Optimizacija z omejitvami: eksperimentalni rezultati s samo-prilagodljivim algoritmom diferencialne evolucije

Janez Brest, Mirjam S. Maučec, Borko Bošković, Sašo Greiner, Viljem Žumer
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova 17, 2000 Maribor, Slovenija
E-pošta: janez.brest@uni-mb.si

Constrained Optimization: Experimental Results obtained by Self-Adaptive Differential Evolution Algorithm

Evolutionary algorithms (EAs) usually have difficulties to solve constrained optimization problems with many equality constraints.

This paper presents empirical results on a set of selected benchmark functions. The results were obtained by our self-adaptive differential evolution algorithm. The paper outlines that the equality constraints really cause difficulties in EAs.

1 Uvod

V članku obravnavamo optimizacijo težkih problemov z omejitvami. Najprej na kratko opišimo nalogo, ki jo rešujemo.

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 mora biti omejena, ne rabi pa biti zvezna ali odvedljiva.

Pri reševanju problemov z omejitvami ponavadi uporabimo kazensko funkcijo. Rešitev $\mathbf{x}$ je *dopustna*, če

$$g_i(\mathbf{x}) \leq 0, \text{ za } i = 1, 2, ..., q, \text{ in}$$

$$|h_j(\mathbf{x})| = 0, \text{ za } j = q+1, q+2, ..., m,$$

kjer omejitve z enakostmi ($h_j(\mathbf{x}) = 0$) transformiramo v omejitve z neenakostmi: $|h_j(\mathbf{x})| - \epsilon \leq 0$. Neenakostne omejitve so tudi $g_i(\mathbf{x}) \leq 0$. V okviru CEC2006 *Special session on constrained real parameter optimization* [6] 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 vrednosti, ki predstavljajo posamezno kršitev omejitev in vsoto, 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 [8].

Preostanek prispevka je organiziran takole. V drugem poglavju predstavimo sorodna dela in opišemo testne funkcije, ki smo jih vzeli iz literature. Tretje poglavje govori o številu evaluacij in uspešnosti algoritma. V četrtem poglavju predstavljamo eksperimentalne rezultate pri optimizaciji z omejitvami na izbranih testnih funkcijah. Sledi še zaključno poglavje z idejo za nadaljnje delo.

2 Motivacija, pregled sorodnih del in testne funkcije

Algoritem diferencialne evolucije (DE) (Differential Evolution) [10, 11, 9] je v zadnjih letih postal popularen za numerično optimizacijo funkcij. Algoritem se ponaša z dobro konvergenco [7].

Algoritem DE lahko prištevamo k evolucijskim algoritmom. Evolucijski algoritmi potrebujejo le funkcijo, ki jo želimo minimizirati (običajno 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.

V našem predhodnem prispevku [3] smo opravili študijo zmogljivosti samo-prilagodljivega algoritma diferencialne evolucije na testnih funkcijah, ki so bile posebej izbrane za *CEC2006 Special session on constrained real parameter optimization* [6].

V članku [5] prikazujemo nadaljnje eksperimentalne rezultate omenjenega algoritma na podmnožici istih testnih funkcij [6], s poudarkom na velikosti populacije NP.

Bralcu, ki bi ga zanimalo več o samo-prilagodljivem algoritmu diferencialne evolucije, priporočamo članka [2, 1]. O nastavitvah kontrolnih parametrov pri algoritmu diferencialne evolucije smo pisali v članku [4], ki je bil predstavljen na konferenci ERK'05 v okviru delavnice AVN (Algoritmi po vzorih iz narave).

V naših predhodnih študijah, predvsem [3, 5], smo ugotovili, da ima naš algoritem težave pri reševanju naslednjih testnih funkcij [6]: $g03$, $g13$, $g17$ in $g22$.

Tabela 1: Testne funkcije $g03$, $g13$ in $g17$.
Table 1: The benchmark functions $g03$, $g13$ and $g17$.

