SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT

Diferencijska evolucija

Zoran Dodlek, 0036429614

Voditelj: *doc. dr. sc. Marin Golub*

Zagreb, prosinac, 2008.

Sadržaj

1. Uvod

Diferencijska evolucija (DE) je stohastički, populacijski, optimizacijski evolucijski algoritam koji su uveli Storn i Price 1996. godine. Napravljen je za optimiziranje funkcija s realnim varijablama (eng. *real valued functions*).

Po svojim značajkama, DE je najbliža evolucijskoj strategiji (tip (μ/ρ + λ) - ES). Njihova glavna razlika leži u mutaciji. Dok kod ES mutacija djeluje na samo jednu jedinku, kod DE mutacija djeluje na sve jedinke (vektore) u populaciji. Slučajno se odaberu 3 vektora, te napravi zbrajanje jednog vektora s težinskom razlikom druga dva vektora. Prema toj razlici (diferencija), DE je dobio ime. Rekombinacija DE je križanje u *D* točaka jer ovisi o slučajno odabranim realnim brojevima iz intervala [0,1]. [4]

Formulacija općenitog problema optimizacije glasi:

Za funkciju dobrote

$$f : X \subseteq \mathbb{R}^D \to \mathbb{R} \tag{1.1}$$

gdje skup svih mogućih rješenja optimizacijskog problema X (eng. *feasible region*) nije prazan.

Minimizacijski problem je pronalazak:

$$\begin{aligned} &x^* \in X \text{ za kojeg vrijedi } f(x^*) \leq f(x),\ \forall x \in X, \\ &f(x^*) \neq -\infty \end{aligned} \tag{1.2}$$

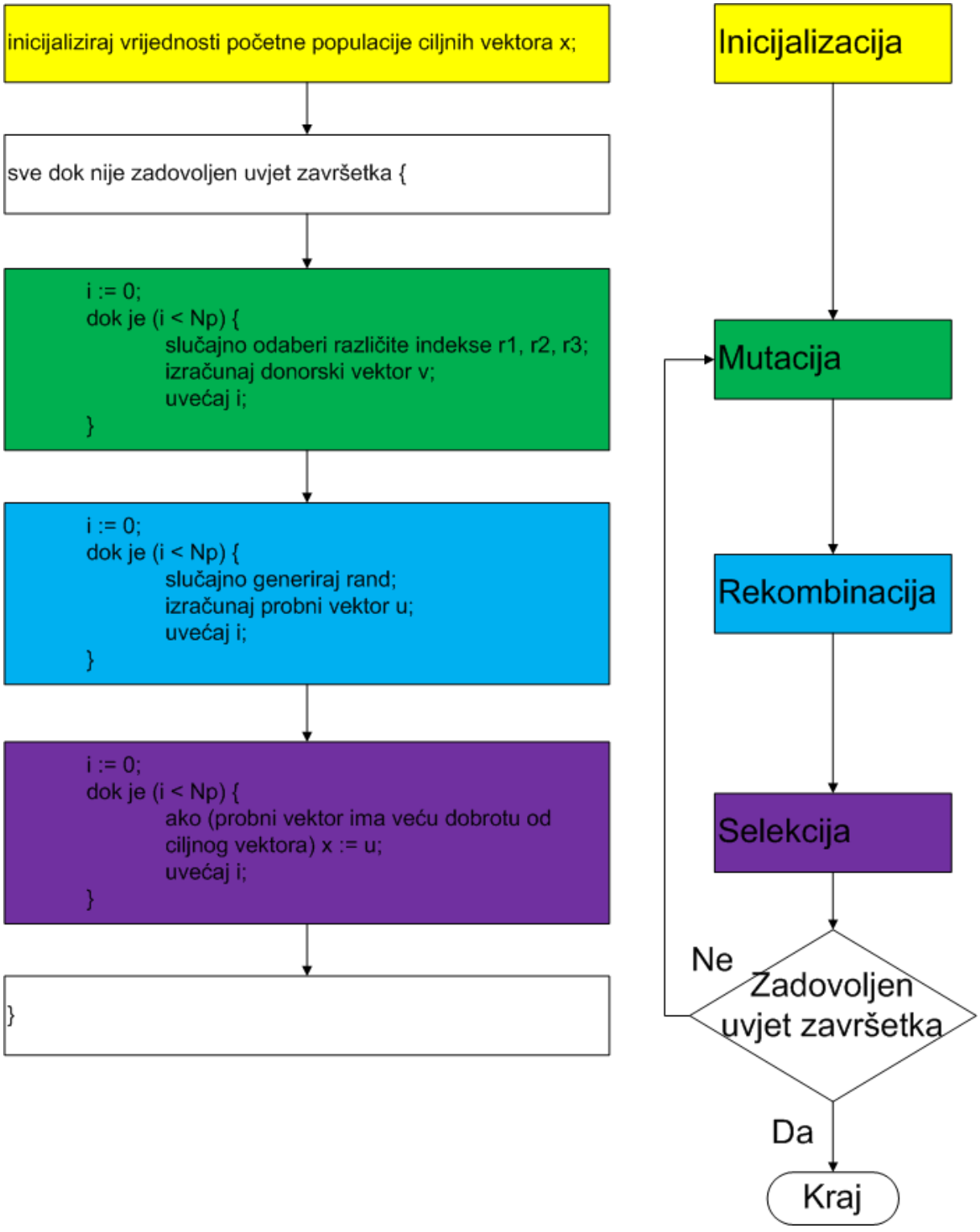
Funkcija dobrote (eng. *objective function*) se ponekad naziva funkcija cijene (eng. *cost function*).

1.1 Primjene algoritma

Globalna optimizacija je potrebna u raznim poljima, poput inženjerstva, statistike i financija. Mnogi praktični problemi imaju funkcije dobrote koje nisu diferencijalne, kontinuirane, linearne ni predvidljive. Takve funkcije su često i višedimenzionalne, te imaju šumove i mnogo lokalnih ekstrema i ograničenja. Takvi problemi se mogu teško ili nikako riješiti analitički. DE se može upotrijebiti radi pronalaska približnog rješenja za takve probleme. [1]

2. Opis algoritma

2.1 Pseudokod



Slika 2.1 Pseudokod DE

2.2 Notacija i algoritam ukratko

Traži se optimizacija funkcije *f* s *D* realnih parametara. Odabere se veličina populacije *Np* (koja mora biti veća od 3). Parametri vektora (eng. *parameter vectors*) su oblika:

$$x_{i,G} = \left[x_{1,i,G}, x_{2,i,G}, ... x_{D,i,G} \right], \; i = 1, 2, ..., Np. \tag{1.3}$$

gdje je G broj generacije.

