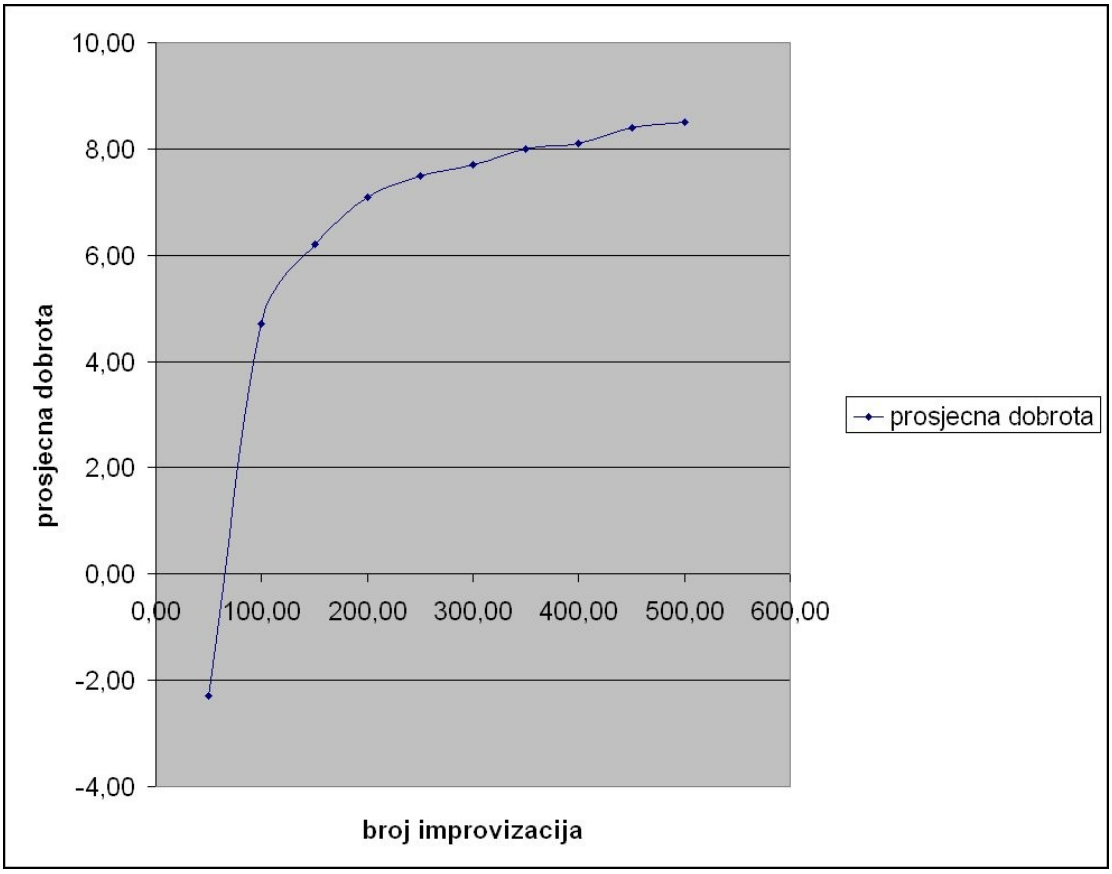
4.2. Kretanje prosječne dobrote u odnosu na broj generacija

Prosječna dobrota cijele populacije raste eksponencijalno, ali nakon određenog broja generacija rast prosječnog poboljšanja se smanjuje. Nakon 200 generacija dobrota je već dovoljno visoka da pruži kvalitetno rješenje i daljnje generacije tek blago povećavaju prosječnu dobrotu. Zbog elitizma najbolje nađeno rješenje je sačuvano tako da čak i ako populacija konvergira na neko lošije, najbolje rješenje svih generacija nije izgubljeno.



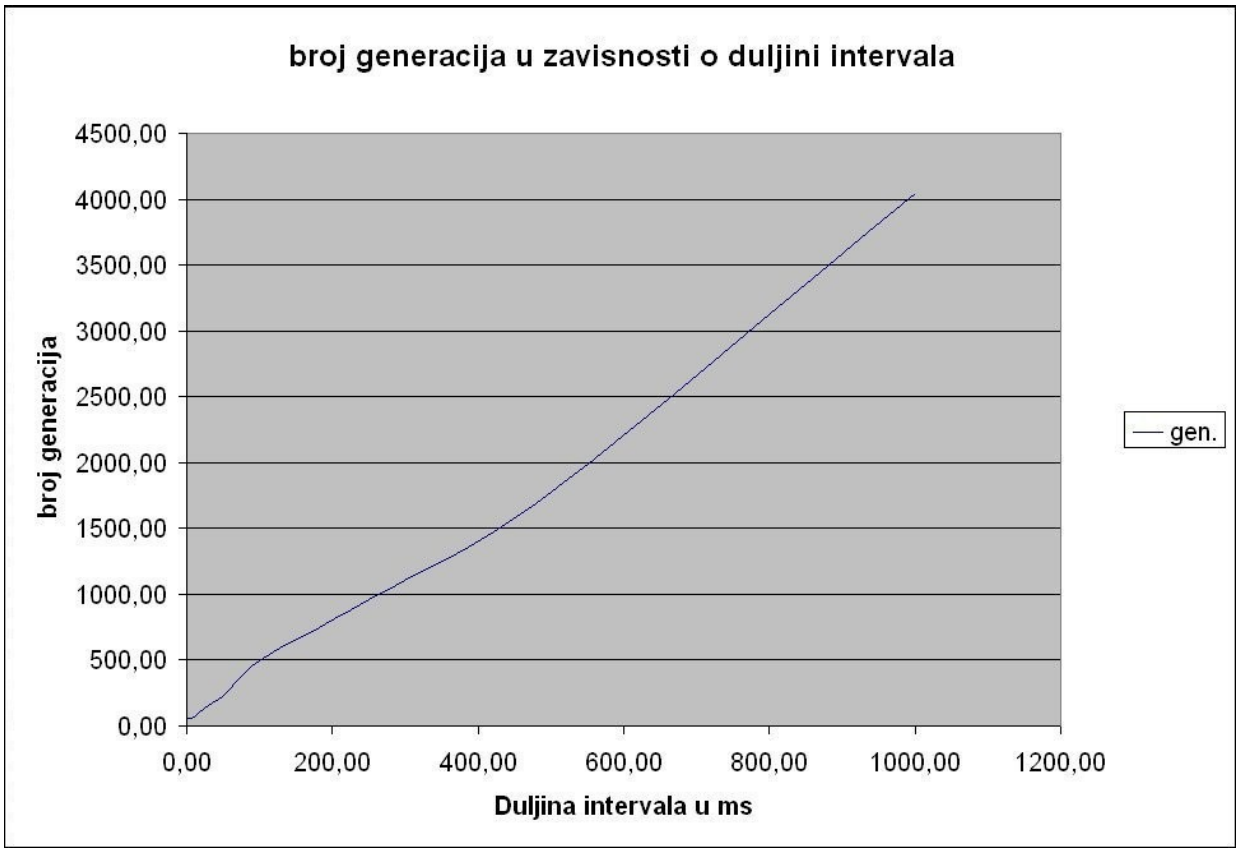
Slika 4-13 Prosječna dobrota populacije u genetskom algoritmu

