Paper

Evolutionary Computation for Global Optimization – Current Trends

Jarosław Arabas

Institute of Electronic Systems, Warsaw University of Technology, Warsaw, Poland

Abstract—This article comments on the development of Evolutionary Computation (EC) in the field of global optimization. A brief overview of EC fundamentals is provided together with the discussion of issues of parameter settings and adaptation, advances in the development of theory, new ideas emerging in the EC field and growing availability of massively parallel machines.

Keywords—evolutionary computation, global optimization.

1. Evolution of Evolutionary Computation

Since 1980'ies we have been witnessing a growing popularity of a family of algorithmic techniques which originated in 1960'ies and have been inspired by findings in fields of genetics and natural evolution. Although many pioneering approaches had been introduced (see [1] for an overview), only several survived and are nowadays called the Evolutionary Computation (EC) [2]. Until mid 1990'ies, the mainstream EC work was presented at three major conferences: International Conference on Genetic Algorithms (ICGA), Parallel Problem Solving from Nature (PPSN) and Workshop on Foundations of Genetic Algorithms (FOGA). In 1994 the IEEE Congress on Evolutionary Computation (CEC) was started, in 1995 the Genetic Programming (GP) conference was launched, and the year 1999 witnessed the birth of the annual Genetic and Evolutionary Computation Conference (GECCO) which combines the ICGA and the GP in one event. Two recognized international journals publish works on the EC: since 1993, the MIT Press has been releasing the *Evolutionary Computation* journal and the IEEE has been publishing the *Transactions on Evolutionary Computation* since 1997. The Polish accent is the annual National Conference on Evolutionary Computation and Global Optimization which started in 1996[1].

The idea behind the EC is quite straightforward – take a population of points from some search space, assign them numbers that reflect their probability to survive the selection process, and perform a randomized selection. The selected points undergo a randomized variation, yielding a new population of points, and the process is iterated many times. Despite of the simplicity of the idea, several named approaches have been defined that usually differ only in small details. A newcomer to the EC field may be greatly surprised by recognizing that it includes:

- Differential Evolution,
- Evolution Strategies,
- Evolutionary Programming
- Genetic Algorithms,
- Genetic Programming,
- Memetic Algorithms,

to mention only few important branches in an alphabetical order.

When studying EC methods one has to get used to the metaphor of genetics and evolution, which strongly influenced the vocabulary. Instead of points from the search space we speak of *chromosomes* (or *individuals*), instead of the objective function to be optimized we speak of the *fitness function* which defines the selection probability, variation of points is performed by the *genetic operations* which are called *crossover* and *mutation*, etc.

A popular opinion about EC is that an important factor that attracted researches to take a closer look at the EC is the appealing metaphor. Perhaps this observation motivated researchers to look for other metaphors from the nature, and since late 1990'ies we have been observing a tendency to introduce various population based techniques which share the idea of selection and variation, but they are named (in an alphabetical order):

- Artificial Immune Systems [3], [4],
- Estimation of Distribution Algorithms [5],
- Particle Swarm Optimization [6],

to mention a few representatives. A common name of *metaheuristics* has been suggested for the EC and the aforementioned techniques to avoid a naming burden [7], [8].

In this article it is attempted to comment on the current state of the EC, which is a very hard task and will be always more or less subjective. Therefore the bibliography will be presented that provides more detailed descriptions of ideas that have been roughly sketched in the text. Much more detailed general presentation of EC methods can be found *inter alia* in [2], [9], [10], [11].

The paper is composed of four sections. The first section briefly comments on the history of EC development. Section 2 overviews the taxonomy of optimization tasks that

[1] In year 2011 the conference was organized in Warsaw by the Warsaw University of Technology and Cardinal Stefan Wyszynski University.

are considered along with the EC and outlines the algorithm of a typical EC method for global optimization. In Section 3 it is attempted to define main lines of the EC development for global optimization that can be found in the literature. Section 4 presents concluding remarks and outlooks possible future development trends.