Koriste se 3 vektora:

- Ciljni vektor (eng. *target vector*) $x_{i,G}$
- Donorski vektor (eng. *donor vector*) $v_{i,G}$
- Probni vektor (eng. *trial vector*) $u_{i,G}$

Donorski vektor $v_{i,G+1}$ nastaje slučajnim odabirom 3 ciljna vektora $x_{i,G}$, kada se napravi zbrajanje jednog vektora s težinskom razlikom druga dva vektora (mutacija). Probni vektor $u_{i,G+1}$ nastaje od elemenata ciljnog vektora $x_{i,G}$ i elemenata donorskog vektora $v_{i,G+1}$ (rekombinacija). Ciljni vektor $x_{i,G+1}$ nastaje selekcijom između probnog vektora $u_{i,G+1}$ i ciljnog vektora $x_{i,G}$ (selekcija).

2.3 Inicijalizacija

Definiranje gornje i donje granice za svaki parametar *j* :

$$x_j^L \le x_{j,i,1} \le x_j^U \tag{1.4}$$

Slučajnim odabirom inicijaliziraju se vrijednosti parametara uniformno na intervalu $\left[x_j^L, x_j^U \right]$.

Svaki od *Np* parametara vektora prolazi kroz mutaciju, rekombinaciju i selekciju kao što je vidljivo na slici 2.1.

2.4 Mutacija

Mutacija proširuje prostor potrage rješenja (eng. *search space*). Za dani parametar vektora $x_{i,G}$ slučajno se odaberu tri vektora $x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$ s međusobno različitim indeksima.

Izračuna se:

$$\begin{aligned} &v_{i,G+1} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G}), \\ &\text{gdje je } r1, r2, r3 \in \left[1, 2, ..., Np \right], \; r1 \ne r2 \ne r3 \end{aligned} \tag{1.5}$$

$v_{i,G+1}$ je donorski vektor (eng. *donor vector*).

Mutacijski faktor *F* je konstanta na intervalu [0,2]. On kontrolira brzinu i robusnost potrage rješenja. Tj. manja vrijednost povećava vjerojatnost konvergencije, ali povećava i rizik zaustavljanja u lokalnom ekstremu.

2.4.1 Varijacije mutacije

Umjesto slučajno odabranog $x_{r1,G}$ odabere se najbolji. [2]

Umjesto jednog oduzimanja, odabere se više vektora radi bolje varijacije:

$$v_{i,G+1} = x_{r1,G} + F*(x_{r2,G} - x_{r3,G} + x_{r4,G} - x_{r5,G}) \tag{1.6}$$

2.5 Rekombinacija

Uključuje uspješna rješenja od prijašnje generacije. Probni vektor (eng. *trial vector*) $u_{i,G+1}$ nastaje od elemenata ciljnog vektora (eng. *target vector*) $x_{i,G}$ i elemenata donorskog vektora $v_{i,G+1}$. $\mathrm{I}_{\mathrm{rand}}$ predstavlja slučajno odabran broj iz intervala [1,2,...,*D*]. Vjerojatnost križanja (eng. *crossover*) *CR* je konstanta na intervalu [0, 1]. Parametar vektora $u_{i,G+1}$ s indeksom j = $\mathrm{I}_{\mathrm{rand}}$ će biti jednak $v_{i,G+1}$. $\mathrm{I}_{\mathrm{rand}}$ uvijek osigurava različitost $u_{i,G+1}$ i $x_{i,G}$ (tj. $u_{i,G+1} \neq x_{i,G}$) i onda kada je *CR* jednak nuli. Elementi donorskog vektora ulaze u probni vektor s vjerojatnošću *CR*.

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} \text{ ako rand}_{j,i} \leq CR \text{ ili } j = \mathrm{I}_{\mathrm{rand}} \\ x_{j,i,G} \quad \text{ako rand}_{j,i} > CR \text{ i } j \neq \mathrm{I}_{\mathrm{rand}} \end{cases}, \tag{1.7}$$

gdje su:

$i = 1, 2, ..., Np$

$j = 1, 2, ..., D$

$\text{rand}_{j,i} \sim U[0,1]$

2.6 Selekcija

Ciljni vektor $x_{i,G}$ se uspoređuje s probnim vektorom $u_{i,G+1}$, te onaj s većom dobrotom (tj. manjom funkcijskom vrijednosti) prolazi u slijedeću generaciju

$$x_{i,G+1} = \begin{cases} u_{i,G+1} \text{ ako } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} \qquad \text{inače} \end{cases}, \; i = 1, 2, ..., Np \tag{1.8}$$

Budući da se prosječna funkcijska vrijednost nikad neće pogoršati, selekcija je elitizam.

Mutiranje, rekombinacija i selekcija se nastavljaju sve dok nije zadovoljen neki uvjet zaustavljanja, koji je najčešće broj odrađenih generacija ili dovoljno dobra vrijednost funkcije cijene. U uvjetu zaustavljanja algoritma zadaje se potrebna točnost rezultata ili sigurnost ispravnosti rezultata. Npr. veći broj odrađenih generacija poboljšava ispravnost rezultata, ali potrebno vrijeme izvršavanja algoritma se produljuje.

3. Primjer rada DE

Rad DE prikazan je na jednostavnom primjeru optimiranja funkcije dobrote jedne realne varijable. Zadatak je pronaći globalni optimum funkcije $f(x) = x^2$ u intervalu [-5,5]. Globalni minimum postiže se u točki (0,0). Algoritam je pokrenut s slijedećim vrijednostima:

$$F = 1;\ D = 1;\ Np = 4;\ CR = 0.9 \tag{1.9}$$

Uvjet zaustavljanja je $f(x) \leq 0.01$, za neki x iz populacije vektora $x_{i,G}$.

3.1 Prva iteracija

1. korak: inicijalizacija. Prilikom inicijalizacije generira se početna populacija (prva generacija) s pomoću generatora slučajnih brojeva:

Tablica 3.1 Početna populacija $x_{i,1}$ i dobrote jedinki

	i	**Populacija** $x_{i,1}$	f($x_{i,1}$)
1. generacija	1	**-2.841187**	8.072344
	2	**1.785889**	3.189400
	3	**-0.702515**	0.493527
	4	**-2.186279**	4.779816

U tablici 3.1 postoje četiri vektora populacije $x_{i,1}$ od kojih treći ima najbolju dobrotu.

