AUTOMATYKA
CZASOPISMO TECHNICZNE
TECHNICAL TRANSACTIONS
AUTOMATIC CONTROL
WYDAWNICTWO POLITECHNIKI KRAKOWSKIEJ
1-AC/2012
ZESZYT 25
ROK 109
ISSUE 25
YEAR 109

MATEUSZ DZIEDZIC*

CLUSTERING BASED POPULATION SIZE REDUCTION METHOD FOR EVOLUTIONARY ALGORITHMS

METODA REDUKCJI LICZNOŚCI POPULACJI ALGORYTMÓW EWOLUCYJNYCH OPARTA NA KLASTERYZACJI

Abstract

Nowadays, due to the growing dimensionality of optimisation problems, numerous studies are dedicated to reduction of metaheuristics computational requirements. Reducing size of the population during optimisation process is one of the promising research trends in the field of Evolutionary Algorithms. The purpose of this paper is to clarify the subject in form of a survey of population size reduction methods already proposed and to present preliminary results of a new method based on the clustering technique. Introduced method was implemented in the framework of Differential Evolution algorithm and verified on a set of real-parameter benchmark functions.

Keywords: Metaheuristics, Evolutionary Algorithms, Differential Evolution, Population Size Reduction, Clustering

Streszczenie

Obecnie, ze względu na ciągły wzrost wymiarowości problemów optymalizacyjnych, liczne prace poświęcone są zmniejszeniu zapotrzebowania metaheurystyk na zasoby obliczeniowe. Jednym z obiecujących kierunków badań w przypadku algorytmów ewolucyjnych jest redukcja liczności populacji w trakcie procesu optymalizacji. Niniejszy artykuł ma na celu przybliżenie tej tematyki w formie przeglądu dotychczas zaproponowanych metod oraz prezentację wstępnych wyników autorskiej metody opartej na technice klasteryzacji. Przedstawiona metoda została zaimplementowana w strukturę algorytmu ewolucji różnicowej i zweryfikowana za pomocą standardowego zestawu funkcji rzeczywistych wielu zmiennych.

Słowa kluczowe: kluczowe: metaheurystyki, algorytmy ewolucyjne, ewolucja różnicowa, redukcja liczności populacji, klasteryzacja

* Mgr inż. Mateusz Dziedzic, Katedra Automatyki i Technik Informacyjnych, Wydział Inżynierii Elektrycznej i Komputerowej, Politechnika Krakowska oraz Studia Doktoranckie, Instytut Badań Systemowych, Polska Akademia Nauk.

1. Introduction

At present, real-valued global optimisation problems over continuous spaces are ubiquitous. Therefore, growing interest is observed in subject of optimisation heuristics which allows to bypass most limitations of traditional methods: they are able to cope with non-differentiable, nonlinear and multimodal cost functions. Moreover, they present high potential for parallelization, which has recently become a required feature with the development of widely available multicore and GPGPU computer hardware. Popular branch of metaheuristics are Evolutionary Algorithms (EAs) which satisfy all above conditions. However, the need to cope with more and more computation demanding functions constantly stimulates extensive work on reducing time and resources requirements. Adjusting the size of the population during optimisation process is one of the promising research trends in the field of Evolutionary Algorithms.

This paper has two main purposes. The first one is to review the current state of knowledge of methods of the size adjustment (reduction) of Evolutionary Algorithms population. The second purpose is to present preliminary results of the newly proposed method based on a clustering technique.

The proposed size reduction method was implemented in the framework of Differential Evolution (DE) [27, 28] algorithm and tested on a set of real-parameter single-objective benchmark functions of the 2012 Black-Box Optimization Benchmarking (BBOB 2012)[1] noiseless testbed [14]. DE with proposed size reduction procedure was compared with standard DE and DE with population size initially narrowed to the value corresponding to number of function evaluations in investigated DE implementation.

This paper is organised as follows. Second and third section aims to present the topic of EA in general, and DE in particular, and their basics derived from nature. The two following sections form a survey of self-adaptation and population size self-adjustment techniques. The main part of the paper are sections six and seven, the former one introducing the proposed method of population size reduction, and the latter describing the computational example and presenting the results of preliminary experiments. Finally, a summary of the work and conclusions are presented in section 8.