2. Evolutionary Computation and Optimization

2.1. Taxonomy of Optimization Tasks

Current applications of the EC are usually related to optimization of various kind. With D let us denote the domain of a problem to be solved (i.e., the set of all possible representations of the solution). The domain contains points that are feasible, i.e., any infeasible point is definitely not a solution. They can be evaluated using the objective function $q : F \to R$, where $F \subseteq D$ is the set of feasible solutions. Then the optimization task is defined as the task to find either the global minimum or any local minimum of the objective function. Depending on the domain type we can distinguish:

- combinatorial optimization when the domain is countable or finite, e.g., D is the set of binary vectors, the set of permutations or the set of graphs,
- continuous optimization when the domain is R^n.

If $F \neq D$ then the optimization task is a constrained one. Some researchers consider the multimodal optimization where in addition to the global optimum, the solver is expected to find as many local optima as possible. For an overview of optimization problems and methods with a particular stress on the global optimization the reader is referred to [12]–[14].

Another generalization of the optimization task which has gained a growing attention for past 10 years in the EC community is the multiobjective optimization where, instead the objective function, a mapping is considered $\mathbf{q} : F \to R^m$. Then the task is to find nondominated points, where a point $\mathbf{x} \in F$ is called nondominated when there exists no other point $\mathbf{y} \in F$ such that for all $i = 1, ..., m \quad q_i(\mathbf{y}) \leq q_i(\mathbf{x})$ and for some j it holds $q_j(\mathbf{y}) < q_j(\mathbf{x})$.

From the historical perspective, early EC methods have not been designed as optimization tools (cf. a famous statement "Genetic algorithms are not function optimizers" by Kenneth de Jong [15]). They have been viewed in terms of adaptation which is less strict and formalized than optimization. Moreover it is even not necessary to explicitly define any objective function, e.g., in the tournament selection, it is only needed to compare two points to choose the better one. Still it has become a common practice to use EC methods as optimization tools. This practice has been criticized by Wolpert and Macready in their famous No Free Lunch Theorem (NFL) for optimization [16]. They claimed that no search method would perform consistently better than any other if one considers all objective functions which can be defined in the search space. Publication of NFL was initially somewhat shocking for researchers working on the EC development since its naive interpretation led to the conclusion that no real development is possible. This interpretation of NFL was then criticized by authors who showed that although NFL is correct, it does not necessary mean that all algorithms are equally good for subsets of problems defined in the search space [17].

EC has a strong relationship to artificial intelligence (AI). Maybe the most popular application of EC methods in AI is the Neural Network (NN) training process. The link between the EC and NNs is so strong that it gave an inspiration to establish in 1993 the International Conference on Genetic Algorithms and Neural Networks (ICANNGA). If we take a perspective that the optimization task consists in learning the global optimum using the experience gathered from previously generated points, then we can agree that EC can be classified as an AI method [18].

Stress on optimization properties of the EC has been increasing during their development. It seems that this is a natural consequence of the tendency to apply EC methods in practice. It is also possible to theoretically analyze behavior of an EC method using the same terms as introduced for well-established optimization tools. Thus the theoretical analysis of EC methods often concentrates on the convergence in a weak sense [19], [20].

Development of EC methods as optimization tools has been facilitated by benchmark functions; some of them have been introduced even in mid-1970'ies [21]. Until very recently there was however no clear agreement about the testing procedure, which made the published results hardly comparable. Few years ago there have been two benchmark sets introduced which define a standardized testing procedure and criteria for evaluating optimization methods based on statistics of results from multiple independent runs of each method [22], [23]. It should be however noted that, although the benchmarking process is usually limited to search spaces containing either binary or real vectors, there are many EC applications where specific nonstandard representations are processed [11]. It seems that the ability to process such nonstandard representations is one of major advantages of EC.

2.2. Evolutionary Algorithm for Global Optimization

A typical Evolutionary Algorithm (EA) is depicted in Fig. 1. State of the algorithm in the iteration number t is defined by the population P^t which contains μ points; the i-th point is labeled with P_i^t. In each iteration a population O^t is created by mutation and crossover of points selected from P^t. Population P^{t+1} is defined by a replacement procedure that either accepts points from O^t only or allows to pass some points from P^t. Selection of points is a random process which favors better ones, i.e., points with a lower objective function value (in the minimization case) will be selected with a higher probability than others. In the replacement phase the new population is defined by

