SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1358

Učinkovitost prikaza s pomičnom točkom u evolucijskim algoritmima

Matija Renić

Zagreb, lipanj 2010.

SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1358

Učinkovitost prikaza s pomičnom točkom u evolucijskim algoritmima

Matija Renić

Zagreb, lipanj 2010.

Hvala!

Svima u obitelji koji su mi omogućili ovaj rad, posebno roditeljima

Sneki i didu, za stipendiju

Marini, što je slušala moja gunđanja

Ostalima koji su me pratili i bili tu

Doc. Dr. Sc. Domagoju Jakoboviću na mentorstvu

Sadržaj

1. Uvod

Evolucijski algoritmi predstavljaju skup algoritama koji se služe imitacijom nekih od evolucijskih principa koji su prisutni u prirodi, u svrhu rješavanja određenih problema. Osnovna prednost ovih algoritama nad drugima je preslikavanje rješenja problema faktorijelne ili eksponencijalne složenosti u probleme polinomijalne složenosti. Algoritam ne jamči pronalazak optimalnog rješenja jer ne pretražuje cijelu domenu rješenja, ali su pronađena rješenja najčešće optimalna ili vrlo blizu optimalnih.

Genetski algoritmi su podskup evolucijskih algoritama [1]. Analogija evolucije kao prirodnog procesa i genetskog algoritma kao metode optimiranja, očituje se u procesu selekcije i genetskim operatorima. Mehanizam odabira nad nekom vrstom živih bića u evolucijskom procesu čine okolina i uvjeti u prirodi. U genetskim algoritmima ključ selekcije je funkcija cilja, koja na odgovarajući način predstavlja problem koji se rješava.

Definicija genetskog algoritma uključuje funkciju cilja, genetske operatore križanja i mutacije, prikaz i selekciju. Prikaz, ili evolucijskim riječnikom rečeno genotip, definira način na koji će jedinke populacije rješenja biti reprezentirane na računalu. Genetski operatori moraju biti prilagođeni prikazu nad kojim vrše operacije i stvaraju nove, valjane jedinke populacije.

Početak drugog poglavlja ovog rada opisuje prikaz rješenja u obliku niza brojeva s pomičnom točkom. Neki od postojećih operatora križanja nad navedenim prikazom su opisani u nastavku poglavlja. Praktični dio zadatka, ugrađivanje podrške za prikaz s pomičnom točkom u ECF, razrađen je u trećem, a eksperimentalna ocjena učinkovitosti ugrađenog prikaza u odnosu na binarni prikaz, u četvrtom poglavlju. Osim toga u četvrtom poglavlju provedeno je ispitivanje učinkovitosti dva implementirana operatora križanja. Kroz zaključak su komentirani rezultati dobiveni eksperimentom i postavljene smjernice za daljnje istraživanje i razvoj.

2. Prikaz s pomičnom točkom

Rješavanje nekih problema pomoću genetskih algoritama ponekad ne daje optimalni rezultat u željenoj preciznosti. Uz problem preciznosti neki od problema na koje nailaze evolucijski algoritmi su problem prerane konvergencije cijele populacije u neki lokalni optimum i nemogućnost finog podešavanja rješenja u trenutku kada se populacija približila globalnom optimumu.

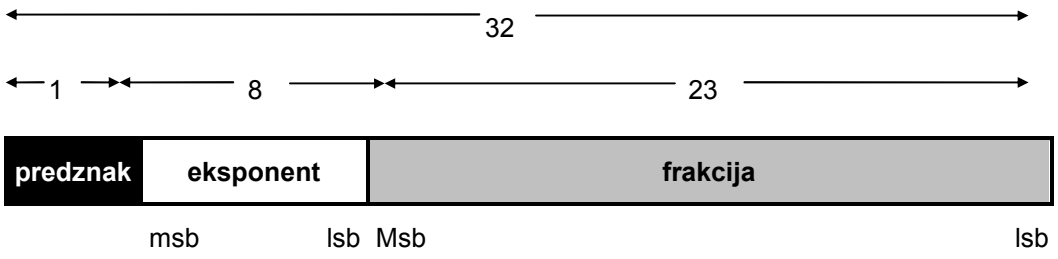
Binarni prikaz se koristi u najvećem dijelu istraživanja i implementacija genetskih algoritama. Osim toga ovaj način kodiranja olakšava teoretsku analizu i omogućava jednostavne i učinkovite genetske operatore. Usprkos tome što je najčešće korišteni prikaz rješenja, nailazi na prepreke kada se koristi za rješavanje višedimenzionalnih problema koji zahtjevaju veliku preciznost. Tako za problem sa sto varijabli unutar intervala [-500, 500] gdje je zahtjevana preciznost od šest decimala iza decimalne točke, vektor rješenja u binarnom prikazu sadrži 3000 bita. Skup svih mogućih rješenja za ovakav problem doseže 10^{1000} članova.

Sam rezultat genetskog algoritma i sve njegove prednosti nisu strogo uvjetovani prikazom, stoga nije loše razmisliti o upotrebi većih abeceda od binarne za prikaz kromosoma. Korištenjem samo drugačijeg prikaza, a istog genetskog algoritma, mogu se ublažiti ili eliminirati neke negativne pojave i što je najvažnije prikaz se približava domeni problema. Samim time prisiljeni smo i operatore bolje prilagoditi problemu tj. karakteristikama domene. Općenito prikaz rješenja treba čim vjernije opisivati domenu rješenja, a s druge strane biti pogodan za čim jednostavnije i učinkovitije manipulacije operatorima, naravno sve uz dobra rješenja kao rezultat.

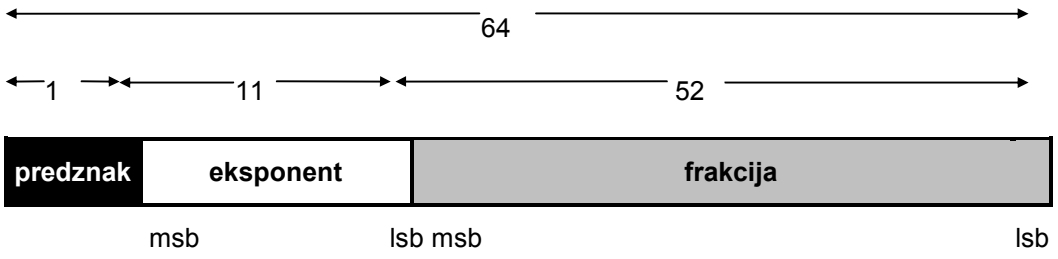
Binarni prikaz kromosoma je cijeli broj, zapisan kao niz nula i jedinica, koji predstavlja realni broj iz nekog intervala. Najčešće se koristi prirodni binarni kod, a ponekad i Grayev kod čija je karakteristika da se točke susjedne u domeni problema, nakon kodiranja, razlikuju u samo jednom bitu. Kod prirodnog binarnog kodiranja lako se može dogoditi da se dvije točke koje su susjedne u domeni rješenja, u domeni prikaza rješenja razlikuju za više od 2 bita. Ako iz domene rješenja uzmemo za primjer 127 i 128 čija je razlika jedan, a njihovi prikazi u binarnom obliku su 01111111 i 10000000, vidimo da je udaljenost između prikaza

puno više od dva. Razlikuju se u točno 9 bita. Na prvi pogled ta pojava ne djeluje kao da može značajno utjecati na izvođenje algoritma, ali kada se pogleda iz perspektive operatora mutacije može dovesti do nekih zanimljivih pojava. Neka se mutirana jedinka razlikuje od svoga originala samo u jednom bitu kojem se mijenja vrijednost. Vrlo lako na taj način možemo skočiti npr. s rješenja 0 na 128 što je prilično velik skok, a nije uvijek poželjna pojava.

Alternativni prikaz je prikaz sa pomičnom točkom čije korištenje ne povlači ugrađivanje posebnog mehanizma za dekodiranje kakav je prisutan kod binarnog prikaza jer i sam kromosom je realan broj zapisan prema normi IEEE 754-1985 (Slika 2.1, Slika 2.2). Osim toga održavanje populacije ne zahtjeva neke posebne napore za programera jer gotovo svi programski jezici današnjice podržavaju takav prikaz. Potrebno je jedino pažljivo prilagoditi ili osmisliti operatore križanja i mutacije.



