Elektrotehniški vestnik 75(5): 299–304, 2008
Electrotechnical Review, Ljubljana, Slovenija

Globalna optimizacija problemov z velikim številom dimenzij

Janez Brest, Aleš Zamuda, Borko Bošković, Viljem Žumer

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

Povzetek. V članku obravnavamo algoritem diferencialne evolucije (DE) za numerično optimizacijo. V zadnjem času ga uporabljamo v številnih praktičnih aplikacijah. V članku opisujemo optimizacijo, kjer iščemo globalni optimum testnih funkcij z dimenzijami 100, 500 in 1000. Predstavili bomo rezultate, ki smo jih dobili na izbranih testnih funkcijah z dvema algoritmoma diferencialne evolucije iz literature. Prvi je originalni algoritem DE, drugi pa je samoprilagodljivi algoritem DE. Slednji na izbranih testnih primerih daje boljše rezultate.

Ključne besede: evolucijski algoritem, optimizacija, iskanje globalnega optimuma, diferencialna evolucija, samoprilagajanje

High-dimensional Global Optimization

Extended Abstract. In this paper we present performance evaluation of the original differential evolution (DE) and our self-adaptive differential evolution algorithm. The algorithm, which we named jDE [4, 6], uses a self-adapting control parameter mechanism on control parameters F and CR during the optimization process. The performance of the algorithm is evaluated on a set of benchmark functions provided for the CEC 2008 special session on high-dimensional real-parameter optimization [18].

To solve high-dimensional problems [12, 18], cooperative coevolution [13, 15] is used in literature. Liu et al. [11] uses FEP (fast evolutionary programming) with cooperative coevolution (FEPCC) to speed-up convergence rates on the large-scale problems, Bergh and Engelbrecht [20] applies a Cooperative Approach to Particle Swarm Optimisation (PSO), Yang, Tang, and Yao uses differential evolution with cooperative coevolution (DECC) [21].

The performance of the jDE and the original DE algorithms is evaluated on a set of seven benchmark functions provided for the CEC 2008 special session [18] each for three dimensions $D = 100$, $D = 500$, and $D = 1000$. The obtained results (see the tables and figure) show that the self-adaptive jDE algorithm outperforms the original DE algorithm for each tested function.

Key words: evolutionary algorithm, optimization method, global optimum search, differential evolution, self-adaptation

1 Uvod

V članku predstavljamo eksperimentalne rezultate algoritmov diferencialne evolucije (DE) na testnih funkcijah iz posebne sekcije mednarodnega simpozija evolucijskega računanja *CEC 2008 Special Session and Competition on High-Dimensional Real-Parameter Optimization* [18]. V naboru je sedem testnih funkcij, ki jih je treba ovrednotiti pri treh različnih dimenzijah (100, 500 in 1000). Vnaprej predpisano oziroma omejejo je število ovrednotenj funkcij FEs, kot prikazuje tabela 1. Predpisano število ovrednotenj določene funkcije je ustavitveni pogoj posameznega zagona algoritma. Pri vsaki funkciji je treba narediti 25 zagonov algoritma z različnim semenom psevdonaključnega generatorja števil. Samih testnih funkcij [18] v tem članku ne prikazujemo, ker so preobsežne, ampak smo v tabeli 2 zbrali le nekatere njihove lastnosti. Pri vseh testnih funkcijah je treba poiskati globalni minimum. Začetna populacija je izbrana naključno (po enakomerni porazdelitvi) med spodnjo in zgornjo mejo, ki sta definirani za vsako spremenljivko testne funkcije. 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 navzdol omejena in ni nujno, da je zvezna ali odvedljiva.