The preliminary version of this paper was presented at ICACIT'2011 conference [10]. This paper gives an extensive description of the proposed method and verifies it on more standard set of benchmark functions.

2. Optimisation inspired by nature – Evolutionary Algorithms

Evolutionary Algorithms [4, 5] constitute one of the areas of biologically inspired metaheuristics. They adopt basics of evolution derived from nature [4, 5], such as a population of individuals (solution candidates), the mechanisms of reproduction, crossover and mutation, and selection of the best individuals, to solve complex optimisation problems. EAs' task is to improve the population iteratively (from current generation to the next one), which in turn should improve the best solution. They are most widely used, like other metaheuristics, in

[1] The BBOB 2012 workshop took place during the 2012 Genetic and Evolutionary Computation Conference (GECCO 2012), 7–11 July 212, Philadelphia USA.

NP-hard problems [4, 5]. EAs' biggest drawback is the lack of certainty of finding the optimal solution, but often it suffices to find a solution that is good enough in a shorter time [4, 5].

Usually, the initial population consists of randomly generated individuals (although there are also implementations exploiting already at this point knowledge about the optimised problem in order to accelerate optimisation process, see e.g. [9,16, 31]), which are involved in the processes of crossover and mutation, then go through the reproduction and best individuals selection to form the population in the next generation. Evaluation and selection of individuals are based on their quality, represented by the optimised function's (often referred to as quality function) value $f(x)$ of the candidate x.

More formally, given a solution space $\boldsymbol{X} \in \mathbb{R}^n$ and an optimised function f: $\boldsymbol{X} \rightarrow \mathbb{R}$, metaheuristic task is to of find best solution candidate $x^* \in \boldsymbol{X}$ that minimises the function $f(x)$

$$x^* = \arg\min_{x \in X} f(x) \tag{1}$$

Currently, very interesting technique belonging to the EAs is the Differential Evolution, which was used in experiments described in this paper.

3. Differential Evolution

Differential Evolution is a relatively new metaheuristic based on the principle of EA. Results of studies (see e.g. [23, 26]) indicate very good effects in the case of multidimensional, real-parameter optimisation problems [26]. DE utilizes Np n-dimensional (where n is a dimensionality of optimised function) real-parameter vectors (referred here as individuals) as a population and is based on a very simple crossover and mutation operators, and requires a small number of parameters. In addition to the values of a mutation scaling factor $F \in \mathbb{R}$ (usually $F \in [0, 2]$) and crossover coefficient $Cr \in \mathbb{R}$ ($Cr \in [0, 1]$), the population size Np and termination conditions, including the maximum number of generations $t_{\max}$, should be determined (which is typical for this class of algorithms).

The basic variant of DE [28] (for modifications, called schemes, see e.g. [19, 27, 28]), was used in this paper. In generations 0 (randomly initialised) to $t_{\max}$ for each individual x_i, $i = 1, 2, ..., Np$, the new candidate solutions c_i for subsequent generations are created by randomly (with probability Cr) combining two specimens:

(1) candidate proposal: $y_i = x_{r1} + F(x_{r2} - x_{r3})$, where x_{r1}, x_{r2}, x_{r3} are random, distinct specimens from the present generation, and F is a parameter describing the impact of differential vector $(x_{r2} - x_{r3})$ on an individual x_{r1};
(2) the individual x_i,

i.e.:

$$c_{i,j} = \begin{cases} y_{i,j}, & \text{if } U_j(0,1) \le Cr \text{ or } j = R_i \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{2}$$

where $c_{i,j}$, $y_{i,j}$, $x_{i,j}$ denotes, respectively, j-th element of i-th candidate solution, candidate proposal and individual, $R_i \in 1, 2, ..., n$ is a randomly chosen index ensuring that c_i is distinct

from x_i in at least one component and $U_j(0, 1)$ denotes uniformly distributed random number over [0, 1].

Then, in the process of selection, a better (in terms of optimised function) specimen of the two, the individual x_i or the solution candidate c_i, is chosen to the next generation.

Even though DE requires such a small number of parameters, the optimal choice of their values causes problems, therefore in recent years intensive studies have been carried out on the mechanisms of adaptation and self-adaptation of the parameters of EAs, including the DE.

