Elektrotehniški vestnik 74(1-2): 55–60, 2007
Electrotechnical Review, Ljubljana, Slovenija

Velikost populacije pri algoritmu diferencialne evolucije

Janez Brest, Viljem Žumer, Mirjam Sepesy Maučec

Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, 2000 Maribor, Slovenija
E-pošta: janez.brest@uni-mb.si

Povzetek. V članku obravnavamo algoritem diferencialne evolucije (DE) za numerično optimizacijo. Algoritem DE je v zadnjem desetletju postal popularen in ga uporabljamo v številnih praktičnih aplikacijah na področjih optimizacije. Algoritem ima tri kontrolne parametre (F - faktor skaliranja, CR - kontrolni parameter križanja, NP - velikost populacije). Izbira ustreznih vrednosti kontrolnih parametrov je ponavadi odvisna od problema, ki ga rešujemo, in od uporabnika zahteva predhodno znanje oziroma izkušnje. Čeprav je pri algoritmu DE izbira parametrov zelo pomembna, ni nobene metodologije za določitev njihovih vrednosti. V članku se bomo omejili le na parameter velikosti populacije. Analizirali bomo vpliv velikosti populacije na uspešnost algoritma DE pri reševanju optimizacijskih problemov z omejitvami.

Ključne besede: evolucijsko računanje, optimizacija, iskanje globalnega optimuma, velikost populacije

Population Size in Differential Evolution Algorithm

Extended Abstract. In this paper we present an experimental analysis showing that the population size NP is an important control parameter in the differential evolution algorithm.

Differential Evolution (DE) [16, 17, 15, 11, 12, 10, 18] has been proven to be a powerful evolutionary algorithm for global optimization in many real problems [13, 14]. Although the DE algorithm has been proven to be a simple yet powerful evolutionary algorithm for optimizing continuous functions, users are still faced with the problem of preliminary testing and hand-tuning of the evolutionary parameters prior to commencing the actual optimization process [18].

Different problems usually require different settings for the control parameters. Self-adaptation allows an evolution strategy to adapt itself to any general class of problems by reconfiguring itself accordingly and without any user interaction [1, 2, 7]. Based on the experiment in [5], the necessity of changing control parameters during the optimization process is also confirmed. In literature, self-adaptation is usually applied to the F and CR control parameters [5, 4], where F is a scaling factor and CR is a crossover rate.

In our previous paper [6], a performance of the self-adaptive differential evolution algorithm is evaluated on a set of 24 benchmark functions provided for the CEC2006 Special session on constrained real-parameter optimization [9]. The method in [6] extends individuals that have not only decision variables but also control parameters F and CR. These parameters are changed/optimized by DE, too. The authors utilize lexicographic ordering in which the constraint violation precedes the objective function to solve constrained problems. In [6] the control parameter NP is set to 200.

The number of evaluations of an evolutionary algorithm is computed as a product of NP and the number of generations. The third control parameter NP gets usually less attention in literature, mainly because it affects only the finish time of the evolutionary process when the number of generations is fixed.

The scenario is quite different when the number of evaluations is fixed (an algorithm stops after the number of performed evaluations has reached the predetermined value). The question is how to determine NP to get the best results.

In this paper experimental results of a self-adaptive differential evolution algorithm are evaluated on the set of 24 benchmark functions provided for the CEC2006 Special session on constrained real-parameter optimization. We especially focus on the third DE control parameter, namely NP.

The results confirm our assumption that NP is an important parameter of the DE algorithm. For a set of benchmark functions the best value of the population size seems to be about 120. The optimal value depends on the nature of the problem being solved.

The results show that it would be reasonable to include the NP parameter into the adaptation or even self-adaptation scheme as well.

Key words: evolutionary computation, optimization method, global optimum search, population size

Prejet 25. avgust, 2006
Odobren 24. januar, 2007

1 Uvod

V članku predstavljamo algoritem diferencialne evolucije (DE) (ang. *Differential Evolution*) [17] za numerično optimizacijo funkcij. Algoritem DE je v zadnjih letih postal popularen in ga uporabljamo v številnih praktičnih aplikacijah, predvsem zato, ker algoritem odlikuje dobra konvergenca [16, 17, 15, 11, 12, 10, 18, 5].