2. korak: mutacija. U tablici 3.2 nalaze se slučajno odabrani indeksi r1, r2, r3 i populacija vektora $v_{i,2}$ koja je nastala primjenom formule (1.5):

Tablica 3.2 Populacija vektora $v_{i,2}$ s indeksima vektora $x_{i,1}$ od kojih je nastao

i	r1	r2	r3	**Populacija** $v_{i,2}$
1	2	1	3	**-0.352783**
2	1	2	4	**1.130981**
3	3	2	1	**3.924561**
4	1	4	3	**-4.324951**

3. korak: rekombinacija. Prema formuli (1.7), slučajnim odabirom se izaberu parametri vektora $u_{i,2}$. Budući da je D = 1 (1.9), iz čega slijedi da je $I_{rand} = 1$, formula se pojednostavljuje:

$$u_{j,i,G+1} = v_{j,i,G+1} \tag{1.10}$$

Tablica 3.3 Populacija vektora $u_{i,2}$ i dobrote jedinki

i	**Populacija** $u_{i,2}$	f($u_{i,2}$)
1	**-0.352783**	0.124456
2	**1.130981**	1.279118
3	**3.924561**	15.402179
4	**-4.324951**	18.705201

U tablici 3.3 je prikazana populacija vektora $u_{i,2}$ koja je identična populaciji vektora $v_{i,2}$.

4. korak: selekcija. U tablici 3.4 su prikazane dobrote populacija vektora $x_{i,1}$ i $u_{i,2}$, te populacija vektora $x_{i,2}$ s njegovom vrijednosti funkcije dobrote. Prema formuli (1.8) donosi se odluka o vektoru koji se dalje nasljeđuje.

Tablica 3.4 Nastanak druge generacije vektora $x_{i,2}$

i	f($x_{i,1}$)	f($u_{i,2}$)		**Populacija** $x_{i,2}$	f($x_{i,2}$)
1	8.072344	0.124456	2. generacija	**-0.352783**	0.124456
2	3.189400	1.279118		**1.130981**	1.279118
3	0.493527	15.402179		**-0.702515**	0.493527
4	4.779816	18.705201		**-2.186279**	4.779816

Gotova je druga generacija. Provjerava se da li je zadovoljen uvjet zaustavljanja. Budući da ne postoji niti jedan *f*(*x*) koji je manji od 0.01, algoritam se nastavlja.

3.2 Druga iteracija

U tablici 3.5 prikazana je cijela treća generacija koja je stvorena iz populacije $x_{i,2}$ prema koracima 2, 3 i 4. iz prethodnog poglavlja. Jedino se četvrti vektor populacije $x_{i,3}$ promijenio u odnosu na populaciju $x_{i,2}$. Uvjet zaustavljanja još nije zadovoljen.

Tablica 3.5 Kompletna treća generacija

	i	r1	r2	r3	**Populacija $v_{i,3}$**	**Populacija $u_{i,3}$**	f($u_{i,3}$)	**Populacija $x_{i,3}$**	f($x_{i,3}$)
3. generacija	1	1	3	4	**1.130981**	**1.130981**	1.279118	**-0.352783**	0.124456
	2	2	1	3	**1.480713**	**1.480713**	2.192511	**1.130981**	1.279118
	3	4	3	1	**-2.536011**	**-2.536011**	6.431352	**-0.702515**	0.493527
	4	2	3	1	**0.781249**	**0.781249**	0.610350	**0.781249**	0.610350

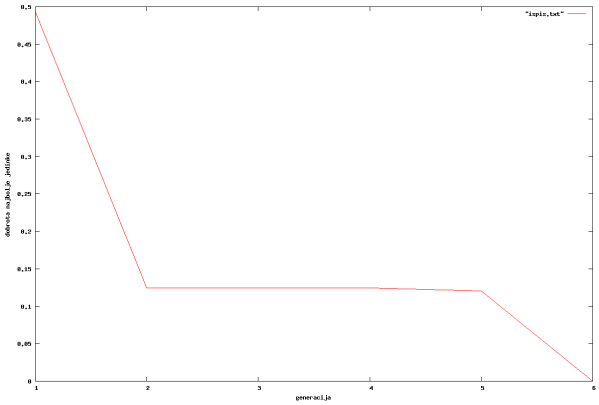
3.3 Ostale iteracije:

U tablici 3.6 prikazane su ostale generacije populacija vektora $v_{i,G}$, $u_{i,G}$ i $x_{i,G}$, gdje je G generacija prikazana s lijeve strane tablice.

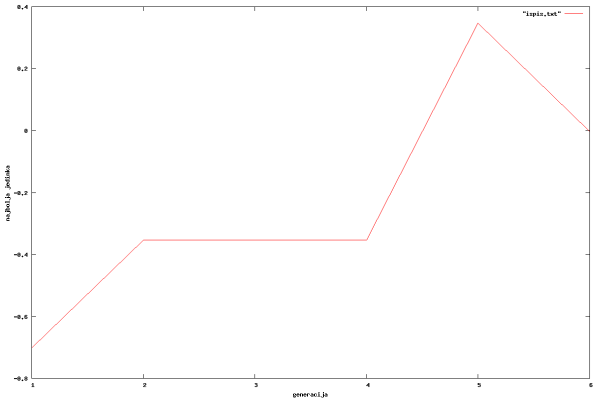
Tablica 3.6 Kompletnih ostalih generacije

	i	r1	r2	r3	**Populacija v**	**Populacija u**	f(u)	**Populacija x**	f(x)
4. generacija	1	1	3	4	**-1.836547**	**-1.836547**	3.372905	**-0.352783**	0.124456
	2	2	1	3	**1.480713**	**1.480713**	2.192511	**1.130981**	1.279118
	3	4	3	1	**0.431517**	**0.431517**	0.186207	**0.431517**	0.186207
	4	2	3	1	**0.781249**	**0.781249**	0.610350	**0.781249**	0.610350
	i	r1	r2	r3	**Populacija v**	**Populacija u**	f(u)	**Populacija x**	f(x)
5. generacija	1	1	3	4	**-0.702515**	**-0.702515**	0.493527	**-0.352783**	0.124456
	2	2	1	3	**0.346681**	**0.346681**	0.120188	**0.346681**	0.120188
	3	4	3	1	**1.565549**	**1.565549**	2.450944	**0.431517**	0.186207
	4	2	3	1	**1.915281**	**1.915281**	3.668301	**0.781249**	0.610350
	i	r1	r2	r3	**Populacija v**	**Populacija u**	f(u)	**Populacija x**	f(x)
6. generacija	1	1	3	4	**-0.702515**	**-0.702515**	0.493527	**-0.352783**	0.124456
	2	2	3	4	**-0.003051**	**-0.003051**	0.000009	**-0.003051**	0.000009
	3	4	3	1	**1.565549**	**1.565549**	2.450944	**0.431517**	0.186207
	4	2	1	3	**-0.437619**	**-0.437619**	0.191510	**-0.437619**	0.191510

U petoj iteraciji (šesta generacija) dolazi do ispunjenja uvjeta zaustavljanja algoritma, jer postoji barem jedan *f*(*x*) koji je manji od 0.01. Globalni optimum je pronađen u točci T(-0.003051, 0.000009). Na slici 3.1 se može vidjeti dobrota najbolje jedinke kroz generacije, a na slici 3.2 se može vidjeti najbolja jedinka kroz generacije.