4. Self-adaptation

The concepts of adaptation and self-adaptation need to be clarified. Both mechanisms are intended to exempt the user from selecting suitable values of parameters, however, they differ significantly. The adaptation of the parameters is based on a priori assumptions, such as the observation that in the initial stage of the optimisation it is worth to pay more attention to search the whole solution space (exploration), then to perform the fine tuning of best solutions (exploitation) [2, 12]. Although this observation seems to be well founded, this mechanism does not take into account many important factors and conditions of the optimised problem.

Self-adaptation is based on the assumption that the adjustment of the parameters values should be based on the current state (or change of state) of optimisation process, the rate of convergence, or quality improvement of the results. This allows incorporating of information about the problem and algorithms behaviour into the adaptation process. The parameter values are often encoded in the individuals' genomes and evolutionary modified, such approach is presented e.g. in [1, 8, 22, 33].

Population size (self-)adjustment constitutes a particularly noteworthy branch of research on adaptation/self-adaption of algorithms. Its purpose is to: relieve the user of the necessity to choose proper size of the population; create a mechanism to avoid stagnation in the local minima without having to introduce new genetic operators; or limit the computational requirements of the algorithm. This paper addresses the latter of these objectives, which can be easily associated with the former two.

5. Population size (self-)adjustment

One of the most widely discussed schemes of population size reduction is the reduction of its population by half at each certain, predetermined, number of generations. This method is associated with the aforementioned observation, concerning the exploration and exploitation. In this case, most frequently used selection method relies on division of the current population into two parts and selection of better individuals of the resulting pairs [6, 7]. Based on this observation more complex mechanisms of self-adaptation of population size is implemented, such as in [34], where the population size is increased and reduced depending on whether the algorithm currently focuses on exploration or on exploitation.

In the second approach utilising the same observation, the population size adjustment is based on the degree of improvement of the quality of individuals [11, 13, 25, 30]. It is assumed

that when considerable improvement occurs it is worth to increase the size of population, because the algorithm is still in the stage of exploration, while when the weakening of the improvements is observed, the number of individuals should be reduced and the exploitation of the best individuals' vicinity should begin.

Self-adjusting of the population size is also often implemented using several coexisting populations of different sizes, which exchange the information about the quality of its individuals and the size of their population [17, 24].

Another popular approach, mentioned earlier in the introduction to this section, is based on storing the information about the parameters (including the size of the population) in the genome and their evolutionary adaptation [18, 29]. However, it must be noted that it is not certain that the individuals with best fitness values comprise best population sizes.

Similarly, attempts were made to relieve completely the user of the necessity to choose the size of the population [3, 15, 20], however, cited works use only the mechanism of increasing the size of the population after reaching convergence, which is not the aim of this study.

6. Proposed method of population size reduction

The proposed method of population size reduction is based on clustering technique [32]. The task of clustering is to divide a set of objects into a certain number of subsets (clusters) with the assumption that a single cluster contains objects similar to each other as much as possible, taking into account that the objects assigned to different clusters should differ significantly from each other. More formal definition of clustering task could be found, among others, in [23]. Main applications of clustering are primarily an unsupervised classification and preliminary data processing.

In the proposed solution individuals constituting the population of EA (in case of this study the DE) undergo at some generation(s) a reduction step, i.e. a process of clustering (without specifying the clustering algorithm) with number of clusters generated lower than actual population size, which leads to reduction of the number of individuals in the generation (algorithm 1).

In the implementation adopted in this paper the best specimens from each of the resulting clusters are transferred to the next generation. However, selecting an individual transferred (cluster representative) is the key element of the method. In addition to aforementioned, also centroids of clusters – though this requires additional calculations of the quality function – or individuals nearest to the obtained clusters' centres (medoids) may be transferred to the next generation. This issue requires further study and will not be considered here.

The proposed method does not solely focus on optimising the quality of individuals, but also on preserving their variety. This allows the diversity of population to be preserved, whose loss is likely to affect methods using elitist selection mechanism.

However, to ensure some elitist nature of the DE after the reduction step, chances of choosing each individual to recombination/crossover step (the x_{r1}, x_{r2}, x_{r3} specimens) are modified such that individuals representing bigger clusters will be chosen with higher probability (it is assumed that higher concentrations of individuals should be formed near the local and global minima).