Harmonijsko pretraživanje pristupa improvizaciji zamjenjivanjem samo jedne jedinke. Stoga je prošireno testiranje na način da se generiranje *n* jedinki smatra jednom improvizacijom algoritma. Tako su algoritmi usporedivi na osnovi količine generiranja novih jedinki. Na slici 4-4 vidljivo je da se relativna stabilnost rješenja dobiva nakon približno dvije stotine improvizacija. Kao i kod genetskog algoritma implementiran je elitizam tako da najbolja jedinka nije izgubljena.



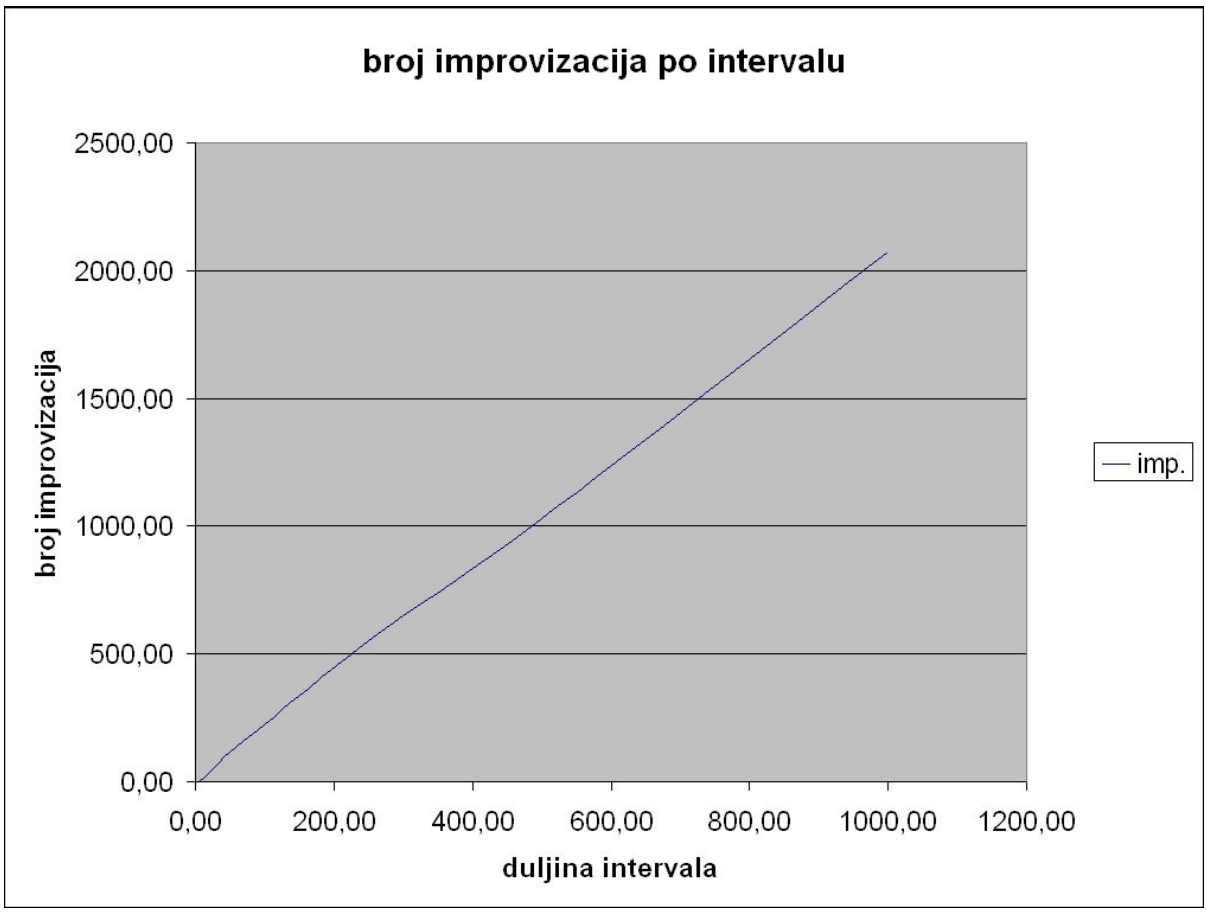
Slika 4-14 Kretanje prosječne dobrote populacije ovisno o broju improvizacija

4.3. Utjecaj diskretnog vremenskog intervala na broj generiranih novih jedinki



Slika 4-15 Porast broja generacija u genetskom algoritmu s duljinom diskretnog intervala

Povećanje duljine diskretnog intervala linearno povećava broj generacija izračunatih u intervalu. Na slikama 4-5 i 4-6 prikazano je kako istovjetno utječe i na harmonijsko pretraživanje. Prikaz je očekivan pri većim duljinama intervala, jer je odnos uzajamnog zaključavanja varijabli i duljine intervala sve manji. Pri vrlo kratkim intervalima do 10 ms se zamjećuje nelinearnost zbog 'kućanskih poslova' zaključavanja varijabli pri preuzimanju najboljih rješenja od dretvi algoritama. Harmonijsko pretraživanje pokazuje otprilike dva puta sporije generiranje jednakog broja novih jedinki u istim diskretnim intervalima nego genetski algoritam.