Slika 2.1: Broj s pomičnom točkom prema normi IEEE 754-1985



Slika 2.2: Broj s pomičnom točkom dvostruke preciznosti prema normi IEEE 754-1985

2.1 Opis prikaza rješenja

S obzirom da se u ovom radu promatra učinkovitost prikaza pomičnom točkom na kontinuiranim optimizacijskim problemima, tj. pronalazi optimum višedimenzionalnih kontinuiranih funkcija, potrebno je prije svega i pobliže opisati odgovarajuću strukturu kromosoma. Svaki kromosom je vektor x (2.1) kojeg čine geni, brojevi s pomičnom točkom ili binarno kodirani brojevi ovisno o implementaciji:

$$x = < e_1, \dots, e_n > \tag{2.1}$$

gdje su $e_1 \dots e_n$ binarni brojevi, odnosno brojevi s pomičnom točkom, n je dimenzija funkcije, a samim time i vektora rješenja.

U nastavku će biti opisani neki genetski operatori specifični za ovakav tip problema i način prikaza.

2.2 Križanje

Križanje je binarni genetski operator čije je glavna zadaća prenošenje značajki jedinki kroz generacije. Križanjem dvaju roditelja stvaraju se nove jedinke djeca po uzoru na roditelje. Tako novi genetski materijal ulazi u populaciju.

$$s_v^t = < v_1, \dots, v_n > \tag{2.2}$$

$$s_w^t = < w_1, \dots, w_n > \tag{2.3}$$

Neka su s_v^t i s_w^t u jednadžbama (2.2) i (2.3) oznake za jedinke koje su odabrane za roditelje u mutaciji. s_c^t označava kromosom djeteta, oznake v, w i c su identifikatori jedinke, a t označava generaciju.

2.2.1 Križanje u intervalu

Prilikom provođenja jednostavnog *križanja u intervalu*, nastaje sasvim nova jedinka s_o^ξ (2.4) preuzimajući na neki način značajke roditelja. Nova jedinka zamjenjuje neku postojeću najčešće slučajno odabranu jedinku populacije. Svaki gen iz vektora kromosoma djeteta c_i je slučajna vrijednost iz intervala koji određuju geni odgovarajućih pozicija u kromosomima roditelja (2.5).

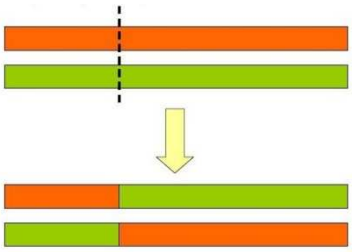
$$s_o^\xi = \ll c_1, \dots, c_n \gg \tag{2.4}$$

(2.1)

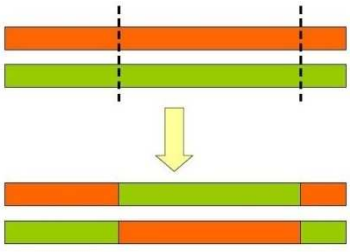
$$c_i = slučajna\ vrijednost\ iz\ intervala(v_i, w_i),\ \ i \in \{1..n\} \tag{2.5}$$

2.2.2 Jednostavno križanje

Operator *jednostavnog križanja* stvara novu jedinku koja se zamjenjuje najčešće slučajno odabranom jedinkom populacije. Razlikuje se križanje u jednoj, na Slika 2.3 ili više točaka na Slika 2.4.



Slika 2.3: križanje u jednoj točki



Slika 2.4: križanje u dvije točke

Kromosom je vektor koji se sastoji od n gena (2.1), a kako su oni brojevi u prikazu s pomičnom točkom, dopuštene pozicije točaka prekida su jedino između gena. Drugačija rješenja bi značajno zakomplicirala izvedbu operatora, a ne bi doprinjela učinkovitosti.

Križanje u jednoj točki točno je opisano izrazima (2.6), (2.7), (2.8), (2.9). Slučajnim odabirom se pronalazi cijeli broj k iz intervala [1..n] koji predstavlja točku prekida. U vektoru rješenja djeteta zamjenjuju se geni sa indeksima $i \geq k$ sa ekvivalentnim genima u drugom roditelju, u ovisnosti o slučajnom broju j . s_{cj}^t je kromosom dijete, a s_v^t i s_w^t su roditelji.

$$s_{cj}^t = \ll v_1, \dots, v_{(k-1)}, w_k, \dots, w_n \gg, \qquad za\ j = 0 \tag{2.6}$$

$$s_{cj}^t = \ll w_1, \dots, w_{(k-1)}, v_k, \dots, v_n \gg, \qquad za\ j = 1 \tag{2.7}$$

$$k = slučajan\ cijeli\ broj\ iz\ intervala(1,n) \tag{2.8}$$

$$j \in \{0,1\} \tag{2.9}$$

Jednostavno križanje u više točaka se provodi analogno prikazanom na Slika 2.4, a vrlo je slično onom s jednom točkom. Prethodi mu slučajan odabir željenog broja točaka prekida u obliku slučajnih cijelih brojeva iz intervala [1..n].

2.2.3 Aritmetičko križanje

Aritmetičko križanje je definirano kao linearna kombinacija dva vektora roditelja s_v^t i s_w^t iz kojih nastaju djeca s_{c1}^t ili s_{c2}^t u ovisnosti o slučajnom broju j (2.10), (2.11), (2.12), (2.13). U slučaju kada je a konstantna vrijednost govorimo o jednolikom aritmetičkom križanju. Kod nejednolikog aritmetičkog križanja a može poprimiti slučajnu vrijednost iz intervala $[0,1]$ ili u naprednijim implementacijama može ovisiti o starosti populacije.

$$s_{cj}^{t} = a \cdot s_{w}^{t} + (1-a) \cdot s_{v}^{t}, \qquad za\ j = 0 \tag{2.10}$$

$$s_{cj}^{t} = a \cdot s_{v}^{t} + (1-a) \cdot s_{w}^{t}, \qquad \mathbf{za\ j = 1} \tag{2.11}$$

$$a \in [0,1] \tag{2.12}$$

$$j \in \{0,1\} \tag{2.13}$$

Još jedna podvrsta aritmetičkog kodiranja je jednostruko aritmetičko kodiranje. Ono je u stvari linearna kombinacija dvaju slučajno odabranih gena u kromosomu roditelja. Ako u križanje ulaze s_v^t i s_w^t, njihovi potomci su s_v^{t+1} i s_w^{t+1} ((2.14), (2.15), (2.16), (2.17), (2.18)) gdje je $k \in [1,n]$ slučajni cijeli broj iz intervala. Parametar a ima jednako značenje kao i u prethodnom primjeru.

$$s_v^{t+1} = < v_1, \dots, v'_k, \dots, v_n > \tag{2.14}$$

$$v'_k = a \cdot w_k + (1-a) \cdot v_k \tag{2.15}$$

$$s_w^{t+1} = < w_1, \dots, w'_k, \dots, w_n > \tag{2.16}$$

$$w'_k = a \cdot v_k + (1-a) \cdot w_k \tag{2.17}$$

$$k \in [1,n] \tag{2.18}$$

2.3 Mutacija

Mutacija je unarni operator i u svom najjednostavnijem obliku, kod binarnog prikaza kromosoma, predstavlja promjenu jednog bita unutar kromosoma. Pospješuje izbjegavanje lokalnih optimuma te povećava raznolikost populacije. Općenito mutacija predstavlja slučajnu promjenu kromosoma.