Algorithm 1

Differential Evolution with clustering based population size reduction (DEc)

Procedure DEc
1. Set iteration number $t \leftarrow 0$;
2. Initialise population pop_0 with Np random individuals x_i, $i = 1, 2, ..., Np$;
3. Evaluate population pop_0 (evaluate $f(x_i)$, $i = 1, 2, ..., Np$);
4. Repeat until stop condition
   4.1. t++;
   4.2. If $t = t_{red} \rightarrow$ perform population size reduction;
      4.2.1. Generate $k < Np$ clusters;
      4.2.2. Select cluster representatives → create new population pop_{t-1};
   4.3. For each $x_i \in pop_{t-1}$;
      4.3.1. Mutation → create candidate proposal y_i;
      4.3.2. DE-crossover → create candidate solution c_i;
      4.3.3. Evaluate $f(c_i)$;
      4.3.4. Selection:
         If $f(c_i) > f(x_i) \rightarrow pop_t(i) = c_i$;
         Else $pop_t(i) = y_i$;
5. Return x_{best}

7. Computational example

To test described reduction method, a set of 24 real-parameter single-objective functions of the BBOB 2012 noiseless testbed [14] was used as a benchmark. In preliminary study 2-, 3-, 5-, 10- and 20-dimensional instances were tested, each with 1000 repetitions to reduce the impact of the stochastic nature of the DE. 10- and 20-dimensional instances turned out to be too time consuming for basic, unoptimised DE implementation and test platform used – in case of more than 50% of functions DE could not reach even a local minimum in a reasonable time, thus results were meaningless and those dimensionalities are not further discussed.

A comparative study was carried out for preliminary experiments on the proposed method. As mentioned in introduction, investigated DE implementation (denoted here DEc for simplicity) was compared with another two instances of DE. Population size in one of them was set to $3/4 \cdot Np_{initial}$, corresponding to the number of function evaluations in DEc, i.e. for half of iterations the population size is $Np_{initial}$, and for the rest is $1/2 \cdot Np_{initial}$, leading to $(1/2 + 1/2 \cdot 1/2) \cdot Np_{initial} = 3/4 \cdot Np_{initial}$. In all instances the basic variant of DE was used as described in section 3. In all computations population size $Np_{initial}$ was set to $5 \cdot n$, parameters F and Cr to 0.8 and 0.5 respectively (values proposed as a rule of thumb in [28], which coincide with our previous study [10]) and 1000 runs of algorithm were performed (results and plots presented below refer to values averaged over those runs).

For clustering used in the introduced population size reduction method the classic k-means algorithm [21] was used. The number of resulting clusters was set to $Np_{initial}/2$ (rounding up), thus population size was reduced by half. In order to simplify the procedure the knowledge of optimal values f_{opt} of all functions were assumed and only one reduction step was carried

out at $t_{red} = t_{target}/2$ generation (t_{target} denotes the average number of generations at which DEs reaches $f_{target} = f_{opt} + 10^{-6}$ or converges during preceding trial runs). Values of t_{target} were also used as termination conditions (combined with f_{target}) for DEs and DEr.

Results show that DEc achieves an error lower than DEs by 25,15% on average (see Table 1), at the same time reducing the number of function evaluations by 25%, which lowers computational requirements if the clustering procedure is well-implemented. In contrast, DEr with reduced population size (same number of function evaluations as DEc) gives an average error higher by 96.87% compared to DEs (Table 1). Better results of DEc in comparison to DEs were achieved primarily through improved performance on ill-conditioned functions and functions with disturbed sensitiveness along one dimension (functions no. 2 – 4, 6, 8 – 10, 13, 14 and 17). This property will undoubtedly be the subject of future research.

Table 1

Absolute (DEs) and relative (DEr and DEc) errors of investigated DE implementations

DIM	DE_{std}	DE_{red} [%][2]	DE_{clust} [%]
2	25.8450	238.7389	107.8783
3	11.7058	360.7053	143.8560
5	12.9651	266.1109	161.6147

For DEs $\Sigma_{funcID}(f_{opt} - \overline{f_{best}})$, where $\overline{f_{best}}$ are averaged best values returned, are given. For DEr and DEc mean (and median) percentages in relation to DEs errors are shown.