$g03$	$f(\mathbf{x}) = -(\sqrt{n})^n \prod_{i=n}^{n} x_i$
	$h_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$
	$0 < x_i \leq 10 (i = 1, ..., n)$
	$f(\mathbf{x}^*) = -1.00050010001000$
	$x_1^* = 0.31624357647283069$ $x_2^* = 0.31624357741433839$ $x_3^* = 0.31624357801234592$ $x_4^* = 0.31624357566401789$ $x_5^* = 0.31624357820552606$ $x_6^* = 0.31624357738855069$ $x_7^* = 0.31624357547294951$ $x_8^* = 0.31624357716488393$ $x_9^* = 0.31624357815592030$ $x_{10}^* = 0.31624357614737491$
$g13$	$f(\mathbf{x}) = e^{x_1 x_2 x_3 x_4 x_5}$
	$-2.3 \leq x_i \leq 2.3 \; (i = 1, 2),$ $-3.2 \leq x_i \leq 3.2 \; (i = 3, 4, 5)$
	$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$ $h_2(\mathbf{x}) = x_2 x_3 - 5 x_4 x_5 = 0$ $h_3(\mathbf{x}) = x_1^3 + x_2^3 + 1 = 0$
	$f(\mathbf{x}^*) = 0.053941514041898$
	$x_1^* = -1.71714224003$ $x_2^* = 1.59572124049468$ $x_3^* = 1.8272502406271$ $x_4^* = -0.763659881912867$ $x_5^* = -0.76365986736498$
$g17$	$f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$ $f_1(\mathbf{x}) = \begin{cases} 30(h_1(\mathbf{x}) + x_1) & 0 \leq x_1 < 300 \\ 31(h_1(\mathbf{x}) + x_1) & 300 \leq x_1 < 400 \end{cases}$ $f_2(\mathbf{x}) = \begin{cases} 28(h_2(\mathbf{x}) + x_2) & 0 \leq x_2 < 100 \\ 29(h_2(\mathbf{x}) + x_2) & 100 \leq x_2 < 200 \\ 30(h_2(\mathbf{x}) + x_2) & 200 \leq x_2 < 300 \end{cases}$
	$0 \leq x_1 \leq 400, 0 \leq x_2 \leq 1000, 340 \leq x_3 \leq 420,$ $340 \leq x_4 \leq 420, -1000 \leq x_5 \leq 1000,$ $0 \leq x_6 \leq 0.5236$
	$h_1(\mathbf{x}) = -x_1 + 300 - \frac{x_3 x_4}{131.078} cos(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} cos(1.47588) = 0$ $h_2(\mathbf{x}) = -x_2 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} cos(1.47588) = 0$ $h_3(\mathbf{x}) = -x_5 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} cos(1.47588) = 0$ $h_4(\mathbf{x}) = 200 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_2^2}{131.078} cos(1.47588) = 0$
	$f(\mathbf{x}^*) = 8853.53967480648$
	$x_1^* = 201.784467214523659$ $x_2^* = 99.9999999999999005$ $x_3^* = 383.071034852773266$ $x_4^* = 420$ $x_5^* = -10.9076584514292652$ $x_6^* = 0.0731482312084287128$

Tabela 2: Testna funkcija $g22$.
Table 2: The benchmark function $g22$.