Diferencialna evolucija (ang. *Differential Evolution*) [17] je popularen algoritem za numerično optimizacijo funkcij in ga uporabljamo v številnih praktičnih aplikacijah, predvsem zato, ker algoritem odlikuje dobra konvergenca [16, 17, 14, 9, 10, 19, 4]. Algoritem DE prištevamo k evolucijskim algoritmom. V našem predhodnem prispevku [6] smo opravili študijo zmogljivosti samoprilagodljivega algoritma diferencialne evolucije na testnih funkcijah z omejitvami (CEC 2006). Samoprila-

Prejet 19. december, 2007
Odobren 15. september, 2008

Tabela 1. Število ovrednotenj testnih funkcij FEs
Table 1: Number of FEs Function Evaluations

Dimenzija D	100	500	1000
FEs	500.000	2.500.000	5.000.000

Tabela 2. Testne funkcije CEC'08
Table 2: CEC'08 Test Functions

F_1	Shifted Sphere Function	UM	SE
F_2	Shifted Schwefel's Problem 2.21	UM	NS
F_3	Shifted Rosenbrock's Function	MM	NS
F_4	Shifted Rastrigin's Function	MM	SE
F_5	Shifted Griewank's Function	MM	NS
F_6	Shifted Ackley's Function	MM	SE
F_7	FastFractal "DoubleDip" Function	MM	SE

UM/MM - unimodalne/multimodalne funkcije (Unimodal/Multi-modal)

SE/NS - ločljive/neločljive funkcije (Separable/Non-separable)

gajanje smo uporabili tudi na testnih funkcijah iz posebne sekcije CEC 2007 za večkriterijsko optimizacijo [22].

Nadaljevanje prispevka je organizirano takole. V 2. in 3. poglavju na kratko opišemo algoritma diferencialne evolucije s samoprilagajanjem in brez njega. Rezultati, ki smo jih dobili s pomočjo obeh algoritmov na optimizacijskih poskusih, so prikazani v 4. poglavju. V 5. poglavju podajamo razpravo o dobljenih rezultatih. Sledi še sklepno, 6. poglavje, kjer podamo tudi smernice za nadaljnje raziskave.

2 Algoritem diferencialne evolucije

Algoritem DE ima tri krmilne parametre (F - faktor skaliranja, CR - krmilni parameter križanja, NP - velikost populacije), ki se pri originalnem algoritmu DE [16, 17] med optimizacijskim postopkom ne spreminjajo [16, 10, 9]. Algoritem diferencialne evolucije (DE) ustvari novega posameznika s kombinacijo starša in izbranih posameznikov iste populacije. Nov posameznik zamenja starša, če ima boljšo ocenitveno vrednost (funkcijsko vrednost). Algoritem DE [16, 17, 14, 8] je opisan v literaturi, tudi v slovenski [5], zato bomo opis algoritma izpustili.

Omenimo le, da pri svojem delu uporabljamo strategijo mutacije "rand/1": $\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F * (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G})$. Mutiran vektor $\mathbf{v}_{i,G}$ izračunamo s pomočjo treh naključno izbranih vektorjev (indeksi r_1, r_2, r_3 so med seboj različni in tudi različni od indeksa i). G označuje generacijo, F in CR pa sta krmilna parametra pri algoritmu DE.

3 Samoprilagodljivi algoritem diferencialne evolucije

Izbira ustreznih krmilnih parametrov je ponavadi odvisna od problema, ki ga rešujemo, in od uporabnika zahteva predhodno znanje oziroma izkušnje. Mehanizem samoprilagajanja omogoča, da se evolucija (sama) prilagodi naravi problema, tako da se ustrezno rekonfigurira. Ta prilagoditev se opravi brez aktivne vloge uporabnika [1, 2, 7]. Poskusi v [4, 3] so pokazali, da s spreminjanjem krmilnih parametrov med optimizacijskim postopkom lahko izboljšamo rezultate. V [4] je predstavljen samoprilagodljivi mehanizem, ki med evolucijo spreminja krmilna parametra F in CR. Na kratko predstavimo ta mehanizem. Nova krmilna parametra $F_{i,G+1}$ in $CR_{i,G+1}$ izračunamo takole:

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

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

Tako dobimo krmilna parametra F in CR pri novem posamezniku. $rand_j, j \in \{1, 2, 3, 4\}$ so naključne realne vrednosti na intervalu $[0, 1)$. τ_1 in τ_2 sta verjetnosti, s katerima uravnavamo krmilna parametra F in CR. Uporabljamo naslednje fiksne vrednosti: $\tau_1 = 0,1; \tau_2 = 0,2; F_l = 0,1$ in $F_u = 0,9$, zato ima F naključno vrednost na intervalu $[0,1; 1,0)$ in CR na intervalu $[0, 1)$.