selecting points from O^t and P^t. In generational EAs, the replacement is defined simply as $P^{t+1} \leftarrow O^t$ and in elitist EAs a number of best points from P^t can be preserved.
When the EA is to solve the global optimization problem, typical mutation is performed as $O_i^t \leftarrow P_k^t + \mathbf{z}$ where $\mathbf{z}$ is a random variate which is normally distributed with zero mean vector and a covariance matrix $\mathbf{C}$. Typical crossover consists in averaging points which results in the following formula for the "mutation and crossover" operation

$$O_j^t \leftarrow \left(\mathbf{w} \cdot P_k^t + (\mathbf{1} - \mathbf{w}) \cdot P_l^t\right) + \mathbf{z},$$

where $\mathbf{w}$ is a vector of weights such that $0 \leq w_j \leq 1$ for all j, and the symbol '$\cdot$' stands for the component-wise product such that $\mathbf{a} \cdot \mathbf{b}$ yields a vector $\mathbf{c}$ and $c_j = a_j b_j$. When $w_j = 1/2$ we speak of the arithmetic crossover. When w_j is a random variate with the Bernoulli distribution we speak of the binomial crossover, and when the probabilities of getting 0 and 1 are equal we speak of the uniform crossover. There are no good indications about the stop criterion since there are no theoretical findings about the EA convergence speed that can be applied for sufficiently general classes of problems (but there exists the convergence speed analysis for the parabola function provided e.g. in [24], [25]). For this reason the stop criterion is usually based on the time budget.

```
t ← 0
P^0 ← initialization()
repeat
  for all i = 1,...,μ do
    if U(0,1) < p_c then
      k,l ← select from (1,...,μ)
      O_i^t ← mutation and crossover(P_k^t, P_l^t)
    else
      k ← select from (1,...,μ)
      O_i^t ← mutation(P_k^t)
    end if
  end for
  P^{t+1} ← replacement(P^t, O^t)
  t ← t + 1
until stop condition satisfied
```

Fig. 1. Outline of an example evolutionary algorithm for global optimization.

The EA is a random procedure, therefore its action can be described in terms of the probability theory and statistics. For binary encoded EAs there is a well developed and widely accepted theory which analyzes populations P^t as a Markov chain [26]. It is possible to analyze statistics of populations, e.g., to check if the probability of hitting the global optimum at least once increases with the generation index or to test if the most frequent population contents will contain the global optimum. Another approach to analyze binary encoded EAs is presented by the schema analysis which analyzes schemata – sets of solutions defined by similarity patterns. It is argued that schemata with over-average fitness are expected to increase their number of representatives in subsequent populations.
In real coded EAs which are used to perform global optimization it has been proved that weak convergence will be observed if it is possible with a nonzero probability that the EA will generate a finite series of points starting from any feasible point and terminating in a neighborhood of any other feasible point, provided that the neighborhood is a nonzero measure set [19]. This result, although important, does not give any information about the dynamics of the population contents. This can be achieved by applying an analysis of the population distribution dynamics that assumes an infinite population model introduced by Qi and Palmieri [27]. This model has been revisited by Karcz-Duleba [28], [29] and Arabas [30] who derived formulas for the limiting values of the population variance in the search space for typical selection methods, Gaussian mutation, arithmetic crossover and elite factor.

3. Selected Topics in Development of EC Methods for Global Optimization

Various mutation types. Gaussian mutation is a very popular choice when using the EA for global optimization. The normal distribution is however "thin tailed" and computer realizations of the normal random variable usually cannot generate points located further than, say, 5 times the standard deviation from its mean. This fact motivated researchers to look for other mutation schemes and in the results several alternative definitions have been introduced, including α-stable mutations and Differential Evolution.

The α-stable distribution has a property that its probability density function in the tail can be approximated by a function proportional to $\exp(-x^\alpha)$. It is stable in a sense that the sum of α-stable distributed variates is also α-stable distributed. Value of $\alpha = 2$ corresponds to the normal distribution. When $\alpha < 2$ the distribution becomes "fat tailed" which means that when α decreases it will be more and more probable to generate a variate from the tail, which may significantly increase the population diversity. A detailed discussion of α-stable mutation is provided in [31].