$g22$	$f(\mathbf{x}) = x_1$
	$0 \leq x_1 \leq 20000, 0 \leq x_2, x_3, x_4 \leq 1 \times 106,$ $0 \leq x_5, x_6, x_7 \leq 4 \times 107, 100 \leq x_8 \leq 299.99,$ $100 \leq x_9 \leq 399.99, 100.01 \leq x_{10} \leq 300,$ $100 \leq x_{11} \leq 400, 100 \leq x_{12} \leq 600,$ $0 \leq x_{13}, x_{14}, x_{15} \leq 500, 0.01 \leq x_{16} \leq 300,$ $0.01 \leq x_{17} \leq 400,$ $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$
	$g_1(\mathbf{x}) = -x_1 + x_2^{0.6} + x_3^{0.6} + x_4^{0.6} \leq 0$ $h_1(\mathbf{x}) = x_5 - 100000 x_8 + 1 \times 107 = 0$ $h_2(\mathbf{x}) = x_6 + 100000 x_8 - 100000 x_9 = 0$ $h_3(\mathbf{x}) = x_7 + 100000 x_9 - 5 \times 107 = 0$ $h_4(\mathbf{x}) = x_5 + 100000 x_{10} - 3.3 \times 107 = 0$ $h_5(\mathbf{x}) = x_6 + 100000 x_{11} - 4.4 \times 107 = 0$ $h_6(\mathbf{x}) = x_7 + 100000 x_{12} - 6.6 \times 107 = 0$ $h_7(\mathbf{x}) = x_5 - 120 x_2 x_{13} = 0$ $h_8(\mathbf{x}) = x_6 - 80 x_3 x_{14} = 0$ $h_9(\mathbf{x}) = x_7 - 400 x_4 x_{15} = 0$ $h_{10}(\mathbf{x}) = x_8 - x_{11} + x_{16} = 0$ $h_{11}(\mathbf{x}) = x_9 - x_{12} + x_{17} = 0$ $h_{12}(\mathbf{x}) = -x_{18} + ln(x_{10} - 100) = 0$ $h_{13}(\mathbf{x}) = -x_{19} + ln(-x_8 + 300) = 0$ $h_{14}(\mathbf{x}) = -x_{20} + ln(x_{16}) = 0$ $h_{15}(\mathbf{x}) = -x_{21} + ln(-x_9 + 400) = 0$ $h_{16}(\mathbf{x}) = -x_{22} + ln(x_{17}) = 0$ $h_{17}(\mathbf{x}) = -x_8 - x_{10} + x_{13} x_{18} - x_{13} x_{19} + 400 = 0$ $h_{18}(\mathbf{x}) = x_8 - x_9 - x_{11} + x_{14} x_{20} - x_{14} x_{21} + 400 = 0$ $h_{19}(\mathbf{x}) = x_9 - x_{12} - 4.60517 x_{15} + x_{15} x_{22} + 100 = 0$
	$f(\mathbf{x}^*) = 236.430975504001$
	$x_1^* = 236.430975504001054$ $x_2^* = 135.82847151732463$ $x_3^* = 204.818152544824585$ $x_4^* = 6446.54654059436416$ $x_5^* = 3007540.83940215595$ $x_6^* = 4074188.65771341929$ $x_7^* = 32918270.5028952882$ $x_8^* = 130.075408394314167$ $x_9^* = 170.817294970528621$ $x_{10}^* = 299.924591605478554$ $x_{11}^* = 399.258113423595205$ $x_{12}^* = 330.817294971142758$ $x_{13}^* = 184.51831230897065$ $x_{14}^* = 248.64670239647424$ $x_{15}^* = 127.658546694545862$ $x_{16}^* = 269.182627528746707$ $x_{17}^* = 160.000016724090955$ $x_{18}^* = 5.29788288102680571$ $x_{19}^* = 5.13529735903945728$ $x_{20}^* = 5.59531526444068827$ $x_{21}^* = 5.43444479314453499$ $x_{22}^* = 5.07517453535834395$

Testne funkcije so prikazane v tabelah 1 in 2. Pri teh funkcijah algoritem ni bil uspešen (pri funkciji $g17$ je bila

uspešnost algoritma majhna). (Omenimo, da funkcija $g20$ nima dopustne rešitve, zato je bil pri njej tudi naš algoritem neuspešen.)

Skupna lastnost funkcij $g03$, $g13$, $g17$ in $g22$ (poimenujmo jih *izbrane funkcije*) je ta, da imajo te funkcije enakostne omejitve. V tem članku prikazujemo rezultate poskusov na izbranih funkcijah, kjer smo spreminjali in proučevali vpliv števila evaluacij.