Samoprilagodljivi algoritem DE imenujemo jDE. Algoritem jDE se od originalnega algoritma DE razlikuje le v tem, da ima prvi samoprilagodljiva krmilna parametra, algoritem DE pa med postopkom evolucije ne spreminja krmilnih parametrov.

4 Eksperimentalni rezultati

V tem poglavju predstavimo rezultate, ki smo jih dobili pri poskusih, kjer smo uporabili originalni algoritem DE in samoprilagodljivi algoritem jDE. Algoritma sta v poskusih imela enake nastavitve parametrov:

- isto seme pri psevdonaključnem generatorju števil,
- začetna populacija je bila izbrana naključno (po enakomerni porazdelitvi) med spodnjo in zgornjo mejo,
- velikost populacije $NP = 100$,
- krmilna parametra $F = 0,5$ in $CR = 0,9$ pri algoritmu DE (fiksna parametra); algoritem jDE uporablja samoprilagodljiva parametra (začetna populacija je bila inicializirana: $F = 0,5$ in $CR = 0,9$).

Za vsako funkcijo in vsako dimenzijo smo 25-krat izvedli algoritem. Najboljšo rešitev (napaka: $f(\vec{x}) - f(\vec{x}^*)$) vsakega zagona smo si zapomnili in dobljenih 25 rešitev uredili naraščajoče. Vrednosti $f(\vec{x}) - f(\vec{x}^*)$ so predstavljene v tabelah 3, 5 in 7 za samoprilagodljivi algoritem jDE, v tabelah 4, 6 in 8 pa za algoritem DE. V tabelah lahko pri vsaki funkciji vidimo najboljšo, sedmo, srednjo (mediano), devetnajsto in najslabšo rešitev ter povprečno vrednost in standardno deviacijo na koncu ob izpolnjenem ustavitvenem pogoju pri ovrednotenjih FEs. Pri funkciji

Tabela 3. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 100$ z algoritmom jDE
Table 3: Error Values Achieved for Problems F_1–F_7, with $D = 100$ by the jDE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	5.6843e-14	7.3152e-02	6.2800e+01	5.6843e-14	2.8421e-14	5.6843e-14	-1.4569e+03
	7	5.6843e-14	1.7001e-01	8.1266e+01	5.6843e-14	2.8421e-14	5.6843e-14	-1.4342e+03
	13 (srednji)	5.6843e-14	2.6638e-01	1.3986e+02	5.6843e-14	2.8421e-14	5.6843e-14	-1.4269e+03
5.00e + 5	19	5.6843e-14	4.6816e-01	1.4790e+02	5.6843e-14	2.8421e-14	5.6843e-14	-1.4238e+03
	25 (najslabši)	5.6843e-14	9.8656e-01	2.0342e+02	5.6843e-14	2.8421e-14	8.5265e-14	-1.4180e+03
	povprečje	5.6843e-14	3.4198e-01	1.2890e+02	5.6843e-14	2.8422e-14	6.1391e-14	-1.4298e+03
	std. dev.	0.0000	2.3059e-01	4.3159e+01	0.0000	0.0000	1.0634e-14	8.2645

Tabela 4. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 100$ z algoritmom DE
Table 4: Error Values Achieved for Problems F_1–F_7, with $D = 100$ by the DE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	5.6843e-14	3.1046e+01	8.9499e+01	1.6832e+02	0.0000	2.5414e-10	-8.9245e+02
	7	5.6843e-14	4.3822e+01	1.3434e+02	4.2061e+02	2.8421e-14	5.1753e-10	-8.7133e+02
	13 (srednji)	5.6843e-14	4.8172e+01	1.4808e+02	5.9794e+02	2.8421e-14	6.7637e-10	-8.6847e+02
5.00e + 5	19	5.6843e-14	5.1356e+01	1.9862e+02	6.2373e+02	2.8421e-14	1.0136e-09	-8.5696e+02
	25 (najslabši)	5.6843e-14	6.6864e+01	5.8562e+02	7.0895e+02	9.8572e-03	2.1420e-09	-8.4167e+02
	povprečje	5.6843e-14	4.8062e+01	1.8766e+02	5.2934e+02	3.9429e-04	8.4423e-10	-8.6485e+02
	std. dev.	0.0000	7.1481	1.0858e+02	1.5189e+02	1.9715e-03	5.2999e-10	1.2485e+01