Za optimizacijo računsko težkih problemov se danes uporabljajo različne tehnike, kot so simulirano ohlajanje, nevronske mreže, optimizacija s pomočjo roja delcev

(ang. *Partical Swarm Optimization* - PSO) in različne evolucijske tehnike.

Algoritem DE lahko prištevamo k evolucijskim algoritmom, ki imajo nekatere prednosti v primerjavi z drugimi metodami. Evolucijski algoritmi potrebujejo le funkcijo, ki jo želimo optimirati. Ponavadi govorimo o iskanju globalnega minimuma.

Izbira ustreznih kontrolnih parametrov je ponavadi odvisna od problema, ki ga rešujemo, in od uporabnika zahteva predhodno znanje oziroma izkušnje. Čeprav je izbira parametrov zelo pomembna, ni nobene metodologije za določitev njihovih "optimalnih" vrednosti.

Pri optimizaciji funkcije je cilj poiskati vektor $\mathbf{x}_{min}$, za katerega velja: $\forall \mathbf{x}$, $f(\mathbf{x}_{min}) \leq f(\mathbf{x})$. Funkcija f ni treba, da je zvezna ali odvedljiva, mora pa biti navzdol omejena.

Algoritem DE ima tri kontrolne parametre (F - faktor skaliranja, CR - kontrolni parameter križanja, NP - velikost populacije), ki se pri originalnem algoritmu DE med optimizacijskim postopkom ne spreminjajo [16, 12, 11].

Mehanizem samoprilagajanja omogoča, da se evolucijska strategija (sama) prilagodi naravi problema, tako da se ustrezno rekonfigurira. Ta prilagoditev se opravi brez aktivne vloge uporabnika [1, 2, 7]. Eksperimenti v [5, 4, 3] so pokazali, da s spreminjanjem kontrolnih parametrov med optimizacijskim postopkom lahko izboljšamo rezultate. V literaturi zasledimo, da avtorji ponavadi uporabljajo samoprilagajanje pri kontrolnih parametrih F in CR [12, 11, 5].

V našem predhodnem prispevku [6] smo opravili študijo zmogljivosti samoprilagodljivega algoritma diferencialne evolucije na testnih funkcijah, ki so bile posebej izbrane za *CEC2006 Special session on constrained real parameter optimization* [9]. Metoda, ki smo jo predstavili pri reševanju optimizacijskih problemov z omejitvami, uporablja leksikografsko ureditev, pri kateri je dopustnost rešitve pomembnejša od njene ocenitvene vrednosti. V omenjenem članku smo uporabljali samoprilagajanje kontrolnih parametrov F in CR, vrednosti kontrolnega parametra NP pa nismo spreminjali. Nastavili smo ga na 200. V tem članku želimo ugotoviti vpliv kontrolnega parametra NP na kakovost rešitve pri optimizacijah z omejitvami.

Število ovrednotenj $nevals$ evolucijskega algoritma izračunamo kot produkt NP in števila generacij. Tretji kontrolni parameter NP ponavadi ni zbudil pozornosti pri raziskovalcih, ki uporabljajo algoritem DE. Po našem mnenju je razlog v tem, da omenjeni kontrolni parameter vpliva le na trajanje evolucijskega postopka, ko fiksiramo število generacij. Včasih raziskovalci primerjajo uspešnost algoritmov le pri vnaprej izbranem številu generacij.

Popolnoma drugačen scenarij nastopi, ko fiksiramo $nevals$ (algoritem ustavimo, ko doseže vnaprej predpisano fiksno število ovrednotenj). Poraja se vprašanje, kako naj določimo/izberemo velikost populacije NP, da bo algoritem dajal najboljše rezultate.