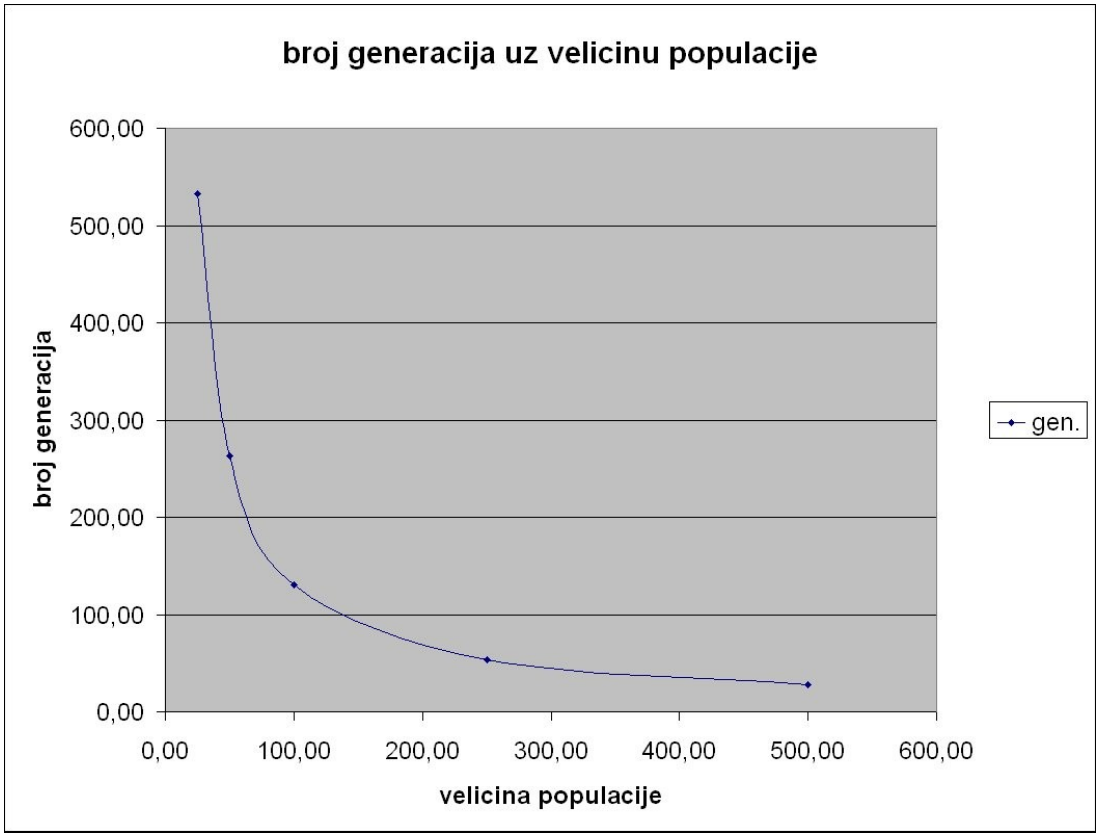
Slika 4-16 Linearno povećanje broja improvizacija sa duljinom diskretnog intervala

4.4. Utjecaj veličine populacije na broj izračunatih generacija i improvizacija

Uvidom u tablicu 4-5 i sliku 4-7 može se zaključiti da povećanje veličine populacije eksponencionalno usporava izvođenje genetskog algoritma. Uzrok leži u višestrukom prolaženju populacijom kroz jednu generaciju algoritma. Prvi put se selektiraju jedinke preniske dobrote i drugi put kada se primjeni vjerojatnost mutacije pojedine jedinke.

Tablica 4-3 Utjecaj veličine populacije na broj generacija genetskog algoritma

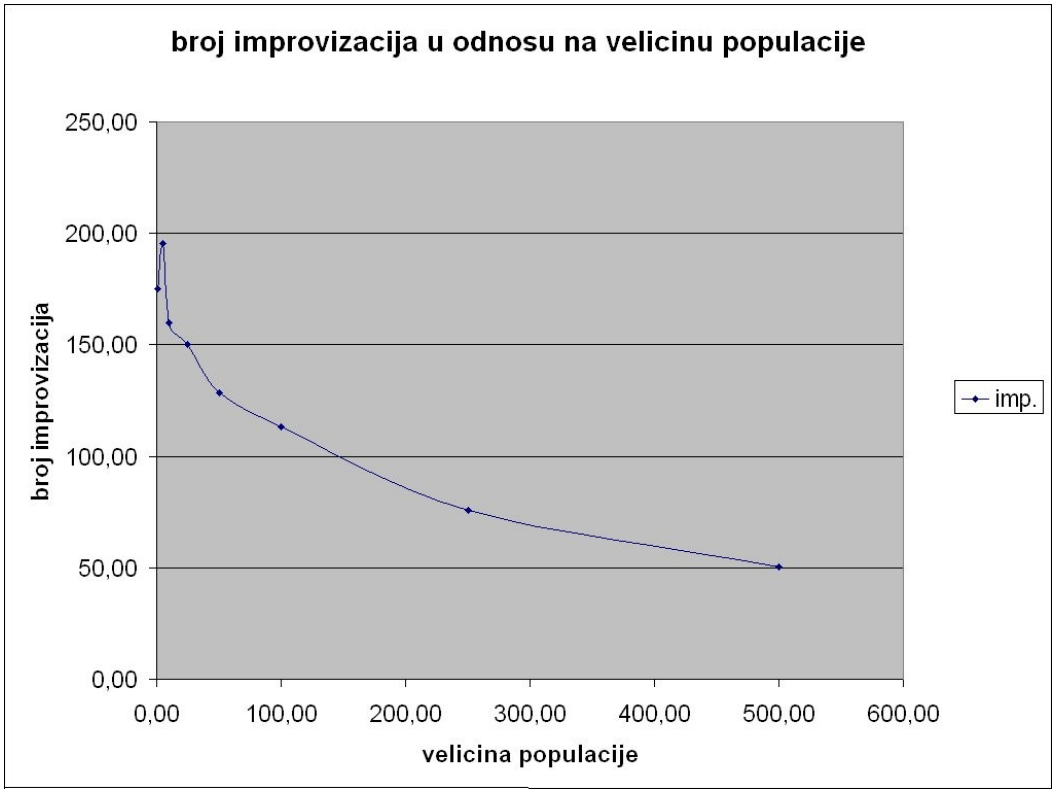
pop.	5	10	25	50	100	250	500
gen.	2776,1	1457,4	532,4	262,8	130,9	54,0	28,3