Tabela 5. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 500$ z algoritmom jDE
Table 5: Error Values Achieved for Problems F_1–F_7, with $D = 500$ by the jDE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	3.9790e-13	3.8175e+01	4.9034e+02	6.2488e+02	1.4210e-13	3.6519e-08	-6.0423e+03
	7	5.1159e-13	3.9040e+01	5.4842e+02	6.8456e+02	1.4210e-13	4.1354e-08	-6.0204e+03
	13 (srednji)	5.6843e-13	3.9637e+01	5.6717e+02	7.1079e+02	1.4210e-13	4.5708e-08	-6.0040e+03
2.50e + 6	19	6.2527e-13	4.0063e+01	6.0013e+02	7.2243e+02	1.7053e-13	4.8368e-08	-5.9843e+03
	25 (najslabši)	7.3896e-13	4.0467e+01	7.5753e+02	7.4040e+02	1.9895e-13	5.6729e-08	-5.9387e+03
	povprečje	5.6616e-13	3.9550e+01	5.8047e+02	6.9994e+02	1.5575e-13	4.5743e-08	-6.0013e+03
	std. dev.	8.9095e-14	6.3865e-01	5.7368e+01	3.0713e+01	1.6654e-14	5.3394e-09	2.5487e+01

Tabela 6. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 500$ z algoritmom DE
Table 6: Error Values Achieved for Problems F_1–F_7, with $D = 500$ by the DE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	2.6712e+03	7.8251e+01	8.9794e+07	5.2376e+03	2.2536e+01	4.5731	-3.6172e+03
	7	3.1558e+03	8.0698e+01	1.1739e+08	5.3264e+03	2.4684e+01	4.7577	-3.5306e+03
	13 (srednji)	3.3843e+03	8.2562e+01	1.2770e+08	5.3914e+03	2.7576e+01	4.8659	-3.5137e+03
2.50e + 6	19	3.5982e+03	8.3484e+01	1.3810e+08	5.4355e+03	2.9927e+01	5.0220	-3.4905e+03
	25 (najslabši)	5.0046e+03	8.7378e+01	1.8786e+08	5.4977e+03	3.1462e+01	5.3125	-3.4787e+03
	povprečje	3.4670e+03	8.2513e+01	1.3129e+08	5.3809e+03	2.7300e+01	4.8964	-3.5156e+03
	std. dev.	5.6655e+02	2.2894	2.4021e+07	6.9047e+01	2.6525	1.9446e-01	2.9557e+01

F_7 ne poznamo optimalne rešitve, zato v tabelah prikazujemo izračunano funkcijsko vrednost ($f(\vec{x})$).

Slika 1 prikazuje grafe za testne funkcije F_1–F_6 za dimenzijo $D = 1000$ za algoritem jDE.

Iz dobljenih eksperimentalnih rezultatov lahko ugotovimo, da samoprilagodljivi algoritem jDE daje boljše rezultate kot originalni algoritem DE.