Neka je s_v^t (2.19) kromosom odabran za mutaciju prije same mutacije, a s_v^{t+1} (2.20) kromosom nakon mutacije. Promjena se odvija samo na k -tom genu (v'_k) do kojega se dolazi slučajnim odabirom. Mutirani kromosom se od svojega originala razlikuje upravo u tom genu.

$$s_v^t = < v_1, \dots, v_n > \tag{2.19}$$

$$s_v^{t+1} = < v_1, \dots, v'_k, \dots, v_n > \tag{2.20}$$

$$k \in [1, n] \tag{2.21}$$

Razni tipovi mutacije se razlikuju u načinu na koji se dolazi do nove vrijednosti slučajno odabranog gena. Osim toga najčešće se definiraju granice domene rješenja na neki interval $[\mathrm{DG}, \mathrm{GG}]$ kako bi se suzio prostor pretrage, gdje su DG i GG donja i gornja granica domene rješenja.

2.3.1 Granična mutacija

Pojavljuje se kao granični slučaj ili jednostavnija implementacija jednolike mutacije. Gen v'_k uz jednaku vjerojatnost poprima vrijednost donje ili gornje granice domene rješenja (DG i GG).

2.3.2 Jednolika mutacija

Izabrani gen v'_k (2.22) poprima vrijednost slučajnog broja unutar intervala $[DD, GG]$.

$$v'_k = slučajna\ vrijednost\ iz\ intervala(DG, GG) \tag{2.22}$$

Jasno je da takav operator mutacije praktički pretražuje cijelu domenu rješenja problema, tokom cijelog vremena provođenja algoritma. No kada se populacija približi globalnom optimumu tj. algoritam ga na neki način raspozna, pretraga cijele domene postaje preširoka i ne daje dobre rezultate [2]. U tom trenutku je puno bolje tražiti na malom području gdje bi se trebao nalaziti optimum nego nasumično skakati po domeni rješenja.

2.3.3 Nejednolika mutacija

Smanjenje prostora pretrage mutacije iz generacije u generaciju omogućava algoritmu fino podešavanje rješenja kada je to potrebno. v'_k je gen koji se mutira prema fomuli (2.23).

$$v'_k = \begin{cases} v_k + \Delta(t, GG - v_k), ako\ slučajna\ znamenka = 0 \\ v_k - \Delta(t, v_k - DG), ako\ slučafna\ znamenka = 1 \end{cases} \tag{2.23}$$

Gdje funkcija $\Delta(t, y)$ vraća vrijednost iz intervala $[0, y]$ takvu da se vjerojatnost da će biti blizu nula povećava kako t raste. Upravo to ponašanje funkcije Δ omogućuje jednoliku pretragu prostora na početku izvođenja, a vrlo finu lokalnu u kasnijim generacijama. Korištena je funkcija definirana prema (2.24):

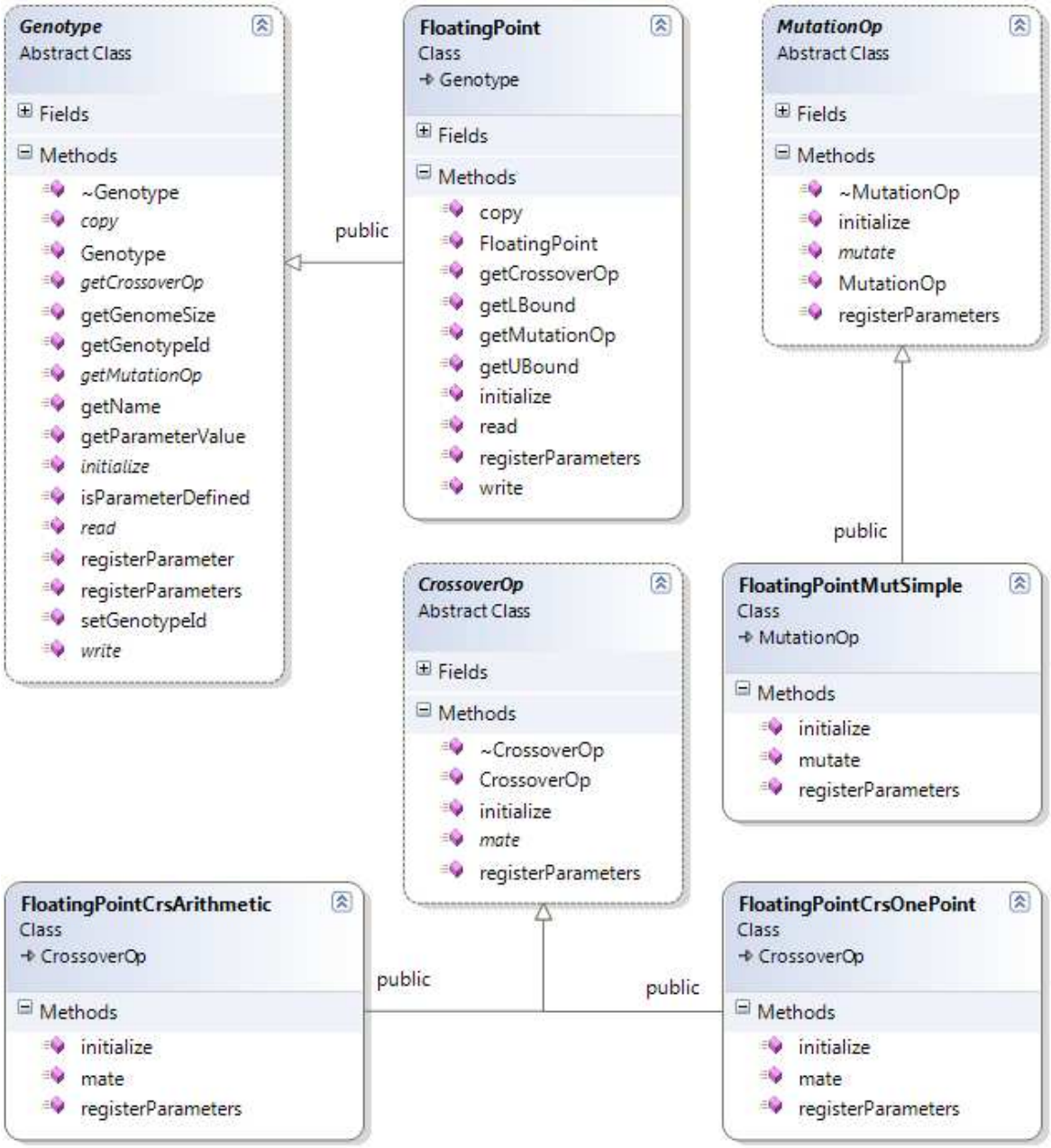
$$\Delta(t, y) = y \cdot \left(1 - r^{\left(1 - \frac{t}{T}\right)^b}\right) \tag{2.24}$$

gdje je r slučajan broj iz $[0,1]$, T maksimalni broj generacija i b sustavski parametar koji određuje stupanj nejednolikosti.

Gore spomenuti operator se može primjenjivati i na cijeli vektor rješenja što će uzrokovati njegovo lagano pomicanje u prostoru.

3. Ostvarenje prikaza s pomičnom točkom

U sklopu praktičnog dijela završnog rada ugrađena je, kao proširenje sustava ECF, podrška za prikaz u obliku niza brojeva s pomičnom točkom dvostruke preciznosti. ECF (*Evolutionary Computation Framework*) je radni okvir namjenjen za bilo koji tip evolucijskog računanja, pisan u programskom jeziku C++. Na Slici 3.1. prikazan je dijagram klasa dijela sustava ECF zaduženog za prikaz rješenja.



Slika 3.1: Dijagram klasa dijela sustava

Metode apstraktnih razreda *Genotype*, *MutationOp* i *CrossOverOp*, vidljive na Slici 3.1 sa nazivima pisanim u kurzivu su apstraktne metode koje je potrebno implemenirati u podrazredima. Apstraktni razredi preko apstraktnih metoda određuju komponente koje moraju biti definirane u izvedenim razredima ali ne definiraju u potpunosti njihovo ponašanje.

Razred svakog implementiranog genotipa, pa tako i u ovom slučaju razred *FloatingPoint*, mora nasljediti apstraktni razred *Genotype*. Razredi koji implementiraju mutaciju nasljeđuju *CrossoverOp*, a razredi zaduženi za mutaciju *MutationOp*. U Tablicama 3.1, 3.2 i 3.3 popisane su metode i njihova osnovna funkcija.