3 Število evaluacij in uspešnost algoritma

Ko primerjamo dva evolucijska algoritma (širše gledano vsak algoritem, ki ima populacijo, npr. genetski algoritem itd.), je najbolj *poštena* primerjava tista, ki temelji na enakem številu evaluacij.

Število evaluacij $nevals$ evolucijskega algoritma izračunamo kot produkt NP in števila generacij. Tretji kontrolni parameter NP ponavadi ni vzbudil 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.

Popolnoma drugačen scenarij nastopi, ko fiksiramo $nevals$ (algoritem ustavimo, ko doseže vnaprej predpisano fiksno število evaluacij). Poraja se vprašanje, kako naj določimo/izberemo velikost populacije NP, da bo algoritem dajal najboljše rezultate. Naš namen ni, da bi odgovorili na zastavljeno vprašanje, ampak želimo raziskati obnašanje našega algoritma v primerih, ko predpostavimo, da imamo dovolj zmogljiv računalnik in lahko število evaluacij poljubno povečamo.

Spomnimo naj, da je bilo na CEC'06 [6] število evaluacij vnaprej omejeno, in sicer $nevals = 500.000$.

Za vsako funkcijo smo 25-krat izvedli algoritem in izmerili uspešnost algoritma. Uspešnost algoritma je (ang. *success rate*) 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. Vrednosti $\mathbf{x}^*$ in $f(\mathbf{x}^*)$ so bile podane v [6].

4 Eksperimentalni rezultati

Opravili smo poskuse, kjer smo algoritem testirali pri treh različnih vrednostih za število evaluacij, in sicer 500.000, dva milijona in 10 milijonov.

Tabele 3, 4 in 5 prikazujejo dobljene rezultate. V prvem stolpcu so prikazane izbrane testne funkcije. Vrednosti v ostalih stopcih pomenijo dobljeno uspešnost algoritma v odstotkih za dano velikost populacije NP. Velikost populacije smo izbrali tako, da smo vključili majhne velikosti populacij (od 10 do 30), velike populacije (od 200 do 500) ter poseben poudarek namenili tudi populacijam, ki vsebujejo okrog 100 posameznikov. Razlog slednji odločitvi je v tem, da je večina avtorjev poslanih prispevkov na sekcijo v okviru konference CEC'06 [6] nastavila parameter $NP = 100$, ne glede na to ali so uporabljali algoritem DE ali PSO (ang. Particle Swarm Optimization). Omenimo, da smo v prispevku [3] nastavili parameter $NP = 200$.

V tabeli 3 opazimo zanimiv rezultat uspešnosti pri testni funkciji $g03$, kjer pri majhnih velikostih populacije dobimo najboljše rezultate, z večanjem velikosti populacije pa kvaliteta rezultatov pada. Nekaj podobnega, a morda manj izrazito, lahko vidimo tudi pri funkciji $g13$.

Če primerjamo rezultate pri funkciji $g03$ v vseh treh tabelah, lahko ugotovimo, da z večanjem števila evaluacij dobimo boljše rezultate, saj se v tabelah večkrat pojavlja tudi vrednost 100.

Iz eksperimentalnih rezultatov v tabelah 3, 4 in 5 lahko sklepamo, da je uspešnost algoritma precej odvisna tudi od kontrolnega parametra NP pri vnaprej predpisanem številu evaluacij.

Naš algoritem je bil neuspešen pri funkciji $g22$. Funkcija $g22$ ima veliko število nelinearnih enakostnih omejitev. Pri tej funkciji je težavno že najti dopustno rešitev (ang. *feasible solution*).

Naš algoritem je v vseh zagonih našel *dopustno rešitev* pri 22 funkcijah že pri $nevals = 500.000$. Spomnimo, da je bilo vseh funkcij 24. Izjemi sta funkciji $g20$ (ki nima dopustne rešitve) in $g22$. Pri funkciji $g22$ je naš algoritem pri iskanju dopustne rešitve pri $nevals = 500.000$ dosegel 33% uspešnost, pri $nevals = 10.000.000$ pa 80% uspešnost.