Study indicates that rates of convergence of DEs and DEc are similar (DEc perform slightly better on ill-conditioned functions, however DEs compensates on the rest), while DEr converge slightly slower (see Fig. 1 for plot for 2-dimensional[3] function no. 13 and appendix B for additional graphs for selected 2-dimensional functions). Particularly noteworthy is that for most functions averaged best values of the DEr population are significantly worse than corresponding values of DEs and DEc. Differences between latter two arise mainly due to random initialisation of populations.

Summarising, results obtained coincide with previous study [10] and indicate that:

– proposed reduction method improves returned values of f_{best}, especially on ill-conditioned functions,
– rate of convergence of DEc was similar to DEs (thus better than DEr),
– results obtained by DEr were noticeable worse than the other two.

8. Conclusions

On one hand, this paper is a survey of self-adaptation methods of Evolutionary Algorithms – in particular, the population size reduction methods are covered, which constitute a promising field of research due to necessity of reducing the computational requirements. On the other, it presents preliminary results of the new, clustering-based method of population size reduction.

[2] In 2- and 3-dimensional cases functions no. 2 and 6 (2-dim.) and no. 2 (3-dim.) has been excluded from the average due to very small values of errors, which resulted in outlying relative error values.

[3] For 3- and 5-dimensional instances rates of convergence (and hence plots) are similar.

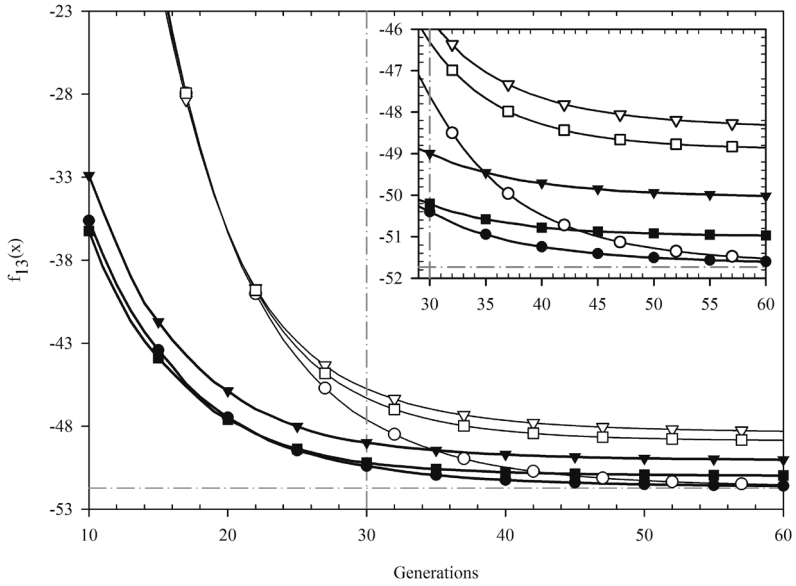

Fig. 1. Mean (empty markers) and best function values of population members for 2-dimensional instance of benchmark function no. 13. Horizontal and vertical lines mark optimal function value and population size reduction generation, respectively. Legend: —●— DEc, —■— DEs, —▼— DEr

Rys. 1. Średnie (puste znaczniki) oraz najlepsze wartości funkcji osobników populacji dla dwuwymiarowego przypadku funkcji nr 13. Poziome i pionowe linie określają, odpowiednio, wartość optymalną funkcji oraz generację, w której nastąpiła redukcja liczności populacji. Legenda: —●— DEc, —■— DEs, —▼— DEr

It is worth mentioning once again, that presented results were obtained by using one of the simplest algorithms for clustering, the *k*-means. Moreover, the simplest scheme of reduction was assumed: only one reduction step was carried out and population was reduced by half. Nevertheless, results achieved were better than standard DE implementation, which require 33% more evaluations of individuals.

Obtained results indicate that the proposed method is a promising direction of research and thus will be developed in the future, particularly in terms of:

– selection of the optimal number of individuals in the population remaining after reduction step;
– determination of the best generation(s) for reduction step(s);
– choice of the form of specimens transferred to the next generation after reduction step (i.e. centroids, medoids, individuals with lowest function value, etc.).

Simultaneously, studies over other methods of determining which individuals can be removed from the population will be carried out.

This contribution is partially supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007–2013 and European Regional Development Fund.

References

[1] Abbas A., *The self-adaptive pareto differential evolution algorithm*, Proceedings of the IEEE Congress on Evolutionary Computation, 2002, 831-836.