Slika 3.1 Dobrota najbolje jedinke kroz generacije



Slika 3.2 Najbolja jedinka kroz generacije

U ovom slučaju, rezultat bi bio identičan onome kada bi uvjet zaustavljanja bio broj generacija (6 generacija).

4. Programska rješenja

Na stranici [3] se nalaze mnoga programska rješenja u raznim programskim jezicima.

4.1 DE - Applet

Ovaj applet rješava polinomni problem uklapanja (eng. *polynomial fitting problem*). Cilj je napraviti polinom određenog stupnja tako da se uklopi u dio grafa omeđen s dvjema crvenim linijama (eng. *tolerance scheme*), uz neku toleranciju (eng. *cost-value*). Funkcija cijene (eng. *cost function*) je suma kvadrata pogrešaka te treba biti minimalna i ona predstavlja uvjet zaustavljanja algoritma.

Postoje dvije varijante T4 i T8. Tx označava Tchebychev polinom prve vrste x-tog stupnja. Za dotični program to predstavlja *y* koordinatu crvene linije u točci $x = \pm 1.2$ koja iznosi 6.07 za T4(x) i 72.66 za T8(x). Vrijednosti se izračunaju iz slijedećih formula:

$$\begin{aligned} T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_8(x) &= 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1 \end{aligned} \tag{1.11}$$

Budući da T8 ima veće ograničenje od T4, program će se duže izvoditi.

Postoji 10 metoda:

- DE/best/1/exp
- DE/rand/1/exp
- DE/rand-to-best/1/exp
- DE/best/2/exp
- DE/rand/2/exp
- DE/best/1/bin
- DE/rand/1/bin
- DE/rand-to-best/1/bin
- DE/best/2/bin
- DE/rand/2/bin

Exp označava varijantu eksponencijalnog križanja, dok bin označava varijantu binomijalnog križanja.

4.2 Rezultati izvođenja DE – appleta

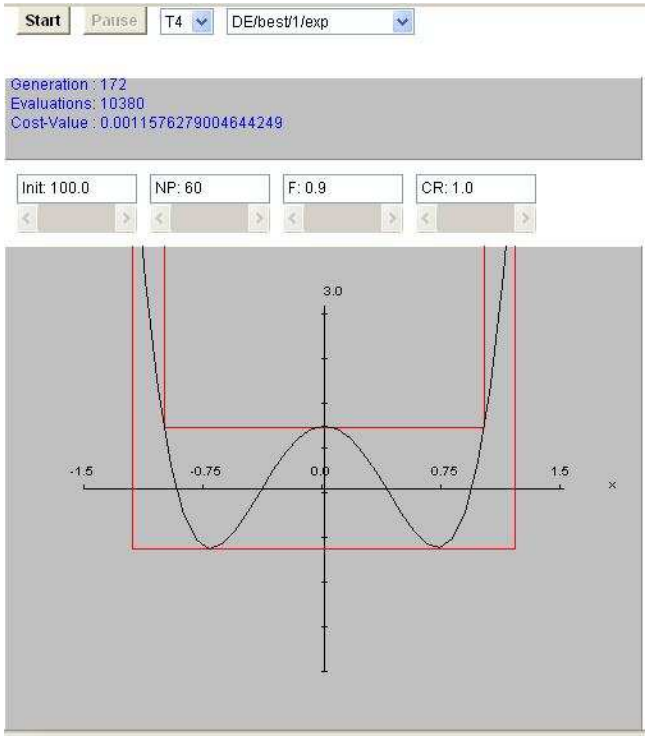
Odabrana je metoda DE/best/1/exp. U svim primjerima korišteni su:

$$\begin{aligned} Init &= 100 \\ CR &= 1 \end{aligned} \tag{1.12}$$

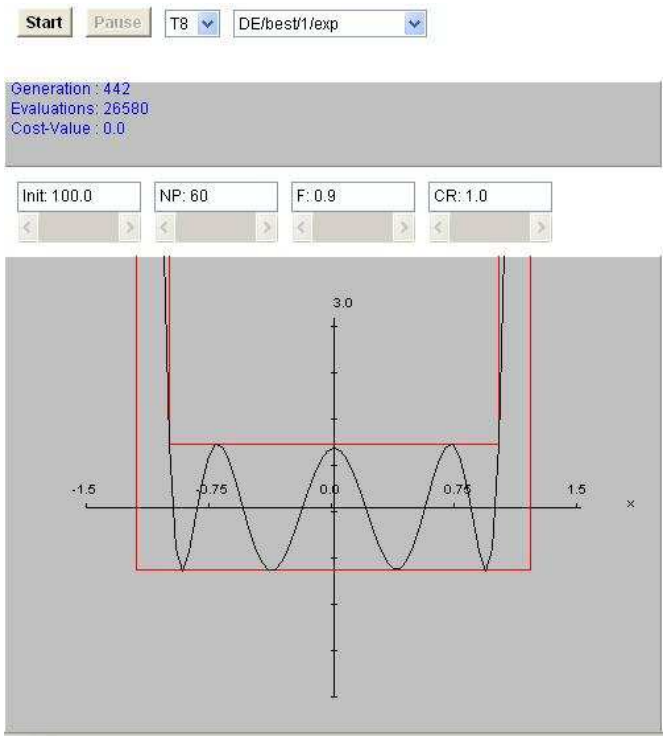
jer manja vrijednost *CR*-a povećava broj generacija potrebnih da ispune uvjet zaustavljanja tj. vrijeme izvođenja, ali daje robusnost. *Init* označava interval [-*Init*, *Init*] u kojem će se izabrati početne vrijednosti.