5 Razprava

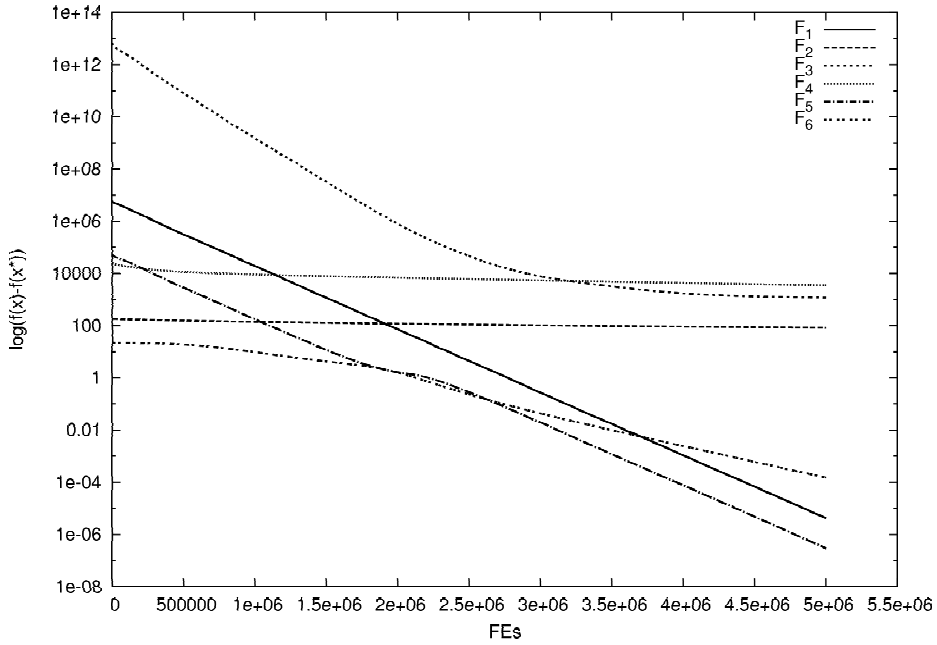
Iz dobljenih eksperimentalnih rezultatov lahko povzamemo, da izbira vrednosti krmilnih parametrov pri algoritmu DE vpliva na kakovost končnih rezultatov optimizacije in da samoprilagajanje krmilnih parametrov lahko izboljša delovanje algoritma DE. Čeprav ima algoritem DE le tri krmilne parametre, je izbira njihovih vrednosti kar zahtevna naloga oziroma mora imeti uporabnik nekaj izkušenj, da za dano nalogo določi vrednosti krmil-

Tabela 7. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 1000$ z algoritmov jDE
Table 7: Error Values Achieved for Problems F_1–F_7, with $D = 1000$ by the jDE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	2.8124e-06	8.2129e+01	1.0359e+03	3.3985e+03	2.1808e-07	1.2240e-04	-1.0946e+04
	7	3.8026e-06	8.2758e+01	1.0835e+03	3.5022e+03	2.4445e-07	1.4004e-04	-1.0886e+04
	13 (srednji)	4.1481e-06	8.3211e+01	1.1430e+03	3.5341e+03	2.7046e-07	1.5244e-04	-1.0847e+04
5.00e + 6	19	4.3810e-06	8.3582e+01	1.2656e+03	3.5629e+03	3.1873e-07	1.6069e-04	-1.0814e+04
	25 (najslabši)	6.3884e-06	8.4007e+01	1.4265e+03	3.6476e+03	4.5636e-07	1.6993e-04	-1.0708e+04
	povprečje	4.2143e-06	8.3179e+01	1.1886e+03	3.5238e+03	2.9424e-07	1.5009e-04	-1.0841e+04
	std. dev.	7.2974e-07	5.1694e-01	1.2562e+02	6.4503e+01	6.4968e-08	1.3482e-05	5.5668e+01

Tabela 8. Napaka pri testnih funkcijah F_1–F_7 pri dimenziji $D = 1000$ z algoritmom DE
Table 8: Error Values Achieved for Problems F_1–F_7, with $D = 1000$ by the DE algorithm

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	1 (najboljši)	2.3032e+05	1.1651e+02	3.3088e+10	1.1568e+04	1.9478e+03	1.4250e+01	-6.7212e+03
	7	2.4412e+05	1.2476e+02	3.8438e+10	1.1948e+04	2.1799e+03	1.4633e+01	-6.6712e+03
	13 (srednji)	2.5739e+05	1.2610e+02	4.1731e+10	1.2017e+04	2.2850e+03	1.4740e+01	-6.6425e+03
5.00e + 6	19	2.7117e+05	1.2799e+02	4.3996e+10	1.2085e+04	2.2990e+03	1.4856e+01	-6.6277e+03
	25 (najslabši)	2.8695e+05	1.3220e+02	4.8194e+10	1.2291e+04	2.5617e+03	1.5097e+01	-6.5950e+03
	povprečje	2.5852e+05	1.2594e+02	4.1476e+10	1.2019e+04	2.2618e+03	1.4740e+01	-6.6517e+03
	std. dev.	1.5439e+04	3.4090	4.0064e+09	1.3897e+02	1.3470e+02	1.9364e-01	3.6292e+01