[2] Arabas J., Michalewicz Z., Mulawka J., *GAVaPS – a genetic algorithm with varying population size*, Proceedings of the 1st IEEE Conference on Evolutionary Computation, 1994, 73-78.

[3] Auger A., Hansen N., *A restart CMA evolution strategy with increasing population size*, Proceedings of the 2005 IEEE Congress on Evolutionary Computation, 2005, 1769-1776.

[4] Bäck T., Fogel D.B., Michalewicz Z., *Evolutionary Computation 1: Basic Algorithms and Operators*, Taylor & Francis, 2000.

[5] Blum C., Chiong R., Clerc M., De Jong K., Michalewicz Z., Neri F., Weise T., *Evolutionary optimization*, [in:] *Variants of Evolutionary Algorithms for Real-World Applications*, Chiong R., Weise T., Michalewicz Z. (Eds.), Springer, 2012, 1-19.

[6] Brest J., Maučec M.S., *Population size reduction for the differential evolution algorithm*, Applied Intelligence, vol. 29, 2008, 228-247.

[7] Brest J., Maučec M.S., *Self-adaptive differential evolution algorithm using population size reduction and three strategies*, [in:] *Soft Computing – A Fusion of Doundations*, Methodologies and Applications, vol. 15, 2011, 2157-2174.

[8] Brest J., Zamuda A., Bošković B., Greiner S., Žumer V., *An analysis of the control parameters' adaptation in DE*, [in:] *Advances in Differential Evolution*, Chakraborty U.K. (Eds.), 2008, 89-110.

[9] Diaz-Gomez P.A., Hougen D.F., *Initial population for genetic algorithms: A metric approach*, Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, 2007, 43-49.

[10] Dziedzic M., *Differential Evolution with Population Size Reduction*, Proceedings of The First International Conference on Automatic Control and Information Technology, vol. „Streszczenia", CD (9 pages), 2011, 28.

[11] Eiben A.E., Marchiori E., Valko V.A., *Evolutionary algorithms with on-the-fly population size adjustment*, [in:] *Parallel Problem Solving from Nature*, Yao X., Burke E.K., Lozano J.A., Smith J., Merelo-Guervos J.J., Bullinaria J.A., Rowe J.E., Tino P., Kabán A., Schwefel H.-P. (Eds.), 2004, 41-50.

[12] Fernandes C., Rosa A., *A study on non-random mating and varying population size in genetic algorithms using a royal road function*, Proceedings of the 2001 Congress on Evolutionary Computation, 2001, 60-66.

[13] Fernandes C., Rosa A., *Self-regulated population size in evolutionary algorithms*, [in:] *Parallel Problem Solving from Nature*, Runarsson T.P., Beyer H.-G., Burke E., Merelo-Guervos J.J., Whitley L.D., Yao X. (Eds.), 2006, 920-929.

[14] Finck S., Hansen N., Ros R., Auger A., *Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions*, Working Paper 2009/20, 3rd GECCO Workshop for Real-Parameter Optimization, 2012.

[15] Harik G.R., Lobo F.G., *A parameter-less genetic algorithm*, Proceedings of the 1999 Genetic and Evolutionary Computation Conference, 1999, 258-265.

[16] Hill R.R., *A Monte Carlo study of genetic algorithm initial population generation methods*, Proceedings of 1999 Winter Simulation Conference, 1999, 543-547.

[17] Hinterding R., Michalewicz Z., Peachey T.C., *Self-adaptive genetic algorithm for numeric functions*, [in:] *Parallel Problem Solving from Nature*, Voigt H.-M., Ebeling W., Rechenberg I., Schwefel H.-P. (Eds.), 1996, 420-429.

[18] Iacca G., Mallipeddi R., Mininno E., Nerif., Suganthan P.N., *Super-fit and population size reduction in compact differential evolution*, Proceedings of 2011 IEEE Workshop on Memetic Computing, 2011, 1-8.

[19] Kaelo P., Ali M.M., *A numerical study of some modified differential evolution algorithms*, European Journal of Operational Research, 169(3), 2006, 1176-1184.

[20] Lima C.F., Lobo F.G., *Parameter-less optimization with the extended compact genetic algorithm and iterated local search*, „Genetic and Evolutionary Computation, Part I", Deb K. (Eds.), 2004, 1328-1339.