U svim primjerima program se izvršio do kraja, tj. vrijedi:

$$cost\text{-}value = 0 \tag{1.13}$$



Slika 4.1 Prikaz rezultata izvođenja appleta s opcijom T4



Slika 4.2 Prikaz rezultata izvođenja appleta s opcijom T8

Na slici 4.1 je odabrana opcija T4, a na slici 4.2 je odabrana opcija T8. Vidljivo je da je potreban polinom većeg stupnja za opciju T8 u odnosu na opciju T4, da bi se riješio problem.

U tablici 4.1 i na slikama 4.1 i 4.2, pokazani su rezultati s uobičajenim postavkama koje generira sam program, tj.

$$\begin{aligned} Np &= 60 \\ F &= 0.9 \end{aligned} \tag{1.14}$$

Tablica 4.1 Rezultati izvođenja uz Np=60 i F=0.9

	T4				T8				Prosjek - T4	Prosjek - T8
Generacije	139	126	150	166	362	442	352	475	145.25	407.75
Evaluacije	8400	7620	9120	10020	21780	26580	21180	28620	8790	24540

Iz tablice 4.1 vidljivo je da je broj generacija i evaluacija trostruko veći za opciju T8 nego za opciju T4.

U tablici 4.2 pokazani su rezultati s slijedećim postavkama

$$Np = 60$$
$$F = 0.5 \tag{1.15}$$

U odnosu na uobičajene postavke smanjen je mutacijski faktor *F*.

Tablica 4.2 Rezultati izvođenja uz Np=60 i F=0.5

	T4				T8				Prosjek - T4	Prosjek - T8
Generacije	217	188	185	143	1976	2075	1516	535	183.25	1525.5
Evaluacije	13080	11340	11100	8640	118620	124560	91020	32160	11040	91590

Iz tablice 4.2 vidljivo je da je broj generacija i evaluacija devet puta veći za opciju T8 nego za opciju T4. Usporedbom rezultata iz tablice 4.1 i 4.2, vidljiv je porast broja potrebnih generacija i evaluacija kada je *F*=0.5, nego kada je *F*=0.9. To pogotovo vrijedi za opciju T8 gdje je taj omjer četiri puta veći. U tablici 4.2 se također vidi i jedan rezultat koji se izdvaja od drugih po svojoj maloj vrijednosti. Ponekad se početni uvjeti inicijaliziraju tako da su već blizu krajnjem rješenju. Zbog toga su mogući takvi rezultati.

U tablici 4.3 pokazani su rezultati s slijedećim postavkama

$$Np = 120$$
$$F = 0.9 \tag{1.16}$$

U odnosu na uobičajene postavke povećana je populacija *Np*.

Tablica 4.3 Rezultati izvođenja uz Np=120 i F=0.9

	T4				T8				Prosjek - T4	Prosjek - T8
Generacije	145	171	156	160	673	538	623	457	158	572.75
Evaluacije	17520	20520	18720	19200	80880	64560	74880	54840	18990	68790

Iz tablice 4.3 vidljivo je da je broj generacija i evaluacija 3.6 puta veći za opciju T8 nego za opciju T4. Usporedbom rezultata iz tablice 4.1 i 4.3, vidljiv je da blagi porast broja generacija i dvostruki (T4) i trostruki (T8) porast broja evaluacija. Slijedeća relacija između broja generacija, broja evaluacija i populacije objašnjava rezultate:

$$Evaluacije = Generacije * Np \tag{1.17}$$

Usporedbom rezultata iz tablice 4.2 i 4.3, vidljivo je da promjena vrijednosti mutacijskog faktora jače utječe na rezultate nego promjena vrijednosti populacije.

4.3 Programsko ostvarenje

Program traži globalni ekstrem zadane funkcije dobrote, tj. minimum ili maksimum funkcije. Funkcija ima *D* neovisnih varijabli i eksplicitnog je oblika.

Primjeri funkcija:

$$\begin{aligned} f(x) &= x \wedge 2 * \sin(x) \\ f(x, y, z) &= x * y / z \end{aligned} \tag{1.18}$$

Dovoljno je zadati desnu stranu jednadžbe, jer se lijeva strana jednadžbe podrazumijeva.

Funkcije se u program zapišu u obliku:

$$\begin{aligned} & x[0] * x[0] * \sin x[0] \text{ ili } pow(x[0], 2) * \sin x[0] \\ & x[0] * x[1] / x[2] \end{aligned} \tag{1.19}$$

Prva funkcija ima jednu (*D*=1) neovisnu varijablu *x*, dok druga funkcija ima tri (*D*=3) neovisne varijable *x, y* i *z*. Primjećuje se da je najveći indeks varijable u programu uvijek jednak *D*-1. Ako to nije slučaj, program neće dobro raditi.

4.3.1 Odabir funkcije dobrote

Moguće su sve funkcije koje se nalaze u standardnoj biblioteci "math.h". Funkcija se napiše u program na mjestu #define funkcija ____ u gore navedenom obliku. Trenutno se tu nalazi funkcija $x[0] * x[0]$.

4.3.2 Macro opcije

U programu se nalazi pet macro opcija koje određuju oblik i mjesto ispisa, način primanja argumenata i optimizaciju koda.

Ako je definiran macro ISPIS_U_DAT, tada se cjelokupni ispis zapisuje u izlaznu datoteku. Inače se ispis ispisuje u stdout (koji je, uobičajeno, ekran).

Ako je definiran macro ISPIS_ZA_GNUPLOT, tada je ispis u formatu koji gnuplot može koristiti, tj. broj_generacije najbolja_jedinka. Inače program ispiše sve relevantne podatke (cijele populacije korištenih vektora i indekse korištene za tvorbu vektora $v_{i,G}$).

Ako je definiran macro PRIKAZ_DOBROTE, tada se uz parametre vektora ispiše vrijednost funkcije dobrote. U suprotnom, se dotični podatak ne ispiše.

Ako je definiran macro OPTIMIZIRAN_KOD, tada se ne koristi vektor $v_{i,G}$ te se smanjuje broj evaluacija. U suprotnom, program radi kao u teoriji (poglavlje 2).

Ako je definiran macro CMD, tada program prima argumente preko komandne linije. Inače ih čita iz ulazne datoteke.

Trenutno su u programu omogućeni macroi ISPIS_ZA_GNUPLOT, OPTIMIZIRAN_KOD, PRIKAZ_DOBROTE i ISPIS_U_DAT, dok je macro CMD onemogućen.