Tablica 3.1: Metode razreda FloatingPoint

Metoda	**Funkcija**
copy	stvara identičnu kopiju genotype objekta
getCrossoverOp *getMutationOp*	stvara i vraća vektor operatora križanja odnosno mutacije
getLbound *getUbound*	vraća donju odnosno gornju graničnu vrijednost domene zadane u konfiguracijskoj datoteci
initialize	inicijalizacija genotype objekta (čitanje i provjera ispravnosti parametara) i definiranje podatkovnih struktura
read	čitanje podataka o genotipu iz XMLNode
write	pisanje XML podataka o genotipu u XMLNode
registerParameters	registracija parametara genotipa (poziva se prije Genotype:initialize)

Tablica 3.2: metode razreda FloatingPointCrsArithmetic i FloatingPointCrsOnePoint

Metoda	Funkcija
initialize	inicijalizacija operatora križanja, poziva se prije prvog križanja
mate	obavlja križanje dva Genotype objekta, oba moraju biti inicijalizirana
registerParameters	registracija parametara sa sustavom (poziva se prije CrossoverOp::initialize)

Razredi zaduženi za križanje jednaki su po svojoj strukturi ali implementiraju različiti operator križanja, a jednako tako moguće je nadograditi sustav s više različitih operatora mutacije.

Tablica 3.3: metode razreda FloatingPointMutSimple

Metoda	Funkcija
initialize	inicijalizacija opratora mutacije, poziva se prije prve mutacije
mutate	obavlja mutaciju nad Genotype objektom koji mora biti inicijaliziran
registerParamaters	registracija parametara sa sustavom (poziva se prije MutationOp::initialize)

3.1 Mutacija

Razredom *FloatingPointMutSimple* ugrađena je jednostavna jednolika mutacija jednaka onoj opisanoj u poglavlju *2.3.2 Jednolika mutacija*

3.2 Križanje

Radni okvir je nadograđen s dva operatora križanja. *FloatingPointCrsOnePoint* implementira jednostavno križanje u jednoj točki prekida, identično onom opisanom u *2.2.2 Jednostavno križanje*, dok je nejednoliko aritmetičko križanje iz poglavlja *2.2.3 Aritmetičko križanje*, ostvareno u razredu *FloatingPointCrsArithmetic*.

4. Eksperimentalna ocjena učinkovitosti

Učinkovitost prikaza s pomičnom točkom se eksperimentalno ocjenjuje u usporedbi s binarnim prikazom. Prije toga uspoređena su dva ugrađena operatora križanja na jednodimenzijskom problemu optimiranja funkcije. Prilikom usporedbe koristit će se isti algoritam i što sličniji parametari sustava, sve na problemima pronalaženja optimuma višedimenzijskih kontinuiranih funkcija. Eksperimenti su provedeni na računalu s procesorom *Intel Core2 Duo CPU T7100 @1.8GHz*, 2GB RAM-a i operacijskim sustavom *Windows 7 Ultimate*.

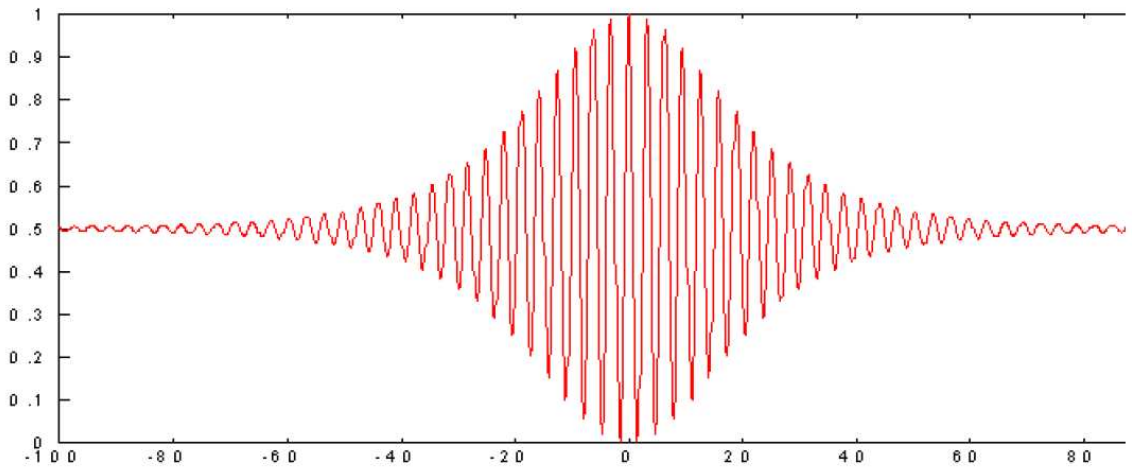
4.1 Algoritam

U svim kasnijim eksperimentima, ako nije drugačije navedeno, korišten je eliminacijski genetski algoritam s turnirskom selekcijom i veličinom turnira $tsize = 3$. Algoritam najprije slučajno odabire $tsize$ jedinki populacije, a zatim u ovisnosti o slučajnom parametru odabire najlošiju ili opet slučajnu jedinku. Odabrana jednika se zamjenjuje novom jedinkom koja je nastala križanjem preostalih jedinki. Ako je broj preostalih jedinki veći od dva, opet se slučajno biraju dvije koje postaju roditelji. Jedinka dijete, nakon nastanka, mutira uz zadanu vjerojatnost, koja u svim eksperimentima iznosi 0.5.

4.2 Ispitne funkcije

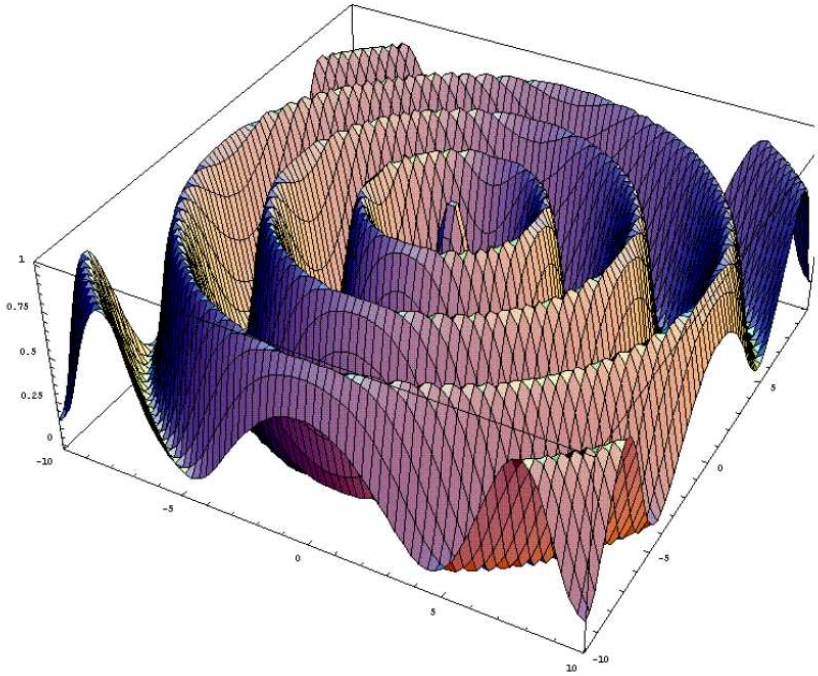
U ovom podpoglavlju definiraju se dvije ispitne funkcije koje će se koristiti u eksperimentima. Prva funkcija f_1 je zadana općenitom formulom za n dimenzijski prostor u (4.1). Graf sa Slika 4.1 odgovara funkciji iz formule (4.2) i predstavlja istu funkciju za $n = 1$. Slika 4.2 prikazuje graf dvodimenzijske funkcije.

$$f_1(\vec{x}) = 0.5 - \frac{\sin^2\sqrt{\sum_{i=1}^{N} x_i^2} - 0.5}{\left[1 + 0.001\sum_{i=1}^{N} x_i^2\right]^2} \qquad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}, \qquad x_1, x_2, \dots, x_N \in [-100,100] \tag{4.1}$$