[21] MacQueen J.B., *Some methods for classification and analysis of multivariate observations*, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967, 281-297.

[22] Omran M.G.H., Salman A., Engelbrecht A.P., *Self-adaptive differential evolution*, [in:] *Computational Intelligence and Security*, Hao Y., Liu J., Wang Y.-P., Cheung Y.-M., Yin H., Jiao L., Ma J., Jiao Y.-C. (Eds.), vol. 3801, 2005, 192-199.

[23] Paterlini S., Krink T., *Differential evolution and particle swarm optimization in partitional clustering*, Computational Statictics & Data Analysis, vol. 50, 2006, 1220-1247.

[24] Schlierkamp-Voosen D., Muhlenbein H., *Strategy adaptation by competing subpopulations*, [in:] *Parallel Problems Solving from Nature*, Davidor Y., Schwefel H.-P., Männer R. (Eds.), 1994, 199-208.

[25] Shi X.H., Wan L.M., Lee H.P., Yang X.W., Wang L.M., Liang Y.C., *An improved genetic algorithm with variable population size and a pso-ga based hybrid evolutionary algorithm*, Proceedings of the 2nd International Conference on Machine Learning and Cybernetics, 2003, 1735-1740.

[26] Storn R., *Differential evolution homepage* (Online: 2011-11-20), http://www.icsi.berkeley.edu/~storn/code.html

[27] Storn R., *On the usage of differential evolution for function optimization*, Biennial Conference of the North American Fuzzy Information Processing Society, 1996, 519-523.

[28] Storn R., Price K., *Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces*, Journal of Global Optimization, vol. 11, 1997, 341-359.

[29] Teo J., *Exploring dynamic self-adaptive populations in differential evolution*, Soft Computing – A Fusion of Foundations, Methodologies and Applications, vol. 10, 2006, 673-686.

[30] Tirronen V., Neri F., *Differential evolution with fitness diversity self-adaptation*, [in:] *Nature-inspired Algorithm for Optimisation*, Chiong R. (Eds.), 2009, 199-234.

[31] Toğan V., Daloğlu A.T., *An improved genetic algorithm with initial population strategy and self-adaptive member grouping*, Computers & Structures, vol. 86, 2008, 1204-1218.

[32] Xu R., Wunsch D., *Clustering*, Wiley-IEEE Press, 2008.

[33] Zamuda A., Brest J., Boskovic B., Zumer V., *Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution*, Proceedings of the IEEE Congress on Evolutionary Computation, 2008, 3718-3725.

[34] Zhuang N., Benten M.S., Cheung P.Y., *Improved variable ordering of BDDS with novel genetic algorithm*, Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 3, 1996, 414-417.

Appendix A: Maximum numbers of generations for benchmark functions

Table A.1

Maximum numbers of generations for benchmark functions.

2-dimensional instances

funcID	1	2	3	4	5	6	7	8	9	10	11	12
$t_{\max}$	36	53	52	53	4	88	29	78	115	57	58	58
funcID	13	14	15	16	17	18	19	20	21	22	23	24
$t_{\max}$	59	58	37	32	53	40	27	46	36	42	24	26

3-dimensional instances

funcID	1	2	3	4	5	6	7	8	9	10	11	12
$t_{\max}$	59	89	91	99	6	128	55	92	118	57	57	60
funcID	13	14	15	16	17	18	19	20	21	22	23	24
$t_{\max}$	102	85	49	38	90	62	39	65	52	59	36	39

5-dimensional instances

funcID	1	2	3	4	5	6	7	8	9	10	11	12
$t_{\max}$	114	172	173	189	9	334	110	209	276	70	58	117
funcID	13	14	15	16	17	18	19	20	21	22	23	24
$t_{\max}$	140	145	77	61	162	111	65	93	92	83	59	69

Values of t_{target} are averaged numbers of generations at which DEs reaches termination condition, i.e. value of $f_{target} = f_{opt} + 10^{-6}$, where f_{opt} is the minimum of the function; or the algorithm converges in local minimum.