Slika 4-17 Opadanje broja izračunatih generacija sa veličinom populacije

Tablica 4-4 Utjecaj veličine populacije na broj improvizacija harmonijskog pretraživanja

pop.	1	5	10	25	50	100	250	500
imp.	11940,2	14651,2	12038,9	11607,3	10393,2	9281,5	5685,4	3598,4



Slika 4-18 Broj improvizacija u odnosu na veličinu populacije tj. harmonijske memorije (HMS)

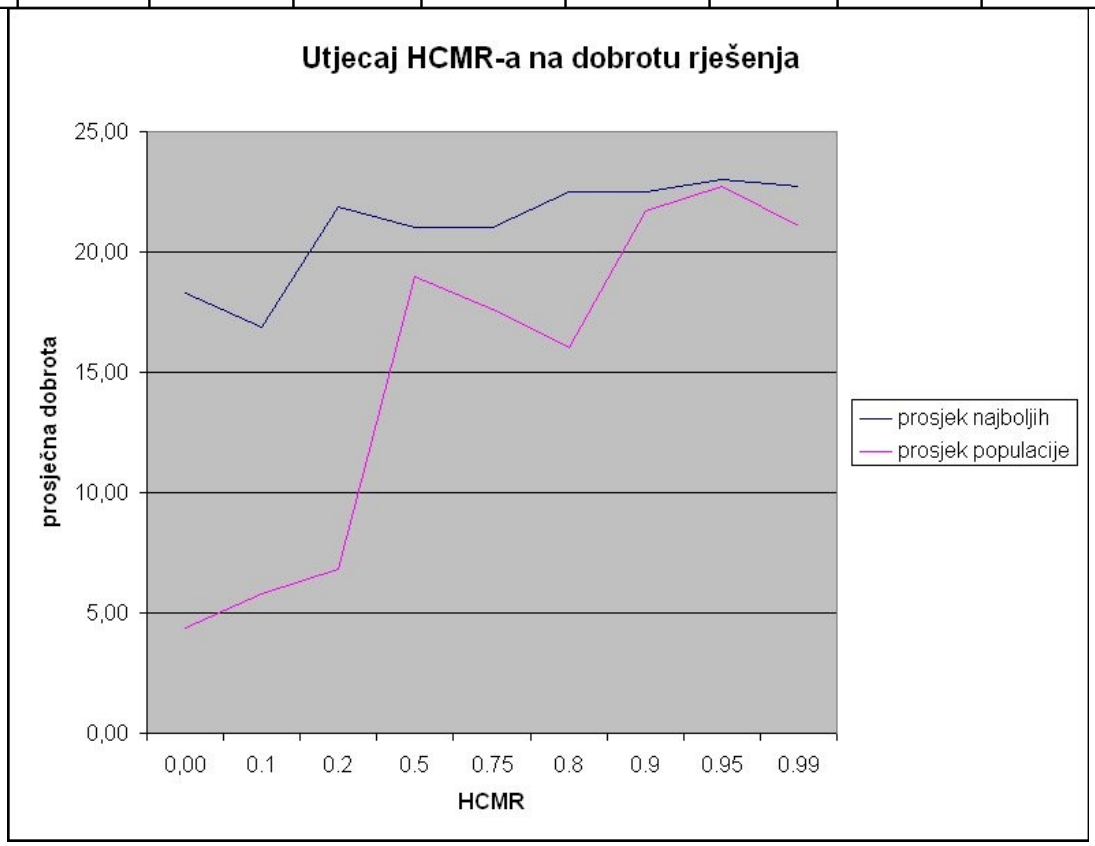
Istraživanje prostora parametara i usporedbe među algoritmima pokazuje da harmonijsko pretraživanje u jednoj iteraciji dodaje manji broj jedinki nego genetski algoritam. Ustroj algoritma dozvoljava takve rezultate. Harmonijsko pretraživanje nadodaje novu jedinku i zamjenjuje najlošiju u populaciji. Time se dobiva gotovo linearni pad brzine ovisno o veličini populacije. To je ujedno i glavna prednost algoritma harmonijskog pretraživanja. Harmonijski algoritam zadržava dovoljan broj improvizacija s rastom populacije dok genetski algoritam počinje gubiti brzinu izvođenja radi previše prolazaka kroz populaciju. Genetski algoritam ipak generira više jedinki po jednoj iteraciji nego algoritam harmonijskog pretraživanja tako da algoritam harmonijskog pretraživanja može relativno sustići brojem improvizacija samo za velike vrijednosti populacija.

4.5. Utjecaj vjerojatnosti odabira iz memorije (HMCR) na kvalitetu rješenja

Tablica 4-7 prikazuje kretanje prosjeka najbolje dobrote jedinke tj. optimalnog rješenja kroz dvadeset mjerenja dok drugi redak prikazuje prosječnu dobrotu cijele populacije.