Slika 4.1: graf funkcije $f_1(x_1)$

$$f_1(x) = 0.5 - \frac{\sin^2(x) - 0.5}{(1 + 0.001 \cdot x^2)^2}, \qquad x \in [-100,100] \tag{4.2}$$



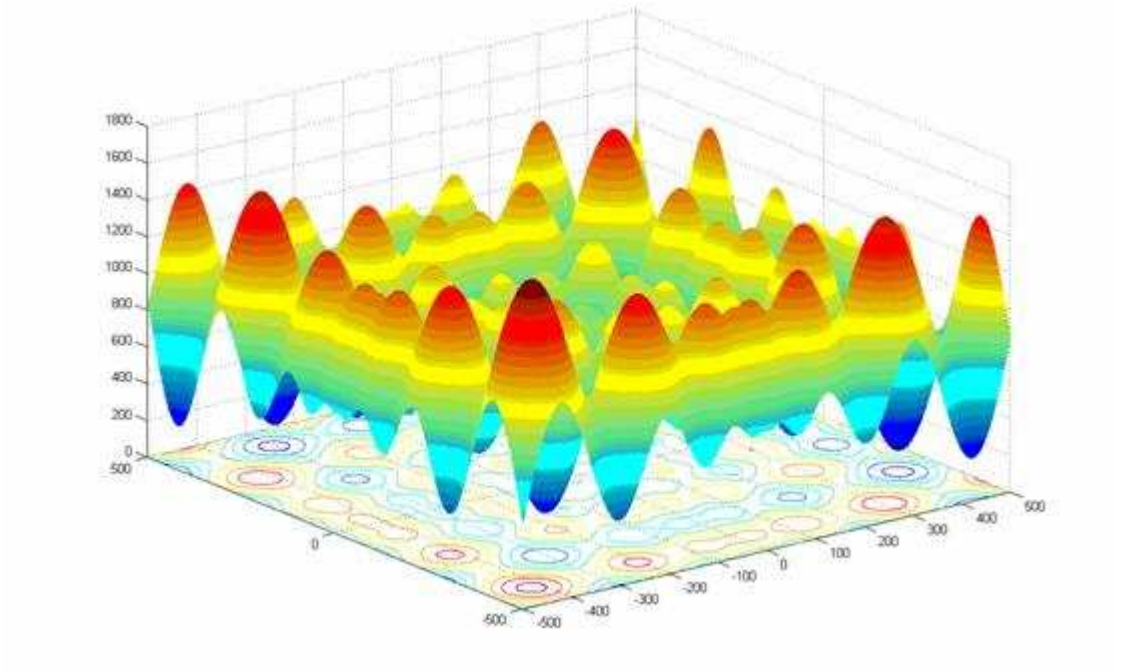
Slika 4.2: graf funkcije $f_1(x_1, x_2)$

U izrazima (4.1), (4.2) zadana je fiksna domena rješenja, dok će se u eksperimentima mjenjati ovisno o ispitnim slučajevima. Funkcija f_1 sadrži više lokalnih i jedan globalni optimum, $f_1(\vec{0}) = 1$. Vrijednost prvog susjednog lokalnog optimuma manja je za 1% vrijednosti globalnog. Najznačajniji dio pretrage rješenja odvija se u vrlo uskom intervalu oko nule.

Funkcija f_2 (Schwefelova funkcija) prikazana Slika 4.3 i definirana u (4.3) je druga ispitna funkcija. Kod ove funkcije je zanimljiva činjenica da je globalni minimum geometrijski prilično udaljen od slijedećeg najboljeg lokalnog minimuma. Algoritmi stoga lako mogu konvergirati u krivome smjeru.

$$f_2(x) = \sum_{i=1}^{N} -x_i \sin\sqrt{|x_i|}$$

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}, \qquad x_1, x_2, \dots, x_N \in [-500, 500] \tag{4.3}$$



Slika 4.3: graf funkcije $f_2(x_1, x_2)$

4.3 Usporedba operatora križanja

Rezultati usporedbe dvaju implementiranih križanja prikazani su u Tablica 4.1. Ispitna funkcija je funkcija f_1, domena rješenja je postavljena relativno široko s obzirom na izgled funkcije $[-100, 100]$. Ostali parametri sustava napisani su u tablici. Vjerojatnost mutacije je vjerojatnost da će jedinka dijete nakon što je stvorena križanjem biti i mutirana. Statistička obrada podataka provedena je na 20 uzoraka za svako križanje.

Tablica 4.1: Usporedba operatora križanja

dimenzija problema	1D	
veličina populacije	30	
broj generacija	500	
vjerojatnost mutacije	0.5	
broj eksperimenata	20	
prikaz	niz brojeva s pomičnom točkom	
križanje	aritmetičko	s točkom prekida
postignut globalni optimum *f(x)=1*	100%	10%
postignut prvi lokalni optimum *f(x)≅0.99*	-	90%
$\overline{\max(f(x))}$	1	0.9097
$\bar{d}$ - prosječno odstupanje	2.23E-5	3.42E-2
σ - standardna devijacija	5.6E-5	7.11E-2
prosječno vrijeme izvođenja	7.3	7.1
prosječna generacija najboljeg rezultata	156	301

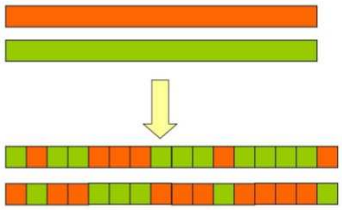
Iz dobivenih rezultata je očito da uz navedene parametre sustava i na navedenoj ispitnoj funkciji aritmetičko križanje dolazi do oprimuma u 100% slučajeva, dok ono u jednoj točki prekida pronalazi samo prvi lokalni optimum u čak 90% slučajeva. Osim toga prosječno odstupanje od maksimuma je tri reda veličine manje kod aritmetičkog križanja. Iako je vrijeme izvođenja vrlo slično, prosječna generacija u kojoj algoritam postiže najbolji rezultat je puno veća kod križanja s jednom točkom prekida. Može se reći da algoritam koji koristi križanje u jednoj točki prekida kasnije dolazi do optimuma. Aritmetičko križanje se na ovom problemu pokazalo brže i točnije.

4.4 Usporedba prikaza

Binarni prikaz koji ćemo koristiti za usporedbu otprije je dio sustava ECF. Osim donje i gornje granice prikaza (DG, GG), parametar sustava je i $preciznost$, broj decimala realnog broja kojeg prikaz predstavlja. Izrazom (4.4) se računa broj bitova koji je potreban za prikaz.

$$brBit \geq log_2\left((GG - DG) \cdot 10^{preciznost}\right) + 1 \tag{4.4}$$

Također, važno je napomenuti da se za svaku jedinku, osim binarne reprezentacije, sprema i dekadska i realna vrijednost. To bitno utječe na performanse algoritma kod velike preciznosti. Operacija mutacije nad ovim genotipom slučajno odabire bit u slučajno odabranoj dimenziji i invertira ga. Implementirano je jednoliko križanje i križanje s jednom točkom prekida. Slika 4.4 prikazuje jednoliko križanje. Svaki bit kromosoma djeteta prepisuje se iz jednog od dva roditelja uz jednaku vjerojatnost odabira.



Slika 4.4: jednoliko križanje

Prikaz u obliku niza brojeva s pomičnom točkom opisan je u poglavlju *2. Prikaz s pomičnom točkom.*

Kako bi se dobili rezultati za što općenitiji slučaj, sustav je konfiguriran tako da u svakoj generaciji nasumce bira koji će oprator križanja, od dva implementirana nad prikazom s pomičnom točkom, biti korišten.

Prije provođenja eksperimenata pretpostavljeno je da će prikaz s pomičnom točkom pokazati svoje prednosti kod rješavanja problema velikog broja dimenzija i problema koji zahtjevaju veliku preciznost. Razlika u vremenima izvođenja trebala bi rasti u korist prikaza s pomičnom točkom porastom. Naime, ažuriranje promjena nad populacijom, uzrokovanih genetskim operatorima, zahtjeva znatno više CPU vremena kod binarnog prikaza nego što je to slučaj kod prikaza s pomičnom točkom. Operatori i sam prikaz s pomičnom točkom su takvi da ne zahtjevaju posebne pretvorbe nakon promjene. Osim toga predpostavka je da će za veće domene prikaz s pomičnom točkom biti precizniji odnosno davati bolje rezultate.