5 Zaključek

V članku smo opravili študijo vpliva števila evaluacij (posredno tudi vpliv velikosti populacije) na rezultat optimizacije z nelinearnimi enakostnimi omejitvami. Eksperimentalne rezulate smo izvedli na izbranih testnih funkcijah iz literature.

Nelinearne enakostne omejitve pri optimizaciji ponavadi pomenijo trd oreh za algoritme v literaturi. Podobno velja tudi za naš samo-prilagodljivi algoritem diferencialne evolucije, kar so potrdili ekperimentalni rezultati v tem članku.

V članku smo ugotovili, da s povečanjem števila evaluacij naš algoritem daje boljše rezultate pri treh od štirih izbranih (težkih) testnih funkcijah z nelinearnimi enakostnimi omejitvami.

Pri funkciji $g22$ naš algoritem ni bil niti enkrat uspešen, in poraja se vprašanje, ali bi še z nadaljnjim povečanjem števila iteracij morda algoritem postal uspešen. Naj bo to smernica za nadaljnje raziskave.

Literatura

[1] J. Brest, B. Bošković, S. Greiner, V. Žumer, M. Sepesy Maučec. Performance Comparison of Self-Adaptive and Adaptive Differential Evolution Algorithms. *Soft Comput*, 2006. Accepted.

[2] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Nu-

Tabela 3: Rezultati uspešnosti algoritma na testnih funkcijah pri $nevals = 500.000$.
Table 3: Success rate for the benchmark functions when $nevals = 500.000$.

	Velikost populacije NP														
Funkcija	10	20	30	50	70	80	90	100	110	120	150	200	250	300	500
$g03$	0	84	72	4	0	0	0	0	0	0	0	0	0	0	0
$g13$	0	40	48	40	40	12	24	20	4	12	0	0	0	0	0
$g17$	0	0	0	24	8	12	12	8	4	16	16	12	0	4	0
$g22$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 4: Rezultati uspešnosti algoritma na testnih funkcijah pri $nevals = 2.000.000$.
Table 4: Success rate for the benchmark functions when $nevals = 2.000.000$.

	Velikost populacije NP														
Funkcija	10	20	30	50	70	80	90	100	110	120	150	200	250	300	500
$g03$	0	84	92	100	100	100	100	100	100	92	44	0	0	0	0
$g13$	0	44	48	40	40	28	52	44	40	60	40	20	32	36	8
$g17$	0	0	28	40	44	32	32	32	52	44	52	48	32	32	20
$g22$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 5: Rezultati uspešnosti algoritma na testnih funkcijah pri $nevals = 10.000.000$.
Table 5: Success rate for the benchmark functions when $nevals = 10.000.000$.

	Velikost populacije NP														
Funkcija	10	20	30	50	70	80	90	100	110	120	150	200	250	300	500
$g03$	0	84	92	100	100	100	100	100	100	100	100	100	100	100	100
$g13$	0	44	48	40	40	28	52	44	40	60	44	36	44	60	44
$g17$	0	0	28	40	44	32	32	32	52	44	52	48	32	32	20
$g22$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

merical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 2006. Accepted.

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

[4] Janez Brest, Borko Boškovič, Sašo Greiner, Viljem Žumer. Nastavitev parametrov pri algoritmu diferencialne evolucije. V: Baldomir Zajc, editor, *Zbornik štirinajste mednarodne Elektrotehniške in računalniške konference ERK 2005*, volume Zvezek B, pages 79–82, Slovenija, September 2005.

[5] Janez Brest, Viljem Žumer, Mirjam Sepesy Maučec. Velikost populacije pri algoritmu diferencialne evolucije. *Elektroteh. vestn.*, 2006. Poslano za objavo.

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

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

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

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

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

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