Tablica 4-5 Utjecaj parametra HMCR na dobrotu rješenja u harmonijskom pretraživanju

HMCR	0	0.1	0.2	0.5	0.75	0.8	0.9	0.95	0.99
po mjer.	18.3	16.9	21.19	21.0	21.0	22.5	22.5	23.0	22.7
min.	(4.4)	(5.8)	(6.8)	(19.0)	(17.6)	(16)	(21.7)	(22.7)	(21.1)



Slika 4-19 Prosječna i maksimalna dobrota populacije u ovisnosti o parametru HMCR

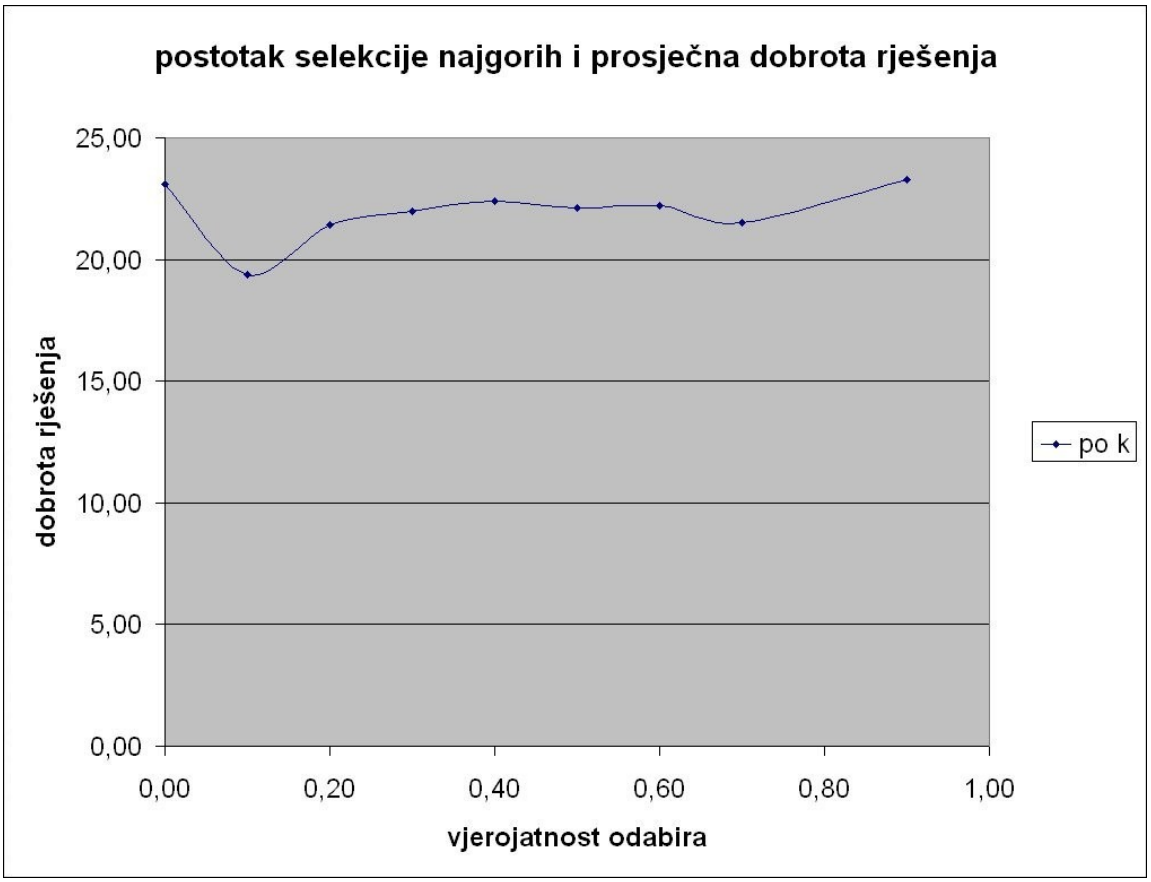
Konstantno generiranje novih rješenja s vjerojatnošću P = 1 –HMCR istražuje nove pravce kretanja. Pri malim vrijednostima parametra HMCR rješenja su češće bila lošija i sudari češći. Pri višim vrijednostima iznad 0.8 algoritam je pokazao mnogo bolje ponašanje što bi objasnilo stabilnost rješenja i u kombinaciji s parametrom PAR značajno bolje performanse.

Prema ranijim radovima [7], [9] pokazalo se točnim da češći izbor iz memorije, tj. viši HMCR, nego novo generiranje, bolje konvergira i iskorištava trenutnu populaciju.

4.6. Utjecaj vjerojatnosti prilagodbe note (PAR) i vjerojatnosti selekcije na kvalitetu rješenja

Tablica 4-6 vjerojatnost selekcije (sel) u genetskom algoritmu i utjecaj na prosječnu dobrotu rješenja

sel.	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9
fit.	23.1	19.3	21.4	22.0	22.4	22.1	22.2	21.5	23.4



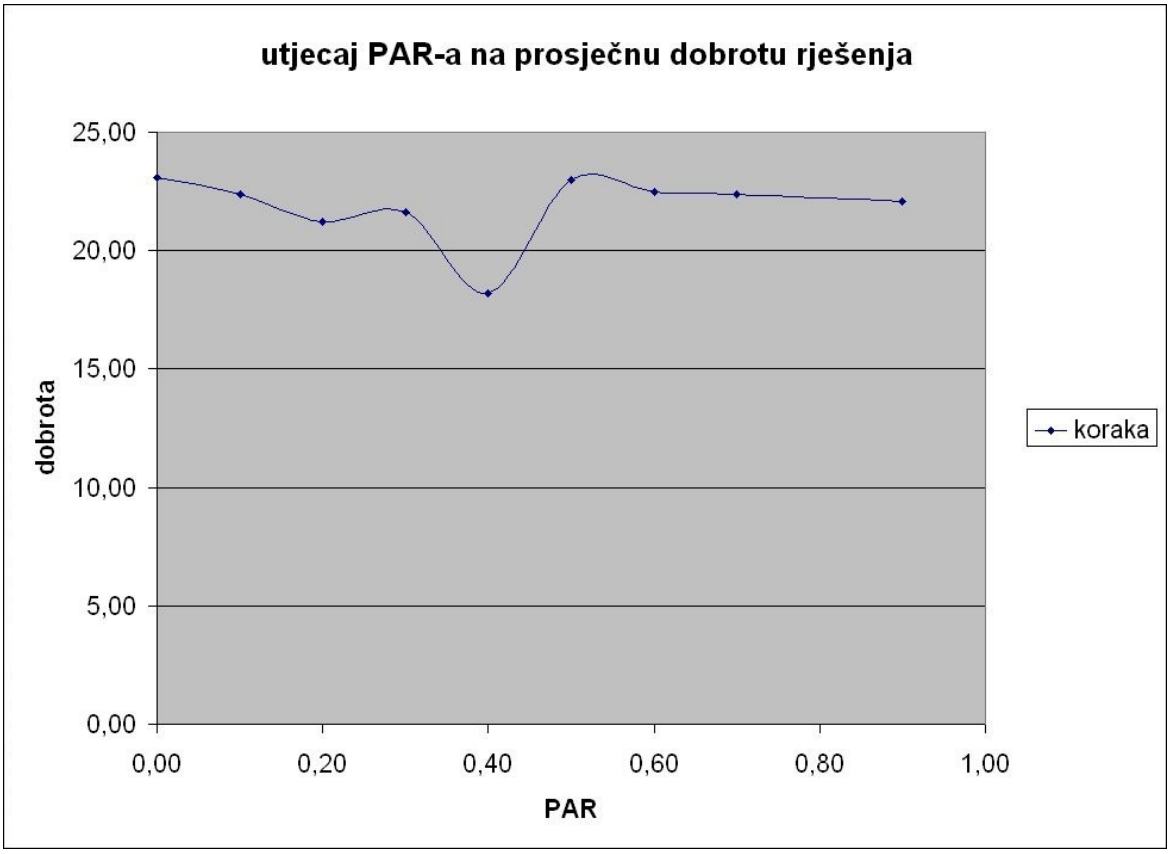
Slika 4-20 Vjerojatnost selekcije najgorih jedinki gentskog algoritma