V tem članku prikazujemo eksperimentalne rezultate samoprilagodljivega algoritma diferencialne evolucije na 24 testnih funkcijah [9], s poudarkom na velikosti populacije NP. Samih testnih funkcij v tem članku ne prikazujemo, ker so preobsežne.

Nadaljevanje prispevka je organizirano takole: V 2. poglavju opišemo algoritem diferencialne evolucije. 3. poglavje govori o optimizaciji z omejitvami in kako omejitve vključimo v algoritem DE. V 4. poglavju predstavimo naš samoprilagodljivi algoritem DE. Rezultati, ki smo jih dobili s pomočjo samoprilagodljivega algoritma na optimizacijskih primerkih, so prikazani v 5. poglavju. V 6. poglavju podajamo razpravo o kontrolnem parametru NP. Sledi še sklepno, 7. poglavje, kjer podamo tudi smernice za nadaljnje raziskave.

2 Algoritem diferencialne evolucije

Algoritem diferencialne evolucije (DE) ustvari novega posameznika s kombinacijo starša in izbranih posameznikov iste populacije. Nov posameznik zamenja starša, če ima boljšo ocenitveno vrednost.

Pri algoritmu DE [16, 17, 15] populacija vsebuje NP D-dimenzionalnih vektorjev oziroma posameznikov:

$$\mathbf{x}_{i,G} = (x_{i,1,G}, x_{i,2,G}, ..., x_{i,D,G}),\ i = 1, 2, ..., NP.$$

Z G označimo generacijo. V eni generaciji algoritem DE uporabi mutacijo in križanje na vsakem vektorju $\mathbf{x}_{i,G}$ in ustvari nov vektor (novega posameznika):

$$\mathbf{u}_{i,G} = (u_{i,1,G}, u_{i,2,G}, ..., u_{i,D,G}),\ i = 1, 2, ..., NP.$$

V eni generaciji algoritem DE ustvari NP novih posameznikov. Nato uporabi selekcijo, da izbere vektorje za naslednjo generacijo ($G + 1$).

Začetna populacija je izbrana naključno (po enakomerni porazdelitvi) med spodnjo ($x_{j,low}$) in zgornjo ($x_{j,upp}$) mejo, ki sta definirani za vsako spremenljivko x_j. Spodnjo in zgornjo mejo definira uporabnik glede na naravo problema, ki ga rešuje. Po inicializaciji algoritem DE izvede več transformacij (operacij) v postopku, ki ga imenujemo evolucija.

2.1 Mutacija

Mutacija za vsak vektor iz populacije ustvari *mutiran* vektor:

$$\mathbf{x}_{i,G} \Rightarrow \mathbf{v}_{i,G} = (v_{i,1,G}, v_{i,2,G}, ..., v_{i,D,G}),$$

$$i = 1, 2, ..., NP.$$

Algoritem DE ustvari mutiran vektor z uporabo ene izmed strategij mutacije. Originalni algoritem DE pozna naslednje strategije:

1. "rand/1": $\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G})$
2. "best/1": $\mathbf{v}_{i,G} = \mathbf{x}_{best,G} + F \cdot (\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G})$
3. "current to best/1": $\mathbf{v}_{i,G} = \mathbf{x}_{i,G} + F \cdot (\mathbf{x}_{best,G} - \mathbf{x}_{i,G}) + F \cdot (\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G})$
4. "best/2": $\mathbf{v}_{i,G} = \mathbf{x}_{best,G} + F \cdot (\mathbf{x}_{r_1,G} - \mathbf{x}_{r_2,G}) + F \cdot (\mathbf{x}_{r_3,G} - \mathbf{x}_{r_4,G})$
5. "rand/2": $\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F \cdot (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) + F \cdot (\mathbf{x}_{r_4,G} - \mathbf{x}_{r_5,G})$

Indeksi r_1, r_2, r_3, r_4, r_5 predstavljajo naključna in paroma različna naravna števila na intervalu $[1, NP]$. Indeksi so različni tudi od indeksa i. F je mutacijski skalirni faktor (realno število) na intervalu $[0, 2]$, ponavadi manjši kot 1. $\mathbf{x}_{best,G}$ je vektor z najboljšo ocenitveno vrednostjo oziroma najboljši posameznik generacije G.