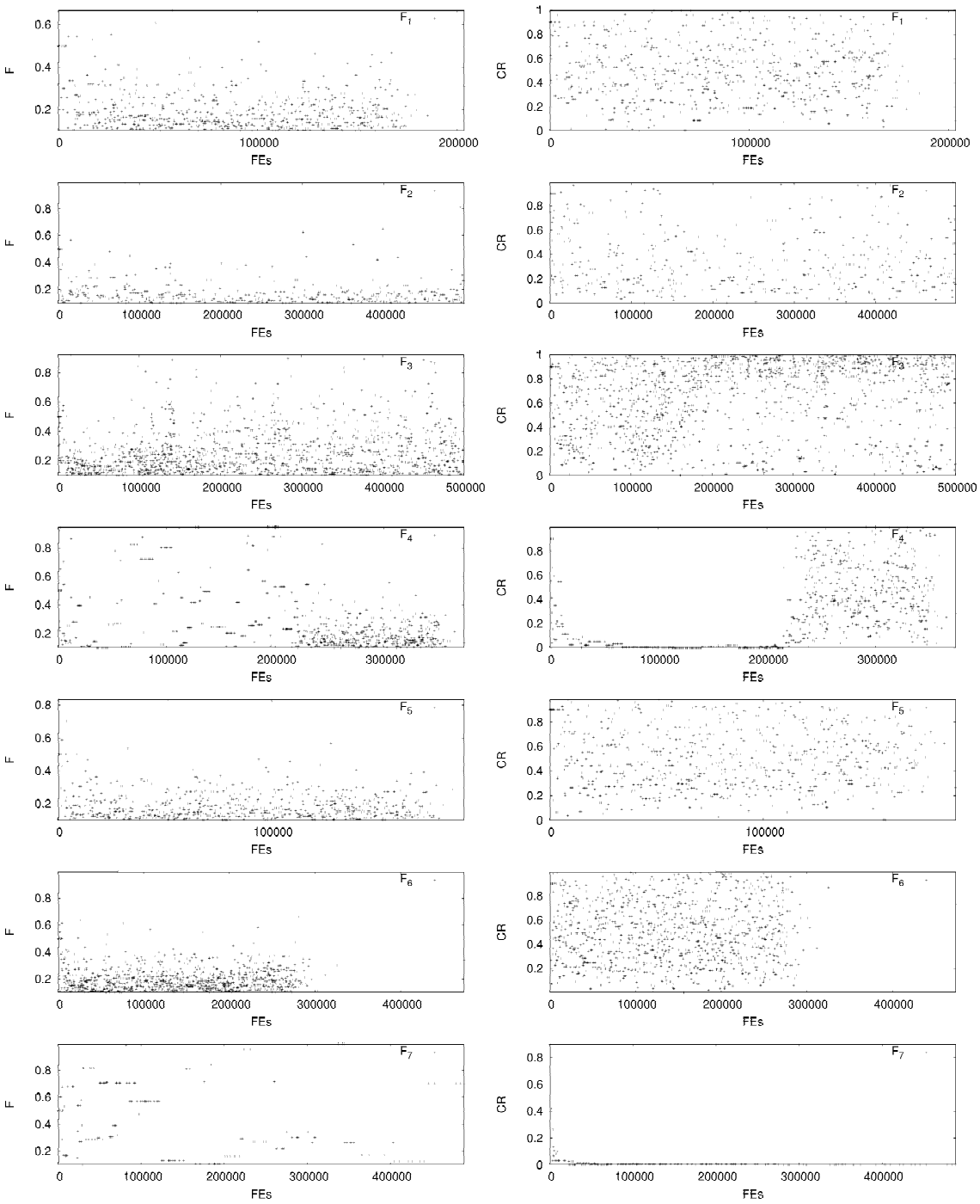


Slika 1. Grafi približevanja za testne funkcije F_1–F_6 pri dimenziji $D = 1000$
Figure 1: Convergence graphs for problems F_1–F_6, with $D = 1000$

nih parametrov.

Nekdo bi lahko komentiral, da morda izbira krmilnih parametrov ni bila "najboljša" in da bi morda z drugimi vrednostmi dobili boljše rezultate. Takemu komentarju sicer ne moremo nasprotovati, a bi poudarili, da zagon enega algoritma na vseh testnih funkcijah za vse tri dimenzije traja približno pet dni. Torej smo pri izračunu rezultatov tega članka potrebovali približno 10 dni (poskusi so bili izvedeni na GNU/Linux x86_64, 2,4 GHz (2x Dual Core AMD Opteron), programski jezik C/C++).