Selekcija određuje postotak lošijeg djela populacije koji treba zamijeniti novim jedinkama. Odabir se bazira na dobroti jedinke i vjerojatnosti selekcije. Ako je slučajno generirani broj između 0 i 1 manji od vjerojatnosti odabira, provjerava se je li dobrota jedinke manja od pola maksimalne deceleracije. Vjerojatnost odabira analogna je varijabli PAR u harmonijskom pretraživanju. Tako se uz određene niže postotke selekcije zadržavaju lošije jedinke, koje bi

kasnije križanjem mogle doprinjeti evoluciji boljeg rješenja. Prema slici 4-10 najbolji iznos parametra je oko 0.5, dakle zadržati oko pola loše populacije.

Tablica 4-7 harmonijsko pretraživanje PAR

PAR	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9
fit	23.1	22.4	21.2	21.6	18.2	23.0	22.5	22.4	22.1



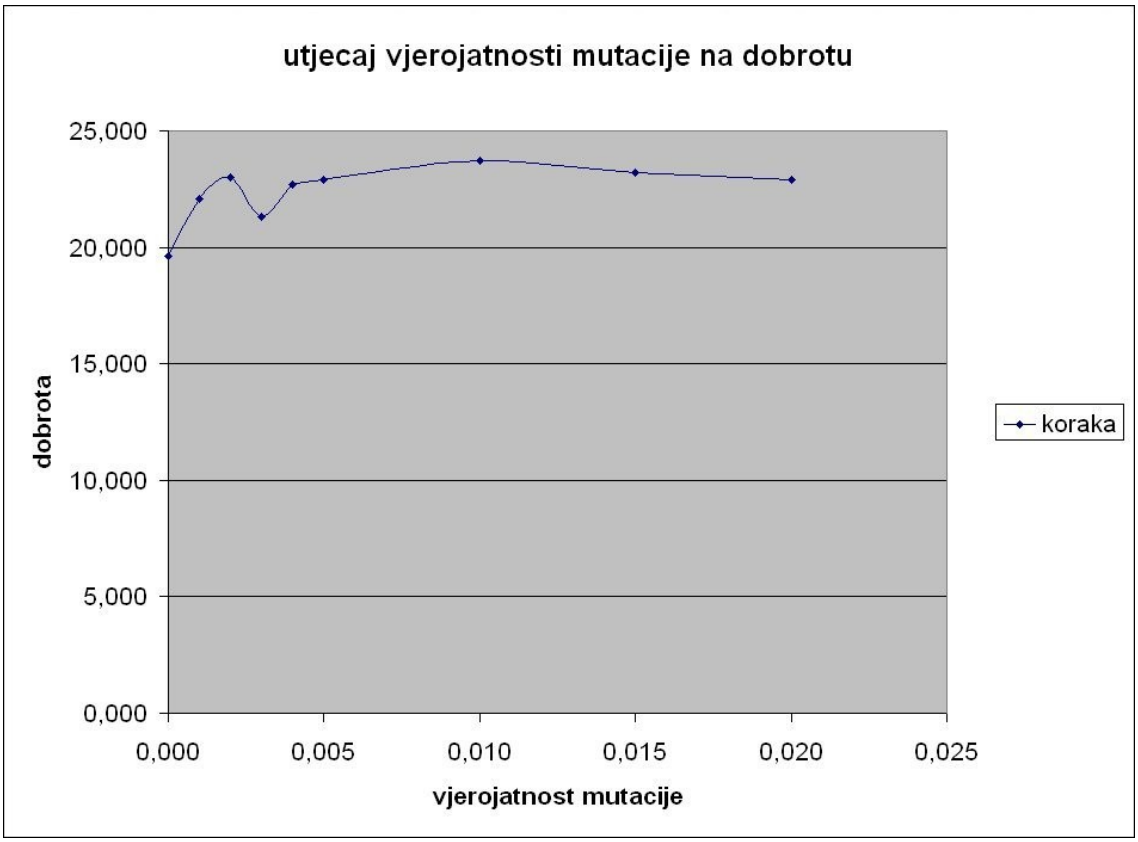
Slika 4-21 Vjerojatnost selekcije improvizacija za uštimavanje

Parametar uštimavanja (PAR) pokazuje na testovima bolju dobrotu pri ostavljanju selektiranih jedinki kao što su bile, za vjerojatnost 0 ili selekciji oko 50% jedinki za diferencijalnu mutaciju. Dobrota pozitivno odgovara na iskorištavanje postojeće populacije.