2.2 Križanje

Po končani mutaciji sledi križanje, ki iz posameznika i in pripadajočega mutiranega vektorja ustvari novega posameznika i:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{če } rand(0,1) \leq CR \text{ ali } j = j_{rand}, \\ x_{i,j,G} & \text{sicer} \end{cases}$$

$$i = 1, 2, ..., NP \text{ in } j = 1, 2, ..., D.$$

CR je kontrolni parameter križanja ali faktor na intervalu $[0, 1)$ in pomeni verjetnost, da se komponenta vektorja pri novem posamezniku ustvari iz komponente mutiranega vektorja. Indeks j_{rand} je naključno izbrano naravno število na intervalu $[1, NP]$. Njegova naloga je, da je vsaj ena komponenta vektorja pri novem posamezniku izbrana iz mutiranega vektorja.

2.3 Selekcija

Glede na ocenitveno vrednost posameznika iz populacije in ocenitveno vrednost pripadajočega novega pozameznika z operacijo selekcije izberemo, kateri od obeh omenjenih posemeznikov bo preživel in bo uvrščen v naslednjo generacijo. Če na primer iščemo globalni minimum, uporabimo za selekcijo naslednje pravilo:

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G} & \text{če } f(\mathbf{u}_{i,G}) \leq f(\mathbf{x}_{i,G}), \\ \mathbf{x}_{i,G} & \text{sicer.} \end{cases}$$

3 Omejitve

V zadnjih letih so bile na področju genetskih algoritmov razvite različne metode za obravnavanje omejitev pri optimizaciji. Avtorji [14, 8] so metode razvrstili v naslednje kategorije:

- *Metode, ki ohranjajo dopustne rešitve.* Ideja teh metod temelji na specializiranih operatorjih, ki transformirajo posameznike z dopustnimi rešitvami spet v posameznike z dopustnimi rešitvami. Metode predpostavljajo le linearne omejitve in dopustne startne točke v začetni populaciji.
- *Metode, ki temeljijo na kazenski funkciji.* Precej evolucijskih algoritmov vključuje pri obravnavi omejitev metode, ki temeljijo na principu zunanje kazenske funkcije, ki kaznuje nedopustne rešitve. Metode se razlikujejo v pomembnih podrobnostih, kako je kazenska funkcija načrtovana in kako jo uporabimo pri nedopustnih rešitvah.
- *Metode, ki razlikujejo med dopustnimi in nedopustnimi rešitvami.* Obstaja nekaj metod, ki poudarjajo razlikovanje med dopustnimi in nedopustnimi rešitvami preiskovalnega prostora. Ena od metod razlikuje med posameniki z dopustnimi in nedopustnimi rešitvami: za vsakega posameznika z dopustno rešitvijo $\mathbf{x}$ in za vsakega posameznika z nedopustno rešitvijo $\mathbf{y}$ velja: $f(\mathbf{x}) < f(\mathbf{y})$, t.j. vsak posameznik z dopustno rešitvijo je boljši od posameznika z nedopustno rešitvijo.
- *Druge hibridne metode.* Te metode združujejo tehnike evolucijskega računanja z determinističnimi postopki numerične optimizacije.

Algoritem DE ne potrebuje posebne razširitve, da bi ga lahko uporabili pri reševanju nalog z omejitvami [19]. Pri reševanju večine problemov z omejitvami lahko uporabimo kazensko funkcijo. Rešitev $\mathbf{x}$ je *dopustna*, če

$$g_i(\mathbf{x}) \leq 0, \text{ za } i = 1, ..., q, \text{ in}$$

$$|h_j(\mathbf{x})| - \epsilon \leq 0, \text{ za } j = q+1, ..., m,$$

kjer omejitve z enakostmi transformiramo v omejitve z neenakostmi. V okviru CEC2006 [9] je (bil) ϵ postavljen na $0,0001$. Povprečje kršitev omejitev (ang. *mean violations*) $\overline{v}$ je definirano takole:

$$\overline{v} = \frac{(\sum_{i=1}^{q} G_i(\mathbf{x}) + \sum_{j=q+1}^{m} H_j(\mathbf{x}))}{m}, \text{ kjer}$$