Poskuse smo izvedli s sekvenčnima algoritmoma. Algoritme, ki temeljijo na diferencialni evoluciji, je mogoče tudi paralelizirati. Z vzporedno izvedbo ali z vzporednimi algoritmi bi poskuse opravili hitreje.



Slika 2. Grafi vrednosti krmilnih parametrov za testne funkcije F_1–F_7 pri dimenziji $D = 100$
Figure 2: Graphs of control paramater values for problems F_1–F_7, with $D = 100$

6 Sklep

V članku smo predstavili eksperimentalne rezultate dveh algoritmov diferencialne evolucije, enega s samoprilagajenjem in drugega brez samoprilagajanja krmilnih parametrov F in CR. Poskuse smo izvedli na testnih funkcijah iz literature, pri katerih je dimenzija $D = 100$, $D = 500$ in $D = 1000$. Na podlagi dobljenih rezultatov lahko povzamemo, da samoprilagajanje krmilnih parametrov F in CR dokaj močno vpliva na kakovost končne rešitve pri danih testnih funkcijah. Kot možnost za nadaljnje raziskave vidimo uporabo koevolucije (ang. *coevolution*).

7 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, and M. Sepesy Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11(7):617–629, 2007. DOI: 10.1007/s00500-006-0124-0.

[4] 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.

[5] J. Brest, V. Žumer, and M. S. Maučec. Population size in differential evolution algorithm. *Elektrotehniški vestnik*, 74(1-2):55–60, 2007.

[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, 2006.

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

[8] Vitaliy Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] 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.

[10] 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.

[11] Yong Liu, Xin Yao, Qiangfu Zhao, and Tetsuya Higuchi. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1101–1108, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press.

[12] Cara MacNish. Towards Unbiased Benchmarking of Evolutionary and Hybrid Algorithms for Real-valued Optimisation. *Connection Science*, 19(4):225–239, 2007.

[13] Mitchell A. Potter and Kenneth De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pages 249–257, Berlin, 1994. Springer.

[14] 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.

[15] D. Sofge, K. De Jong, and A. Schultz. A Blended Population Approach to Cooperative Coevolution for Decomposition of Complex Problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pages 413–418. IEEE, 2002.

[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] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark Functions for the CEC'2008 Special Session and Competition on High-Dimenasional Real-Parameter Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. http://nical.ustc.edu.cn/cec08ss.php.

[19] 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.

[20] F. van den Bergh and A. P. Engelbrecht. A Cooperative Approach to Particle Swarm Optimisation. *IEEE Transactions on Evolutionary Computing*, 3:225–239, 2004.

[21] Zhenyu Yang, Ke Tang, and Xin Yao. Differential Evolution for High-Dimensional Function Optimization. In Dipti Srinivasan and Lipo Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 3523–3530, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.

[22] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Differential Evolution for Multiobjective Optimization with Self Adaptation. In *The 2007 IEEE Congress on Evolutionary Computation CEC2007*, pages 3617–3624. IEEE Press, 2007.

Janez Brest je izredni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Zaposlen je v Laboratoriju za računalniške arhitekture in jezike, kjer se ukvarja s paralelnim in porazdeljenim procesiranjem, s programskimi jeziki, ukvarja pa se tudi z optimizacijskimi raziskavami in evolucijskim računanjem s poudarkom na diferencialni evoluciji in samoprilagajanju krmilnih parametrov.

Aleš Zamuda je leta 2008 magistriral na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Trenutno je podiplomski študent in zaposlen v Laboratoriju za računalniške arhitekture in jezike, kjer raziskuje na področju večkriterijske optimizacije in proučuje mehanizme samoprilagajanja za diferencialno evolucijo. Je član IEEE Computational Intelligence Society.

Borko Bošković je diplomiral leta 2003 na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Od leta 2000 je zaposlen v Laboratoriju za računalniške arhitekture in jezike, kjer se ukvarja s spletnim programiranjem, programskimi jeziki, evolucijskimi algoritmi in iskalnimi algoritmi s poudarkom na algoritmih za igranje iger med dvema igralcema, popolno informacijo in ničelno vsoto.

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.