4.4.1 Eksperiment 1

Unutar prvog eksperimenta optimirana je funkcija $\mathbf{f_1}$ na intervalu $[-100, 100]$ za razne dimenzije i veličine populacije koje im odgovaraju. Korišten je binarni prikaz duljine gena 32 bita. Takav prikaz na zadanoj širini domene daje preciznost na sedam decimala. Ostali parametri algoritma koji nisu prije zadani napisani su u tablici 4.2, a kratica PT u tablici rezultata označava prikaz s pomičnom točkom.

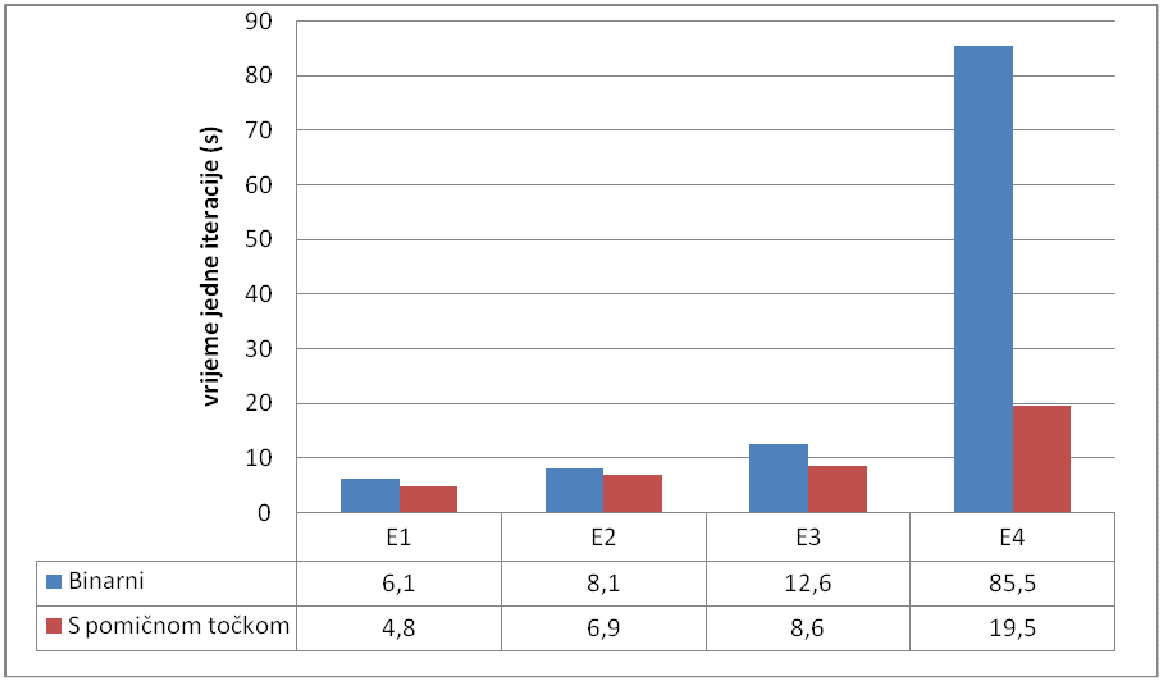
Tablica 4.2: rezultati u eksperimentu 1.

dimenzija problema	1D				2D		10D	
oznaka eksperimenta	E1		E2		E3		E4	
veličina populacije	20		30		60		100	
broj generacija	400		500		500		1000	
vjerojatnost mutacije	0.5		0.5		0.5		0.5	
broj ponavljanja	20		20		20		20	
prikaz	Binarni	PT	Binarni	PT	Binarni	PT	Binarni	PT
postignut globalni optimum; $f(x)=1$	70%	45%	100%	95%	10%	10%	-	-
postignut prvi lokalni optimum; $f(x)\cong 0.99$	30%	55%	-	5%	65%	90%	-	-
postignut lokalni optimum za koji vrijedi: $f(x)>0.8$	-	-	-	-	25%	-	75%	100%
$\overline{max(f(x_i))}$	0.997	0.9998	1	0.9999	0.96	0.988	0.84	0.925
$\bar{d}$ - prosječno odstupanje	-9.75E-1	7.02E-3	4.65E-8	5.06E-4	3.54	2.1	3.73	2.42
σ - standardna devijacija	1.85	0.012	4.79E-8	9.4E-4	5.29	2.46	4.56	2.93
Prosječno vrijeme izvođenja	6.1	4.8	8.1	6.9	12.6	8.6	85.5	19.5

U eksperimentu E1, kada populaciju čini 20 jedinki i radi se na jednodimenzijskom problemu, binarni prikaz daje malo bolje rezultate u odnosu na prikaz s brojem s pomičnom točkom. No, već u slijedećem eksperimentu je vidljivo da povećanjem

populacije na 30 jedinki, oba prikaza daju slične i značajno bolje rezultate za nešto više vremena. I kod ovako malih dimenzija i populacija, vidljiva je prednost u brzini izvođenja na stranu prikaza s pomičnom točkom. Na dvodimenzijskom problemu oba prikaza dolaze do globalnog optimuma u svega 10% slučajeva, no prikaz s pomičnom točkom postiže prvi lokalni optimum u čak 90% slučajeva. Prednost prikaza s pomičnom točkom za kompleksnije probleme većih dimenzija potvrđuje se izvođenjem eksperimenta E4 kada prikaz s pomičnom točkom postiže lokalni optimum za koji vrijedi: $f(x) > 0.8$ u 100% slučajeva, a binarni prikaz u 75%. Prosječni $f(x)$ koji postiže algoritam u ovom eksperimentu je znatno bolji kod prikaza s pomičnom točkom (0.925) nego kod binarnog prikaza (0.84). Također vrijeme izvođenja jedne iteracije je četiri puta veće kod binarnog prikaza. Usporedbu prosječnih vremena izvođenja jedne iteracije prikazuje

Slika
4.5.