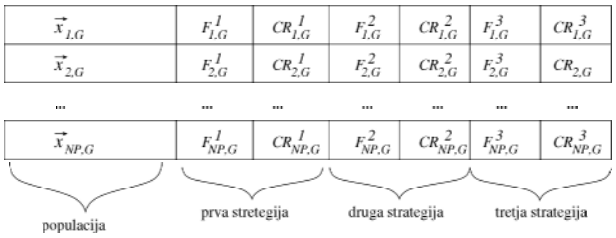
$$G_i(\mathbf{x}) = \begin{cases} g_i(\mathbf{x}) & \text{če } g_i(\mathbf{x}) > 0, \\ 0 & \text{če } g_i(\mathbf{x}) \leq 0, \end{cases}$$

$$H_j(\mathbf{x}) = \begin{cases} |h_j(\mathbf{x})| & \text{če } |h_j(\mathbf{x})| - \epsilon > 0, \\ 0 & \text{če } |h_j(\mathbf{x})| - \epsilon \leq 0. \end{cases}$$

Vsota vseh kršitev omejitev je enaka nič pri dopustnih rešitvah. Omenjena vsota je pozitivna, ko je kršena vsaj ena omejitev. Očitna uporaba kršitev omejitev je pri usmeritvi postopka iskanja proti dopustnim področjem v iskalnem prostoru. Precej raziskav je bilo narejenih na tem področju in bralcu priporočamo pregled tehnik za obravnavanje omejitev v knjigi avtorjev Michalewicz in Fogel [13].

4 Samoprilagodljivi algoritem diferencialne evolucije

V [5] je predstavljen samoprilagodljivi mehanizem, ki med evolucijo spreminja kontrolna parametra F in CR.



Slika 1. Samoprilagajanje, ko imamo tri strategije
Figure 1: Self-adapting: encoding aspect of three DE strategies

Slika 1 prikazuje, kako lahko kontrolna parametra treh originalnih strategij algoritma DE shranimo v vsakega posameznika. Vsaka strategija ima svoja kontrolna parametra. Brez težav lahko v naš algoritem vključimo še več strategij. Pri algoritmu v prispevku [5] je uporabljena le ena strategija, na sliki 1 pa prikazujemo razširitev samoprilagajanja na tri strategije.

Nove kontrolne parametre $F^k_{i,G+1}$ in $CR^k_{i,G+1}$, $k = 1, 2, 3$ izračunamo takole:

$$F^k_{i,G+1} = \begin{cases} F_l + rand_1 * F_u & \text{če } rand_2 < \tau_1, \\ F^k_{i,G} & \text{sicer,} \end{cases}$$

$$CR^k_{i,G+1} = \begin{cases} rand_3 & \text{če } rand_4 < \tau_2, \\ CR^k_{i,G} & \text{sicer.} \end{cases}$$

Tako dobimo kontrolna parametra F in CR pri novem posamezniku. k pomeni izbrano strategijo algoritma DE. Ko se ustvarja nov posameznik, je aktivna le ena strategija. V vsaki iteraciji se izbere ena strategija, ki je aktivna. $rand_j, j \in \{1, 2, 3, 4\}$ so naključne realne vrednosti na intervalu $[0, 1)$. τ_1 in τ_2 sta verjetnosti, s katerima uravnavamo kontrolna parametra F in CR. τ_1, τ_2, F_l, F_u imajo vnaprej določene fiksne vrednosti: $0,1; 0,1; 0,1; 0,9$. Nov F ima naključno vrednost na intervalu $[0,1; 1,0)$, nov CR pa na intervalu $[0, 1)$. Vrednosti $F_{i,G+1}$ in $CR_{i,G+1}$ sta izračunani pred mutacijo in vplivata na mutacijo, križanje in selekcijo novega posameznika $\mathbf{x}_{i,G+1}$.