4.7. Utjecaj mutacije/diferencijalnog operatora na kvalitetu rješenja

Tablica 4-8 vjerojatnost mutacije i utjecaj

mut.	0	0.001	0.002	0.003	0.004	0.005	0.01	0.015	0.02
fit	19.6	22.1	23.0	21.3	22.7	22.9	23.6	23.2	29.9

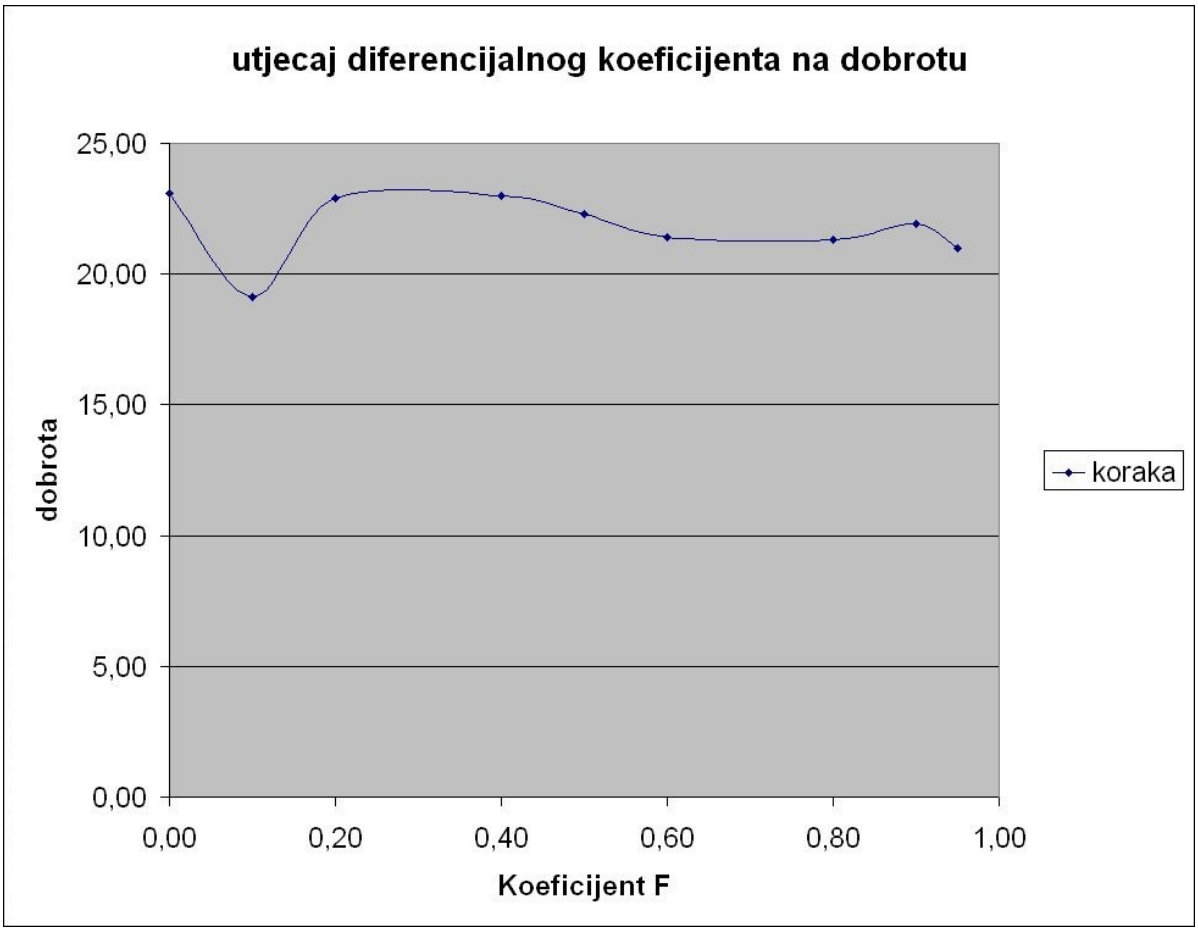


Slika 4-22 Vjerojatnost mutacije jedinki

Mutacija u genetskom algoritmu služi za iskorištavanje populacije i izbjegavanje lokalnih minimuma. Vjerojatnost mutacije se izračunava na osnovi mutacije jednog bita u kromosomu, a zbog fiksne duljine od 64 bita mutacije iznad 1.5% zapravo mutiraju svaku jedinku. To odgovara 0.015 na grafičkom prikazu. Primjetno je da je najbolji odabir oko 0.005 i 0.01 tj. mutirati oko trećinu populacije. Zbog elitizma najbolja jedinka ostaje sačuvana, ali mutacijom se svejedno mogu istraživati drugi predjeli domene.

Tablica 4-9 utjecaj diferecijalnog koeficijenta

F	0	0.1	0.2	0.4	0.5	0.6	0.8	0.9	0.95
fit.	23.1	19.1	22.9	23.0	22.3	21.4	21.3	21.9	21.0



Slika 4-23 Utjecaj diferencijalnog koeficijenta F na prosječnu dobrotu najboljih rješenja

Diferencijalni koeficijent prema [7] ima pozitivnije performanse pri vrijednostima od oko 0.8. U ovom radu je domena ciklička tj. cijelo područje cjelobrojnog tipa. Pri preljevu se jednostavno preskače na drugi kraj domene, i to može donijeti bolje rješenje, ali za sitnije korekcije potreban je manji faktor F kako se i vidi na slici 4-13. Manji faktor za manje pomake pomiče novu notu i time je veća mogućnost za postizanjem lokalnog boljeg rješenja. Zbog slučajnog odabira vektora pri izračunavanju diferencijala, moguće je i s većim faktorom dobiti manje pomake, a pri populacijama koje su prošle više generacija evolucije to je i vjerojatnije. Prema dobivenim rezultatima testiranja, dobri odabiri faktora su ili vrlo mali do 0.05 ili oko sredine spektra između 0.2 i 0.4.