4.3.3 Preporučeni programi

Rješenja problema ispisuju se u odgovarajuću izlaznu datoteku, a preporuča se grafička predodžba i analiza pomoću programskog paketa *gnuplot*, kojeg se besplatno može preuzeti sa http://www.gnuplot.info/. Naravno, u programu mora biti omogućena macro opcija ISPIS_ZA_GNUPLOT, inače gnuplot neće moći nacrtati sliku. Za korisnike operacijskog sustava Windows, preporuča se instalacija na c disk, inače će biti potrebno modificiranje puta u datotekama "nacrtaj funkciju.bat", "nacrtaj graf_vr_dobrote.bat" i "nacrtaj graf.bat".

Za korisnike operacijskog sustava Unix, preporuča se prevoditelj *gcc*, kojeg se besplatno može preuzeti sa http://gcc.gnu.org/. Za korisnike operacijskog sustava Windows, preporuča se prevoditelj *djgpp*, kojeg se besplatno može preuzeti sa http://www.delorie.com/djgpp/. Ako su ti uvjeti zadovoljeni, prevođenje se lako obavi pokretanjem "compile.bat" u konzoli. Napomena: gore navedeni prevoditelji su preporučljivi, iako je dovoljan bilo koji prevoditelj programskog jezika c.

4.3.4 Praktični savjeti odabira ulaznih parametara

Veličina populacije neka bude 5 do 10 puta veća od broja parametara u vektoru:

$$Np \in [5*D, 10*D] \tag{1.20}$$

$F \in [0.4, 1]$ je jako učinkovit raspon. Za brzo rješenje, CR neka bude između 0.9 i 1. [5]

Ako je odabran interval u kojem se sigurno nalazi rješenje, onda je bolje odabrati manja vrijednost mutacijskog faktora jer povećava vjerojatnost konvergencije.

4.4 Rezultati izvođenja programa

Slijede slike izvođenja programa zajedno sa odabirom postavki i funkcijom dobrote.

4.4.1 Primjer 1

Traži se globalni minimum funkcije $f(x) = x^2$.

Postavke:

$$\begin{aligned} &F = 0.9; D = 1; Np = 8; CR = 0.9; stop = 100; ekstrem = 1; \\ &dg[0] = 400; gg[0] = 500 \end{aligned} \tag{1.21}$$

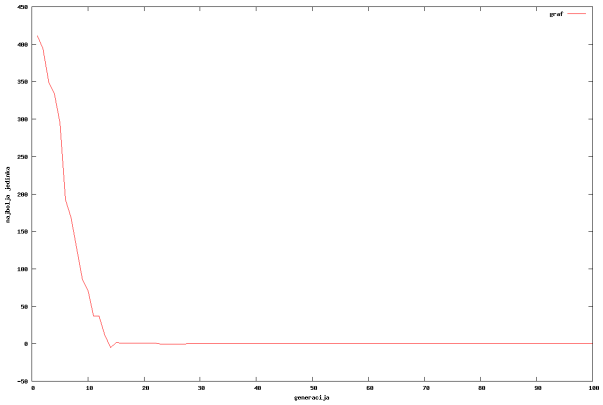
Primjećuje se da je inicijalni interval ne sadrži točku u kojoj se nalazi rješenje. Ipak, program uspije doći do ispravnog rješenja i to zbog mutacije koja proširuje prostor potrage rješenja.

Ispis je oblika: 1 411.2882038631 169157.986636931, gdje je:

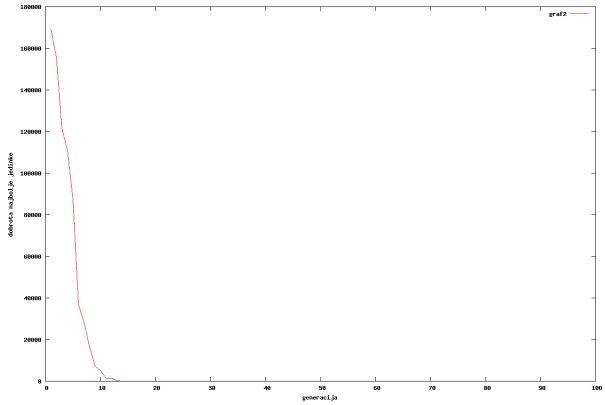
```
generacija=1
najbolja_jedinka=411.2882038631
dobrota_najbolje_jedinke=169157.986636931
```
(1.22)

Budući da je *stop*=100, ispisat će se sto generacija.

Na slikama 4.3 i 4.4 su grafovi napravljeni pomoću programa gnuplot i gore navedenog ispisa. Na slici 4.3 uzete su vrijednosti prvog i drugog stupca ispisa, a na slici 4.4 vrijednosti prvog i zadnjeg (trećeg) stupca ispisa. Budući da se traži globalni minumum, graf na slici 4.4 mora biti padajući.



Slika 4.3 Najbolja jedinka kroz generacije



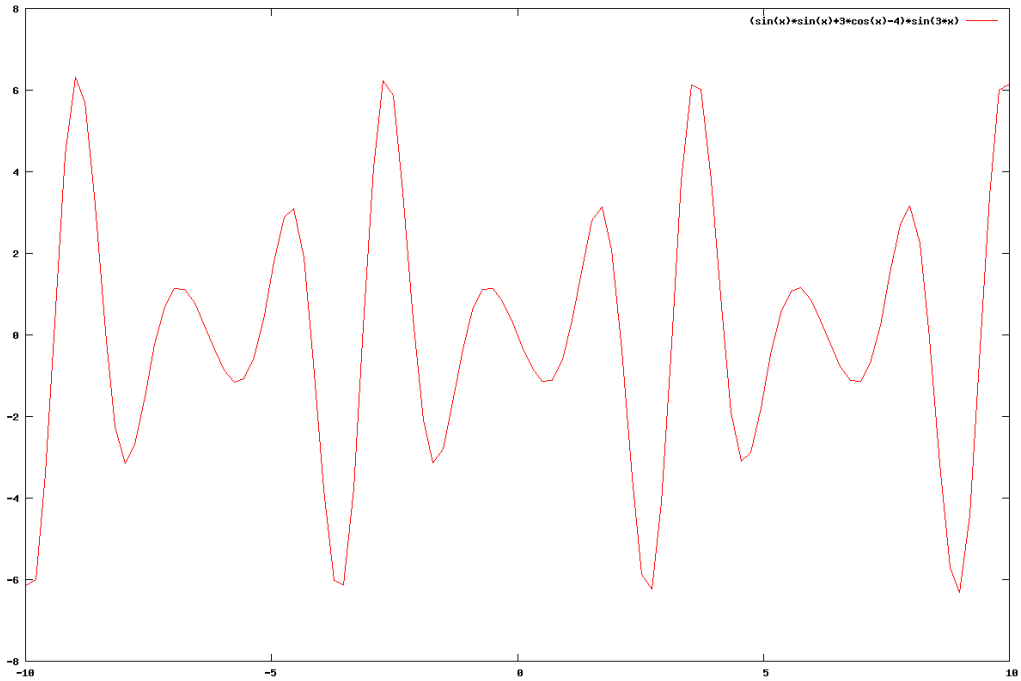
Slika 4.4 Dobrota najbolje jedinke kroz generacije

4.4.2 Primjer 2

Traži se globalni minimum funkcije $f(x) = (\sin(x)^2 + 3*\cos(x) - 4)*\sin(3*x)$.