Pri poskusih v članku [6] predstavljen algoritem jDE-2 uporablja naslednje tri strategije: "rand/1/bin", "current to best/1/bin" in "rand/2/bin". Prvi par samoprilagodljivih kontrolnih parametrov pripada strategiji "rand/1/bin", drugi par parametrov pripada strategiji "current to best/1/bin" itd. Velikost populacije NP je bila 200. Algoritem jDE-2 razlikuje med dopustnimi in nedopustnimi posamezniki: vsaka dopustna rešitev je boljša kot nedopustna rešitev. V članku [6] smo algoritem jDE-2 preizkusili na 24 testnih funkcijah, ki so jih pripravili za *CEC2006 Special session on constrained real parameter optimization* [9]. Naš algoritem je poiskal dopustno rešitev pri 22 funkcijah. Dopustne rešitve algoritem ni našel pri funkcijah $g20$ in $g22$. Za slednjo se je (pozneje) pokazalo, da ni rešljiva oziroma da nima nobene dopustne rešitve (glej končno verzijo tehničnega poročila [9]).

Algoritem jDE-2 [6] uporablja naslednjo tehniko, s katero zmanjšuje prehitro konvergenco posamenikov v lokalni optimum. V vsaki l-ti generaciji algoritem zamenja k najslabših posameznikov v populaciji s posamezniki, ki so naključno izbrani med spodnjo in zgornjo mejo. Pri tej zamenjavi ni ovrednotenj teh posameznikov. Nastavitvi za l in k sta bili 1000 in 70. Pridobili smo ju eksperimentalno.

V tem prispevku smo pri poskusih uporabili algoritem brez zgoraj opisane tehnike.

5 Eksperimentalni rezultati

V tem poglavju predstavimo rezultate, ki smo jih dobili pri poskusih, kjer smo izvedli zagon algoritma jDE-2, opisanega v prejšnjem poglavju.

Naš osnovni namen je bil ugotoviti, kako velikost populacije vpliva na kakovost rešitve našega samoprilagodljivega algoritma. Spomnimo naj, da je število ovrednotenj vnaprej omejeno in je pri vseh testnih funkcijah enako 500.000. Za vsako funkcijo smo 25-krat izvedli algoritem in izmerili uspešnost algoritma. Uspešnost algoritma je definirana kot vsota uspešnosti posameznih zagonov algoritma. Zagon algoritma je uspešen, če je algoritem našel vsaj eno dopustno rešitev $\mathbf{x}$, ki izpolnjuje pogoj $f(\mathbf{x}) - f(\mathbf{x}^*) \leq 0,0001$. Vektor $\mathbf{x}^*$ pomeni globalni optimum (minimum), ki je poznan za vsako testno funkcijo.

Tabela 1 prikazuje dobljene rezultate. V prvem stolpcu so prikazane testne funkcije, in sicer $g01$, $g02$, ..., $g24$. Vrednosti v preostalih stolpcih pomenijo dobljeno uspešnost algoritma (ang. *success rate*) v odstotkih za dano velikost populacije NP.

Predzadnja vrstica tabele 1 prikazuje povprečno uspešnost algoritma za dano velikost populacije. Povprečna uspešnost algoritma je vsota uspešnosti posameznih funkcij, ki jo delimo s 24 (število testnih

Tabela 1. Rezultati na testnih funkcijah
Table 1: Results for the benchmark functions