Appendix B: Graphs of function values for 2-dim. instances of benchmark functions

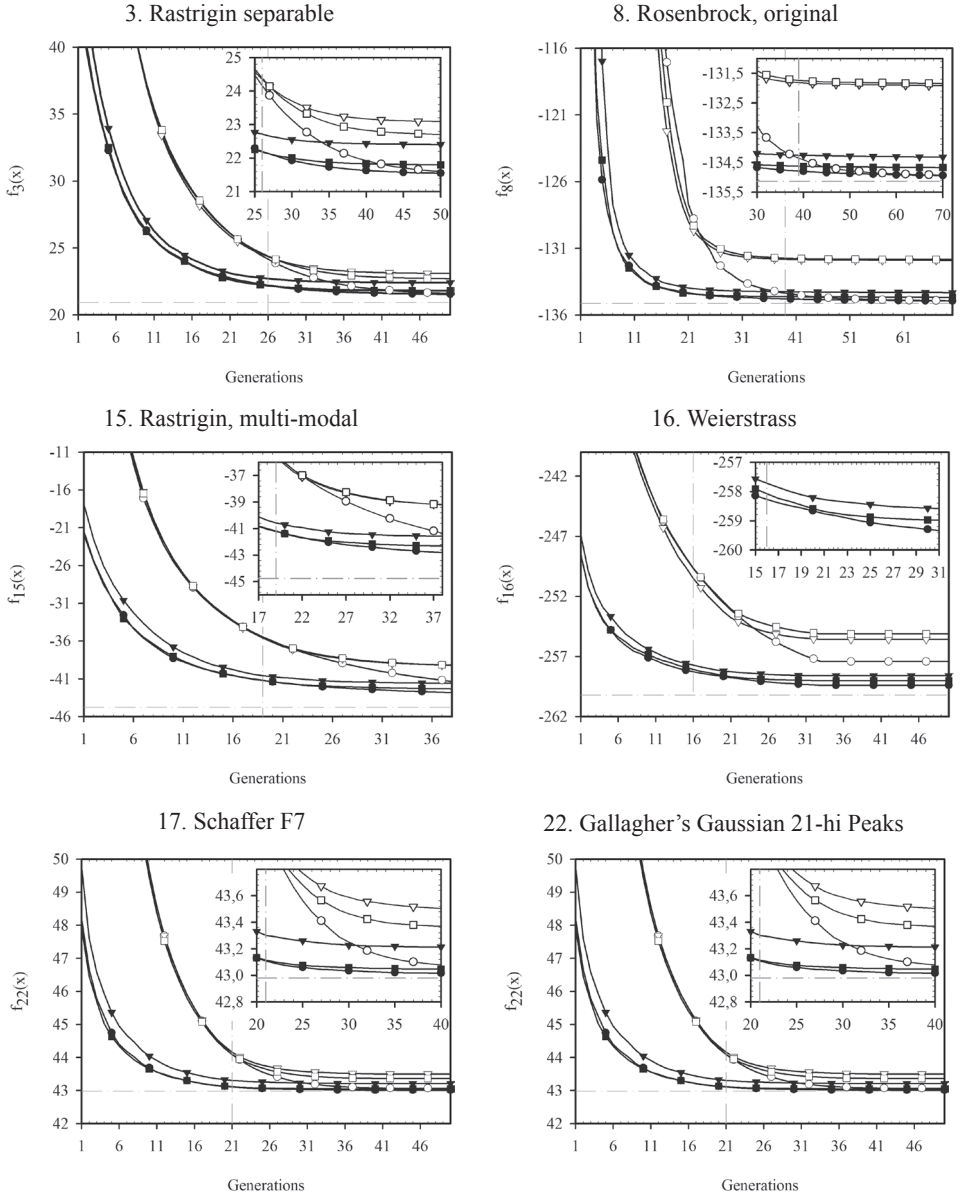

Fig. B.1. Mean (empty markers) and best function values of population members for 2-dimensional instances of benchmark functions. Horizontal and vertical lines mark optimal function value and population size reduction generation, respectively. Legend: DEc, DEs, DEr

Rys. B.1. Średnie (puste znaczniki) i najlepsze wartości funkcji osobników populacji dla dwuwymiarowych przypadków funkcji testowych. Poziome i pionowe linie określają, odpowiednio, wartość optymalną funkcji oraz generację, w której nastąpiła redukcja liczności populacji. Legenda: DEc, DEs, DEr