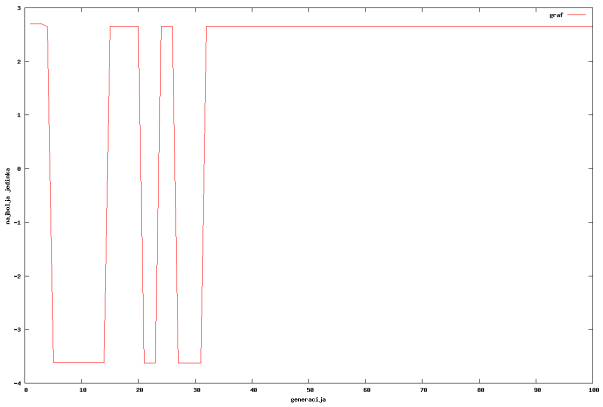
Postavke:

$$F = 0.9; D = 1; Np = 8; CR = 0.9; stop = 100; ekstrem = 1;$$
$$dg[0] = 0; gg[0] = 6.28 \qquad (1.23)$$

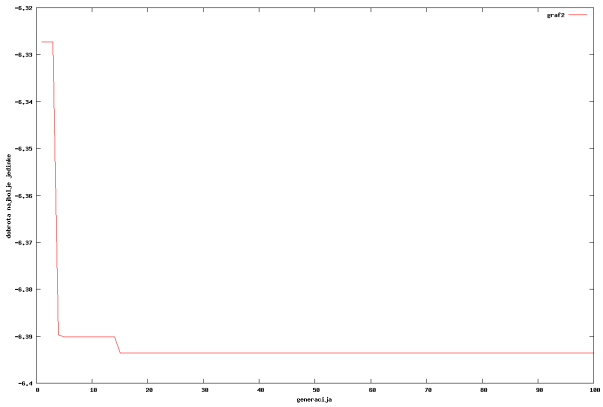


Slika 4.5 Funkcija kojoj se traži minumum

Na slici 4.5 je vidljivo da funkcija ima period $T = 2*\pi$. Ispis je istog oblika kao i u prethodnom primjeru. Na slici 4.6 su vidljivi skokovi koji sugeriraju postojanje više lokalnih ekstrema. Nakon generacije 33 više nema skokova što sugerira pronalazak rješenja. Na slici 4.7 je vidljivo stagniranje nakon generacije 15, iako na slici 4.6 još postoje skokovi. Razlog tome je vrlo mali popravak rješenja koji se ne vidi na toj skali.



Slika 4.6 Najbolja jedinka kroz generacije



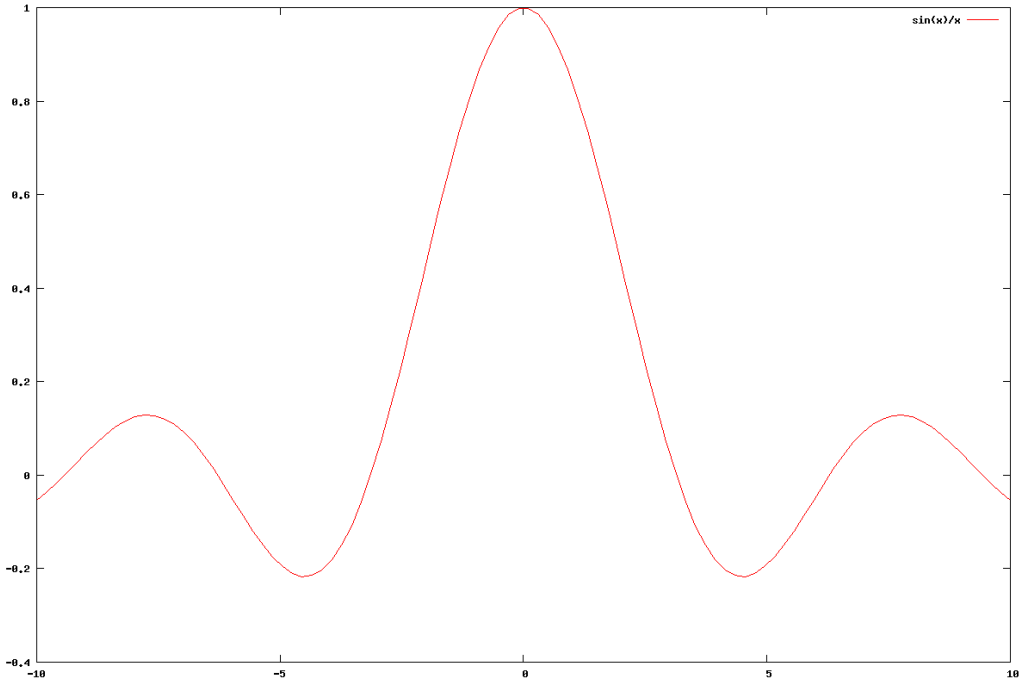
Slika 4.7 Dobrota najbolje jedinke kroz generacije

4.4.3 Primjer 3

Traži se globalni maksimum funkcije $f(x) = \sin(x)/x$.