	Velikost populacije NP														
Funkcija	10	20	30	50	70	80	90	100	110	120	150	200	250	300	500
g01	16	44	80	84	84	92	80	100	84	100	100	100	100	100	100
g02	0	4	24	64	60	72	68	68	88	80	84	80	96	88	100
g03	0	84	72	4	0	0	0	0	0	0	0	0	0	0	0
g04	92	100	100	100	100	100	100	100	100	100	100	100	100	100	100
g05	0	48	68	72	80	96	100	100	100	96	92	88	68	56	16
g06	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
g07	0	0	0	60	100	100	100	100	100	100	100	100	100	100	100
g08	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
g09	0	8	44	100	100	100	100	100	100	100	100	100	100	100	100
g10	0	0	0	64	100	100	100	100	100	100	96	100	100	100	100
g11	68	36	44	32	72	64	84	52	68	64	84	80	88	92	92
g12	92	100	100	100	100	100	100	100	100	100	100	100	100	100	100
g13	0	40	48	40	40	12	24	20	4	12	0	0	0	0	0
g14	0	0	0	4	100	100	100	100	100	100	100	100	100	100	100
g15	0	28	56	96	100	100	100	100	100	100	100	84	92	84	68
g16	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
g17	0	0	0	24	8	12	12	8	4	16	16	12	0	4	0
g18	8	60	96	100	88	96	100	96	96	100	96	100	100	100	100
g19	0	0	0	4	32	64	84	92	100	100	100	100	100	100	100
g20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g21	0	32	44	88	80	64	88	80	64	84	84	92	92	92	80
g22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
g23	0	0	36	88	92	88	100	96	100	96	72	88	76	72	8
g24	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Pov. usp.	28,2	41	50,5	63,5	72,3	73,3	76,7	75,5	75,3	77	76	76	75,5	74,5	69,3
Reš. fun.	9	16	17	22	21	21	21	21	21	21	20	20	19	20	19

funkcij). Najboljšo povprečno uspešnost (77 %) smo dobili pri velikosti populacije $NP = 120$. Zelo blizu temu rezultatu so tudi velikosti populacij 90, 150 in 200, kjer je povprečna uspešnost manjša le za 1 % ali manj.

Zadnja vrstica tabele 1 prikazuje število funkcij, pri katerih je bil algoritem vsaj enkrat uspešen. Opazimo, da je bil algoritem vsaj enkrat uspešen pri 22 testnih funkcijah, ko je bila velikost populacije $NP = 50$. V tem primeru je bila povprečna uspešnost le 63,5-odstotna.

Zanimiv razultat uspešnosti opazimo pri testni funkciji $g03$, kjer pri majhnih velikostih populacije dobimo najboljše rezultate, z rastjo populacije pa kakovost rezultatov pada. Dimenzija funkcije $g03$ je 10. Prav nasprotno obnašanje pa opazimo pri funkcijah $g02$ in $g19$, kjer z rastjo populacije dobimo boljše rezultate.

6 Razprava

Izbira vrednosti kontrolnih parametrov pri algoritmu DE vpliva na kakovost končnih rezultatov optimizacije. Čeprav ima algoritem DE le tri kontrolne parametre, je izbira njihovih vrednosti kar zahtevna naloga oziroma mora imeti uporabnik nekaj izkušenj, da za dano nalogo določi vrednosti kontrolnih parametrov.

V literaturi je po našem najboljšem vedenju malo del, ki bi natančno obdelala vpliv velikosti populacije NP. Omenimo delo [18], kjer avtor Teo opisuje samoprilagodljivi način spreminjanja velikosti populacije. V omenjem delu so bili poskusi opravljeni na testnih funkcijah brez omejitev.

V našem predhodnem prispevku [4] smo opravili primerjavo rezultatov, ki jih je dobil Teo [18], in rezultatov, pridobljenih z našim samoprilagodljivim algoritmom (samoprilaganje kontrolnih parametrov F in CR). Slednji algoritem daje boljše rezultate na testnih funkcijah brez omejitev.

7 Sklep

V članku smo predstavili algoritem diferencialne evolucije, pri katerem nas je zanimal vpliv velikosti populacije (enega od treh kontrolnih prarametrov algoritma) na uspešnost reševanja optimizacije z omejitvami. Poskuse smo izvedli s samoprilagodljivim algoritmom jDE-2 na testnih funkcijah iz literature.

Na podlagi dobljenih rezultatov lahko povzamemo, da velikost populacije NP dokaj močno vpliva na kakovost končne rešitve pri danih testnih funkcijah.