Zaključak

Automatsko upravljanje pokazalo se izvedivim i mogućim uz primjenu evolucijskih algoritama. Algoritmi u vrlo kratkom vremenu dolaze do zadovoljavajućih rješenja. Genetski algoritam je pokazao solidna rješenja prema standardnoj veličini parametara populacije od oko dvjesto jedinki. Tražeći pritom evoluciju od dvije stotine generacija, algoritam postiže kvalitetno rješenje unutar 200 ms.

Harmonijsko pretraživanje zbog svoje izvedbe može podnjeti veće opterećenje populacije i prepreka, ali ne zadržava razumno vrijeme izvođenja. Problem je u prevelikoj osjetljivosti početnih parametara, koji se trebaju istražiti prije nego se primjene na zadani problem. Loš odabir parametara bi dovodio do prosječno lošijih rješenja. Uz dobre parametre, algoritam harmonijskog pretraživanja može postići konkurirajuća rješenja genetskom algoritmu. Harmonijsko pretraživanje se pokazalo potencijalno moćnim optimizacijskim algoritmom, iako ne dovoljno robustnim, niti brzim u odnosu na genetski algoritam. Vjerojatno bi bilo moguće bolje istražiti potencijal algoritma harmonijskog pretraživanja na mnogodimenzijalnim problemima s velikim populacijama, jer se pokazao otporniji na rast broja varijabli.

Algoritme bi bilo moguće primijeniti na upravljanje fizičkim vozilom, koje bi uz specijalizirane i paralelizirane procesore moglo vrlo brzo donositi dobre vektore izbjegavanja prepreka i reakcije na svoju okolinu. Uz odgovarajuću senzornu opremu i dovoljno dobro opisane vanjske faktore, algoritmi bi mogli donositi kvalitetne odluke u vremenskim intervalima od nekoliko stotina milisekundi. Uzme li se u obzir da je ljudska svjesna reakcija na vizualni podražaj između 300 i 500 milisekundi [10], algoritmi ne samo da mogu brzinom konkurirati ljudskom upravljanju, nego i donijeti optimalne odluke. Primjerice pri izbjegavanju automobila u prometu ispred vozača automatskom reakcijom kočenja, algoritam bi mogao naći bolji smjer bez smanjivanja brzine. Naravno, ključne teškoće se nalaze u računalnoj percepciji okoline.

Mnoštvo problema virtualnog upravljanja, koji su već matematički opisani u detalje, npr. programske igre ili dinamički zadaci protoka, burzovna kretanja, upravljanje robotima, bi mogli korisiti evolucijske algoritme i dobrom kalibracijom konkurirati ljudskim izvedbama.

Literatura

[1] Golub, M., Genetski algoritam prvi dio, skritpa, Fakultet elektrotehnike i računarstva, Zagreb, 2010.

[2] Jakobović, D., Genetski algoritmi – predavanje, skritpa, Fakultet elektrotehnike i računarstva, Zagreb, 2008.

[3] Genetic algorithm, http://en.wikipedia.org/wiki/Genetic_algorithm, 2010.

[4] Harmony search algorithm, http://en.wikipedia.org/wiki/Harmony_search, 2010.

[5] Harmony search, http://sites.google.com/a/hydroteq.com/www, 2010.

[6] Geem, Z.W, State-of-the-art in the structure of harmony search algorithm, John Hopkins University, Baltimore, 2009.

[7] Chakraborty et al., An improved harmony search algortihm with differential mutation operator, Jadavpur University, Kolkata, 2009.

[8] Geem, Z.W. et al, Application of harmony search to vehicle routing, John Hopkins University, Baltimore, 2005.

[9] Sarmady, S. An investigation on genetic algorithm parameters, Universiti Sains Malaysia, Penang, 2007.

[10] Nerve impulses, University of British columbia, http://c21.phas.ubc.ca/article/nerve-impulses, 2010.

Skraćenice

DNK		Deoksiribonukleinska kiselina
HMS	Harmony memory size	Veličina harmonijske memorije
HM	Harmony memory	Harmonijska memorija
HMCR	Harmony memory consideration rate	Vjerojatnost odabira iz memorije
PAR	Pitch ajustment rate	Vjerojatnost ‘uštimavanja’ note
NI	Number of improvizations	Broj improvizacija

Popis stranih izraza

fitness function	funkcija dobrote
generational selection	generacijski odabir
steady state selection	eliminacijski odabir
roulette wheel selection	generacijski jednostavni odabir
mutex	uzajamno isključivanje
handshake	rukovanje
overhead	dodatni posao
integer	cijelobrojni tip podataka
overflow	preljev
checkbox	polje za označivanje

Dodatak

Program sadrži polja za unos parametara i kontrolne gumbe za reguliranje izvođenja programa. Grafičko sučelje prati izvođenje programa i izračunavanje najboljeg vektora kretanja. Za detaljnu analizu svakog koraka odabranog algoritma generira se informativna datoteka 'analiza.txt' sa odabranim parametrima i praćenjem svih obrađivanih podataka. Datoteka sadrži redom popis početnih parametara i prosječne dobrote svake generacije, preuzete koordinate i provjeru događaja sudara. Datoteka se generira pri izvođenju programa i zbog operacija čitanja i pisanja usporava izvođenje programa, međutim usporava oba algoritma pa ostaju usporedivi. Ostane li program u izvođenju dovoljno dugo, datoteka može poprimiti veće dimenzije od par stotina megabyte-ova u par minuta.

Program je izvođen na računalu IBM Thinkpad T60 sa Intel Centrino Duo procesorom na 1.83GHz uz 1 GB RAM memorije. Izvođen je na Microsoft Windows XP operativnom sustavu.

Instalacija programske podrške

Program je sadržan u direktoriju Snalazljivi_biciklist, a za pokretanje je potrebno podržavati .NET okružje najstarije verzije 2005 . Nije potrebna dodatna instalacija.

Upute za korištenje programske podrške

Detaljne upute za korištenje su navedene u trećem poglavlju. Program se pokreće u .NET okružju otvaranjem projekta i kompajliranjem ili preko izvršne datoteke 'Snalazljivi_biciklist.exe' iz \Snalazljivi_biciklist\Snalazljivi_biciklist\bin\Debug direktorija u programskoj podršci. Tada se otvara upravljačko grafičko sučelje, upisuju parametri i pokreće program.