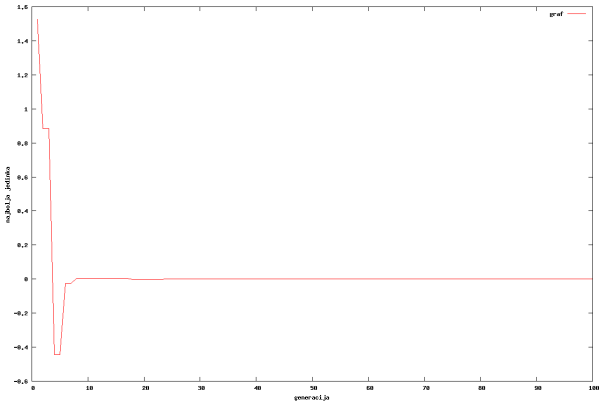
Postavke:

$$F = 0.9; D = 1; Np = 8; CR = 0.9; stop = 100; ekstrem = -1;$$
$$dg[0] = -5; gg[0] = 5 \tag{1.24}$$

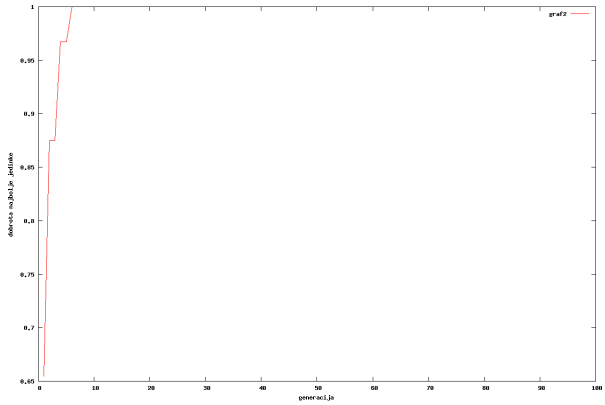


Slika 4.8 Funkcija kojoj se traži maksimum

Na slici 4.8 je vidljivo da funkcija ima mnogo lokalnih ekstrema. Ispis je istog oblika kao i u prvom primjeru. Na slikama 4.9 i 4.10 je vidljivo da dolazak do rješenja nije trajao više od deset generacija. Razlog tome je dobar odabir inicijalnog prostora potrage rješenja. Budući da se traži globalni maksimum, graf na slici 4.10 mora biti rastući.



Slika 4.9 Najbolja jedinka kroz generacije



Slika 4.10 Dobrota najbolje jedinke kroz generacije

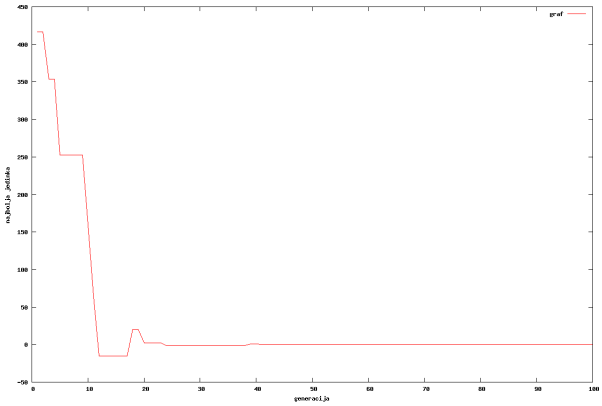
4.4.4 Primjer 4

Isto kao i u primjeru 3, traži se globalni maksimum funkcije $f(x) = \sin(x)/x$.

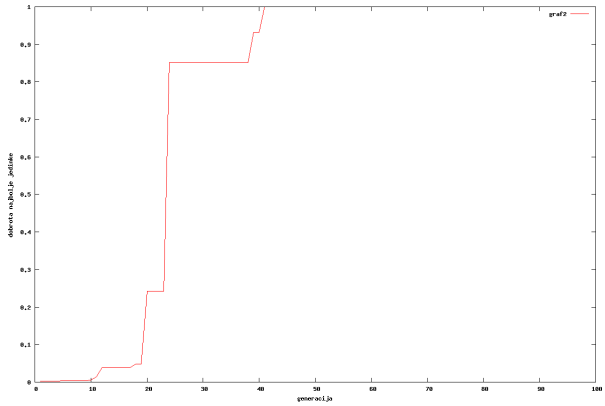
Postavke su iste kao i u primjeru 3, osim inicijalizacijskog intervala:

$$F = 0.9; D = 1; Np = 8; CR = 0.9; stop = 100; ekstrem = -1;$$
$$dg[0] = 400; gg[0] = 500 \tag{1.25}$$

Ispis je istog oblika kao i u prvom primjeru. Na slikama 4.11 i 4.12 je vidljivo da dolazak do rješenja je trajao više od četrdeset generacija. Razlog tome je loš odabir inicijalnog prostora potrage rješenja. Rezultati bi bili još gori, ako bi mutacijski faktor bio manji, jer bi vjerojatnost zaustavljanja u lokalnom ekstremu bila veća. Budući da se traži globalni maksimum, graf na slici 4.12 mora biti rastući.



Slika 4.11 Najbolja jedinka kroz generacije



Slika 4.12 Dobrota najbolje jedinke kroz generacije

5. Zaključak

Ne postoji dokaz konvergencije DE, ali u praksi je djelotvoran za mnoge optimizacijske probleme. Storn i Price (1997) su usporedbom pokazali da je učinkovitiji od simuliranog kaljenja i genetskih algoritama. Ali i Torn (2004) su otkrili da je DE točniji i učinkovitiji od kontrolirane slučajne potrage i drugog genetskog algoritma. 2004. godine Lampinen i Storn su pokazali da je DE točniji (eng. *accurate*) od nekoliko drugih optimizacijskih metoda uključujući genetske algoritme, simulirano kaljenje i evolucijsko programiranje. [1]

5.1 Prednosti:

- dostupnost za praktične aplikacije
- jednostavna struktura
- lakoća uporabe
- brzina dolaska do rješenja
- otpornost na manja odstupanja (robusnost)

5.2 Nedostaci:

Manji problemi nastaju kod:

- nediferencijabilnih funkcija u nekim točkama
- funkcija s mnogo lokalnih ekstrema, koje su djelomično okružene ravnom plohom

U takvim slučajevima, najbolje rješenje ostaje izvan prostora potrage rješenja.

6. Literatura

[1] Kelly Fleetwood – An Introduction to Differential Evolution, http://www.maths.uq.edu.au/MASCOS/Multi-Agent04/Fleetwood.pdf, 25.10.2008

[2] David Craft – Differential Evolution: a stochastic nonlinear optimization algorithm by Storn and Price, http://www.technorati.com/search/ses2_storn_price.pdf, 25.10.2008

[3] http://www.icsi.berkeley.edu/~storn/code.html, 25.10.2008

[4] Projekt 2007/2008: Evolucijski algoritmi – Evolucijske strategije, http://www.zemris.fer.hr/~golub/ga/studenti/projekt2007/es.html, 25.10.2008

[5] Napapan Piyasatian – Differential Evolution: A Simple Evolution Strategy for Fast Optimization, http://www-personal.une.edu.au/~jvanderw/DE_1.pdf, 25.10.2008