Prav tako lahko zapišemo, da vrednost kontrolnega parametra NP ni enolično določljiva, saj je odvisna

od testnih funkcij. Podoben sklep velja tudi za preostala kontrolna parametra algoritma DE. Vemo, da je dobra izbira velikosti populacije pomembna, zato naše nadaljnje raziskave potekajo v smeri prilagajanja (adaptacije) omenjega kontrolnega parametra.

8 Literatura

[1] T. Bäck. Adaptive Business Intelligence Based on Evolution Strategies: Some Application Examples of Self-Adaptive Software. *Information Sciences*, 148:113–121, 2002.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[3] J. Brest, B. Bošković, S. Greiner, V. Žumer. Nastavitev parametrov pri algoritmu diferencialne evolucije. In Baldomir Zajc, editor, *Zbornik štirinajste mednarodne Elektrotehniške in računalniške konference ERK 2005*, Zvezek B, str. 79–82, Slovenija, September 2005.

[4] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Sepesy Maučec. Performance Comparison of Self-Adaptive and Adaptive Differential Evolution Algorithms. *Soft Comput*, 11(7):617–629, 2007. DOI: 10.1007/s00500-006-0124-0.

[5] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006. DOI 10.1109/TEVC.2006.872133.

[6] J. Brest, V. Žumer, and M. Sepesy Maučec. Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In *The 2006 IEEE Congress on Evolutionary Computation CEC2006*. IEEE Press, pages 919-926, 2006.

[7] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing. Springer-Verlag, Berlin, 2003.

[8] S. Koziel and Z. Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[9] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, N. Suganthan, C. A. C. Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical Report #2006005, Nanyang Technological University, Singapore and et al., Dec, 2005. http://www.ntu.edu.sg/home/EPNSugan.

[10] J. Liu and J. Lampinen. Adaptive Parameter Control of Differential Evolution. In *Proceedings of the 8th International Conference on Soft Computing (MENDEL 2002)*, pages 19–26, 2002.

[11] J. Liu and J. Lampinen. On Setting the Control Parameter of the Differential Evolution Method. In *Proceedings of the 8th International Conference on Soft Computing (MENDEL 2002)*, pages 11–18, 2002.

[12] J. Liu and J. Lampinen. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(6):448–462, 2005.

[13] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[14] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[15] J. Rönkkönen, S. Kukkonen, and K. V. Price. Real-Parameter Optimization with Differential Evolution. In *The 2005 IEEE Congress on Evolutionary Computation CEC2005*, volume 1, pages 506 – 513. IEEE Press, Sept. 2005.

[16] R. Storn and K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[17] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[18] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 10(8):673–686, 2006. DOI: 10.1007/s00500-005-0537-1.

[19] R. K. Ursem and P. Vadstrup. Parameter identification of induction motors using differential evolution. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 790–796, Canberra, 8-12 December 2003. IEEE Press.

Janez Brest je diplomiral leta 1995, magistriral leta 1998 in doktoriral leta 2000 na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Od leta 1993 je zaposlen v Laboratoriju za računalniške arhitekture in jezike, kjer se ukvarja s paralelnim in porazdeljenim procesiranjem s poudarkom na razvrščanju opravil. Njegovo področje dela so tudi programski jeziki, ukvarja pa se tudi z optimizacijskimi raziskavami in evolucijskim računanjem s poudarkom na diferencialni evoluciji.

Viljem Žumer je redni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Vodi Inštitut za računalništvo in Laboratorij za računalniške arhitekture in jezike. Področja, s katerimi se ukvarja, so programski jeziki, paralelno in porazdeljeno procesiranje ter računalniške arhitekture.

Mirjam Sepesy Maučec je diplomirala leta 1996 in doktorirala leta 2001 na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Tudi zaposlena je na tej fakulteti, v Laboratoriju za digitalno procesiranje signalov. Njeno raziskovalno področje obsega modeliranje naravnega jezika, statistično strojno prevajanje in računalniško jezikoslovje. Sodeluje tudi pri raziskavah na področju optimizacijskih postopkov.