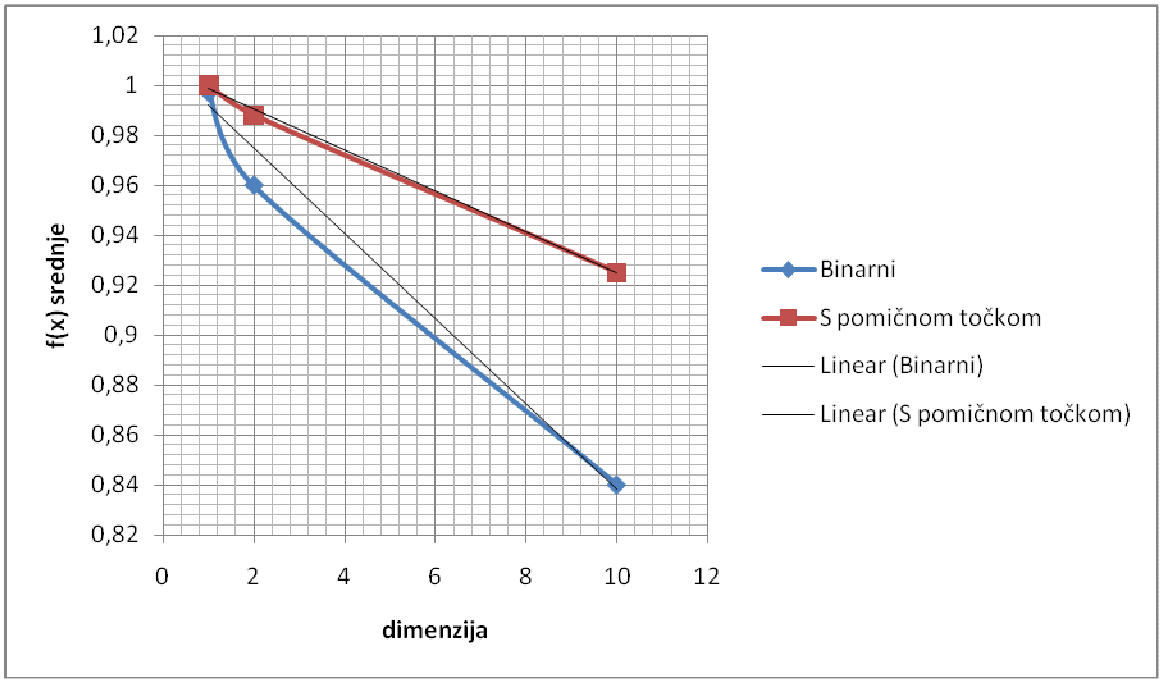


	E1	E2	E3	E4
Binarni	6,1	8,1	12,6	85,5
S pomičnom točkom	4,8	6,9	8,6	19,5

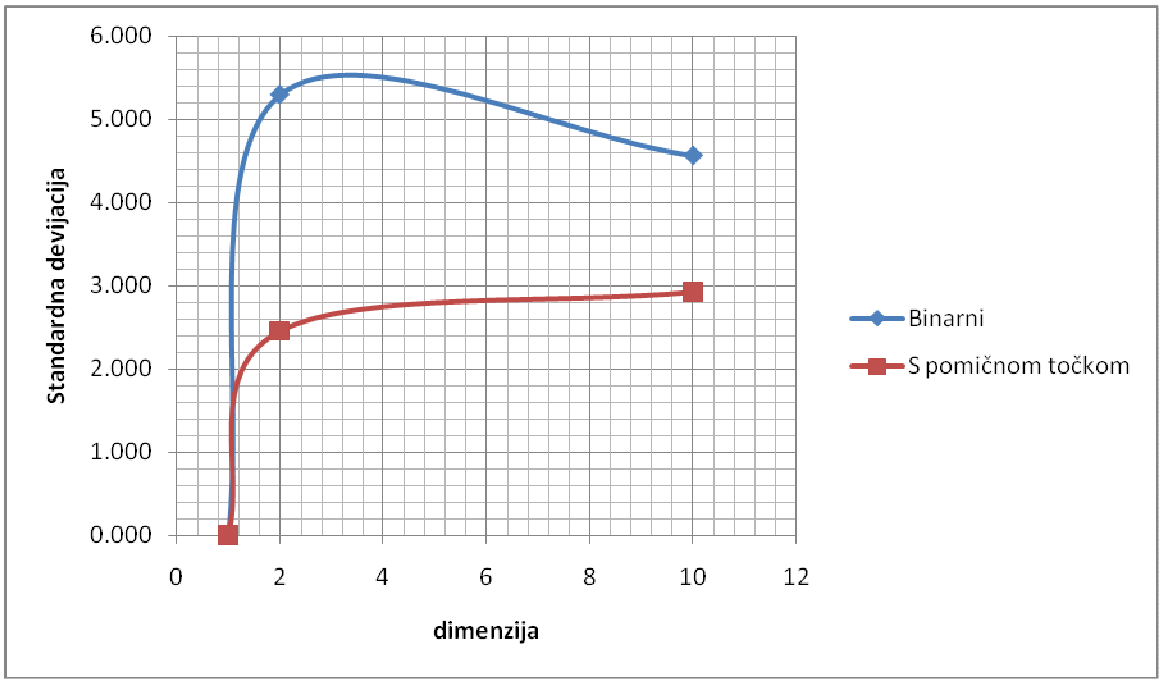
Slika 4.5: grafički prikaz duljine izvođenja eksperimenata

Eksperimentalni rezultati su približno jednaki za jednodimenzijski problem, no prikaz s pomičnom točkom se pokazuje sve bolji proporcionalno s rastom

dimenzije problema. To se najbolje vidi na grafu s prikazanom Slikom 4.6, koji prikazuje ovisnost prosječno dobivenog $f(x)$ o dimenziji problema. Bolji su oni rezultati koji su bliži globalnom optimumu $f(0)=1$. Prikaz s pomičnom točkom je stabilniji, što pokazuju iznosi standardne devijacije, a kretanje standardne devijacije koja raste proporcionalno s dimenzijom problema prikazano je na grafu (slike 4.6 i 4.7).



Slika 4.6: graf ovisnosti f(x) o dimenziji problema



Slika 4.7: graf ovisnosti standardne devijacije o dimenziji problema

Valja primjetiti da je prikazom realnog broja s pomičnom točkom dvostruke preciznosti prema IEEE 754-1985 standardu gustoća realnih brojeva koji se daju prikazati najveća oko nule. Neka postoji pravac na kojem su označeni svi brojevi koji se daju prikazati standardom. Kako se udaljavamo od nule, na takvom pravcu, gustoća brojeva opada s potencijom broja 2.

U ovom eksperimentu optimirana je s funkcija s minimumom $f_1(0) = 1$, upravo tamo gdje je gustoća prikaza s pomičnom točkom najveća. Moguće je da je upravo ta činjenica razlog prednosti prikaza s pomičnom točkom, naravno ako zanemarimo na vrijeme izvođenja koje nema veze s gustoćom prikaza.

4.4.2 Eksperiment 2

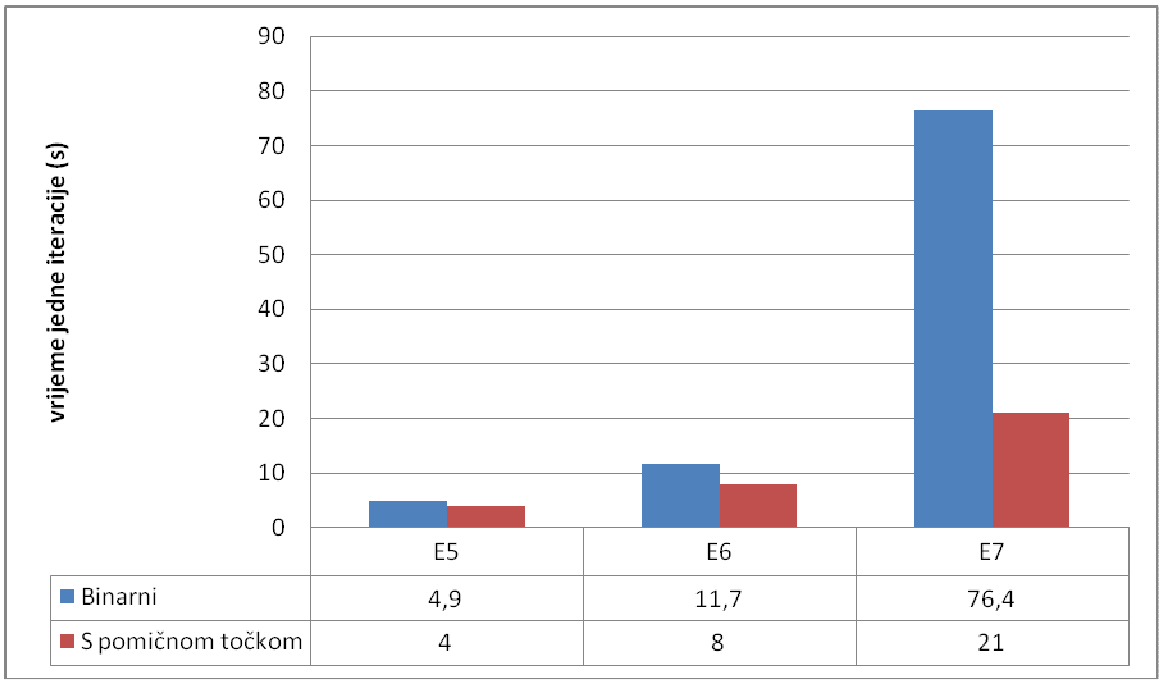
Drugi eksperiment bi trebao pokazati ima li udaljenost globalnog minimuma u odnosu na ishodište značajnu ulogu kod prikaza s pomičnom točkom. Stoga je optimirana funkcija f_2 na intervalu $[-500, 500]$ opisana u poglavlju 4.2. Opet je korišten binarni prikaz preciznosti na 5 decimala i kratica PT u tablici rezultata označava prikaz u obliku broja s pomičnom točkom.

Tablica 4.3: rezultati u eksperimentu 2.

dimenzija problema	**1D**		**2D**		**10D**	
oznaka eksperimenta	E5		E6		E7	
veličina populacije	30		60		100	
broj iteracija	300		500		1000	
vjerojatnost mutacije	0.5		0.5		0.5	
broj ponavljanja	20		20		20	
globalni optimum $f(x)$	-418.9829		-837.966		-4189.829	
prikaz	Binarni	PT	Binarni	PT	Binarni	PT
postignut globalni optimum	100%	100%	95%	100%	-	100%
postignut lokalni optimum; $0.99 \cdot f(x)$	-	-	-	-	20%	-
Postignut lokalni optimum $< 0.99 \cdot f(x)$	-	-	5%	-	80%	-
$\overline{min(f(x))}$	-418.92	-418.96	-833.63	-837.966	-4046	-4189.82
$\bar{d}$ - prosječno odstupanje	0.551	4E-3	1.28	0	67.3	1.82E-2
σ - standardna devijacija	0.477	0.008	4.097	1.17E-13	208.69	0.0397
Prosječno vrijeme izvođenja	4.9	4	11.7	8	76.4	21

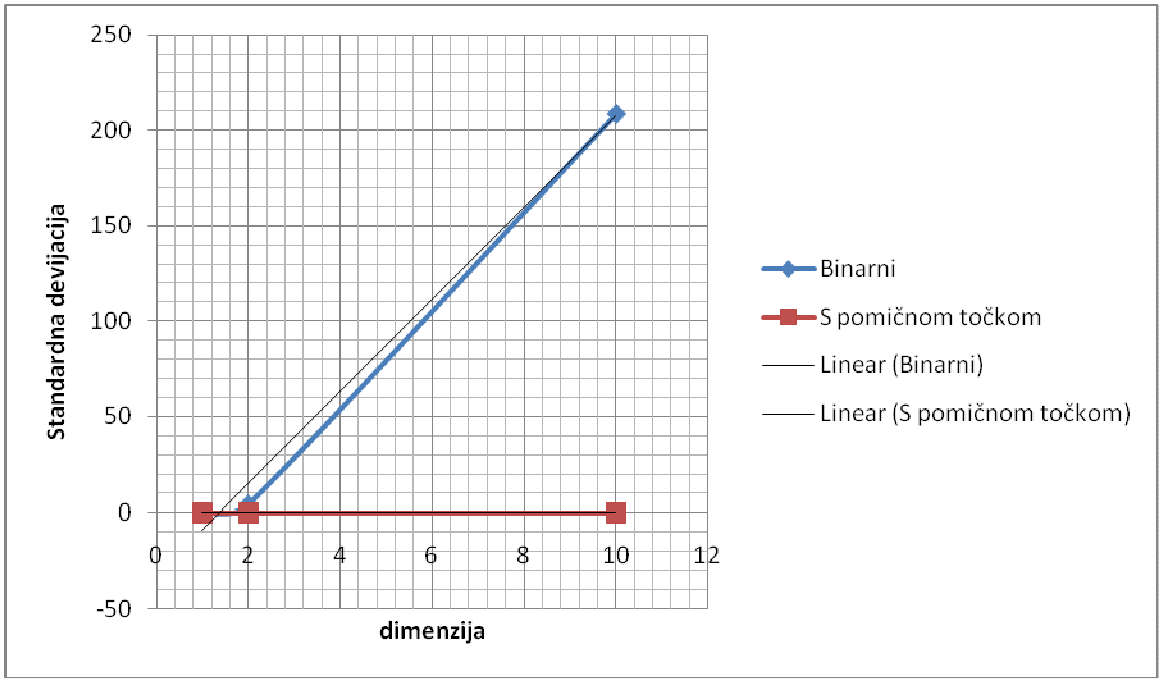
Prikaz s pomičnom točkom u sva tri prikazana eksperimenta u 100% slučajeva postiže globalni optimum. Binarni prikaz pokazuje izvrsne rezultate za jednodimenzijski i dvodimenzijski problem. Ipak lokalni optimum od $0.99 \cdot f(x)$ za problem od deset dimenzija postiže u samo 20% dok su svi ostali rezultati koje daje manji od toga.

Slika 4.8 pokazuje značajno veće vrijeme izvođenja jedne iteracije algoritma za binarni prikaz. Opet je prikaz u obliku brojeva s pomičnom točkom značajno stabilniji (Slika 4.9).



Slika 4.8: grafički prikaz duljine izvođenja eksperimenata

Rezultati koje daje prikaz s pomičnom točkom za ovaj problem su čak i bolji nego u prethodnom eksperimentu. Pretpostavka, da su rezultati algoritma dok koristi prikaz s pomičnom točkom nad problemima u kojima je optimum pomaknut od ishodišta lošiji, ne stoji.



Slika 4.9: graf ovisnosti standardne devijacije o dimenziji problema

5. Zaključak

Prikaz u obliku brojeva s pomičnom točkom očekivano je dao bolje rezultate od binarnog prikaza. Brži je, daje stabilnije i točnije rezultate, posebno kod velikih domena i dimenzija problema. Zaključke treba uzeti s rezervom s obzirom na ipak mali broj ispitivanja za ozbiljnije statističke analize. Osim toga moguće je da bi se binarni prikaz pokazao boljim za neke druge probleme.

Prikaz u obliku niza brojeva s pomičnom točkom, intuitivno bliži domeni problema, omogućava lakše dizajniranje novih operatora tako da uključuju specifičnosti koje svaki problem donosi. Korišteni su jednostavni operatori u oba prikaza, pa daljnja istraživanja treba usmjeriti upravo prema implementaciji i analizi složenijih operatora.

6. Literatura

[1] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Treće prošireno izdanje. Mjesto izdavanja: Springer-Verlag Berlin Heidelberg New York.

[2] Doc. dr. sc. Marin Golub, ZEMRIS, Projekt: Evolucijski Algoritmi, 2007-2008,

http://www.zemris.fer.hr/~golub/ga/studenti/projekt2007/, 5.6.2010.

[3] Dr. sc. Domagoj Jakobović, Evolutionary Computation Framework, 11.05.2010., http://gp.zemris.fer.hr/ecf/, 25.5.2010.

[4] Golub, Marin, Vrednovanje uporabe genetskih algoritama za aproksimaciju vremenskih nizova, magistarski rad, Zagreb: Fakultet elektrotehnike i računarstva,1996.

[5] Hartmut Pohlheim, Examples of Objective Functions, 2006, http://www.geatbx.com/docu/fcnindex-01.html#TopOfPage, od 25.5.2010.

Učinkovitost prikaza s pomičnom točkom u evolucijskim algoritmima

Sažetak

U ovom radu je ocjenjena učinkovitost prikaza s pomičnom točkom na kontinuiranim optimizacijskim problemima. Opisana je implementacija prikaza s pomičnom točkom u radni okvir za evolucijsko računanje ECF, te su implementirani jednostavna mutacija, križanje u jednoj točki prekida i aritmetičko križanje. Opisani su i neki drugi, uobičajeni genetski operatori nad prikazom s pomičnom točkom. Prikazani su i analizirani rezultati eksperimenata za ocjenu učinkovitosti.

Ključne riječi: genetski algoritmi, ECF, prikaz u obliku brojeva s pomičnom točkom, genetski operatori, optimizacijski problem.

Floating-point representation efficiency in evolutionary algorithms

Abstract

This document evaluates the efficiency of floating-point representation on continuous optimization problems. It describes the implementation of representation in framework for evolutionary computation and simple mutation, simple one point and arithmetic crossover are implemented. Also, other genetic operators on floating- point representation, are described. Analyzed results of experiments for evaluation of efficiency are shown.

Keywords: genetic algorithms, ECF, floating-point representation, genetic operators, optimization problem.