Differential Evolution (DE) is sometimes classified as a separate metaheuristic type but the type of variation allows to consider it as an approach to the mutation adaptation. The most innovative idea introduced in DE is to perform mutation according to the formula

$$O_i^t = P_j^t + F\left(P_k^t - P_l^t\right),$$

where P_j^t is the mutated point, F is a parameter called the scaling factor and k and l are indices of points from P^t that have been selected with the uniform distribution in $\{1,...,\mu\}$. Thus the distribution of mutants depends on the distribution of population P^t, which allows for its adaptation to the fitness function shape. DE differs also in the way of organizing relations between crossover and mutation. More details on DE can be found in [32].

Compact EAs and Estimation of Distribution Algorithms. In a classical EA the algorithm state is defined by the population contents, and the EA action can be described by the sampling distribution – the probability distribution that describes the process of generating one point from the population O^t. In compact EAs, as well as in Estimation of Distribution Algorithms (EDAs), the sampling distribution takes the role of the state definition and defines in the same time the process of generating points. In the compact EA in each iteration a small number of points (usually one or two) are generated and the sampling distribution is updated. In the EDA a number of points is generated and the new sampling distribution is estimated from them.

The most successful representative of the EDA line of algorithms is the Covariance Matrix Adaptation Evolution Strategy (CMAES) [33]. According to results of benchmarking on BBOB'09 and CEC'05 [22], [34], CMAES seems to be one of the most successful methods in EC. In CMAES the sampling distribution is normal and it is represented by the mean vector $\mathbf{m}$ and the covariance matrix C. In the iteration t the sampling distribution is used to generate the population P^t. Then a population $e(P^t)$ being a fraction of best points from P^t is selected which is used to compute the mean vector $\mathbf{m}' = E(e(P^t))$ and the matrix $C' = E((P^t - \mathbf{m}')^2)$. Values of $\mathbf{m}'$ and C' are used to update the parameters $\mathbf{m}$ and C for the next iteration.

Tuning parameter values. Practitioners which are potential users of optimization methods usually dream of a single magic button to press and get the problem solved. Therefore the chance that an optimization method will become an acceptable tool for practitioners increases when the method assumes smaller number of parameters. Unfortunately, basic EAs are characterized by many parameters with no clear intuition about their relation with efficiency. For example in the standard EA presented in Fig. 1 the user has to set the values of the population size, the covariance matrix of the Gaussian mutation, the crossover probability and the elite fraction. To reduce the set of user-defined parameters, diversity of adaptation techniques have been developed – see e.g. [35], [36] for an overview. Note that the DE and CMAES fit into the line of adaptive methods to tune the mutation distribution.

Among currently popular approaches to parameter tuning one can distinguish two dominating types. The first one is self-adaptation, where each point is coupled with the algorithm's parameter values. During mutation, the parameter values are also mutated. Is is believed that appropriate parameter values will be more frequently accompanying points that are better than average and therefore they will be reinforced by the selection process. Another popular approach is based on an ensemble of a finite number of parameter settings. Every settings is evaluated by looking at the average increase of quality of points before and after applying transformations defined by the parameter settings. Choice of the parameter settings is randomized and is influenced by their cumulated evaluation – values of parameters that are on the average better than others are more likely to be chosen to perform the transformation.

Hierarchy of metaheuristics. As mentioned in the introductory section there is a variety of approaches in domains of the EC and metaheuristics. Therefore a tendency has appeared to consider hybrid techniques of various kind. Generally speaking, the hybridization may be either generic or hierarchical. In the first case, elements from one metaheuristic may change the algorithm of the other. A good example is to introduce to the standard DE a self-adaptation process, which was originally introduced for Evolution Strategies. In the second case, one metaheuristic method is used to control the other, e.g., by tuning its parameters (metaoptimization [37]).

Parallelization. In every iteration the EA processes many points from the population P^t to define the population P^{t+1}. Observe that the process of selection, crossover and mutation can be done separately for each point from the population O^t. Therefore there is a long tradition of EA parallelization. Until very recently the parallelization was limited by the hardware, since massively parallel computers were expensive and relatively hard to access. Now we are witnessing a big change thanks to the introduction of General Purpose Graphic Processing Units (GPUs). GPUs are multicore processors with typical number of cores of the order of hundreds. It is possible to program them under a number of higher-level programming languages, and the most popular toolkits that allow for this are CUDA and openCL (see e.g. a comparison of performance in [38]).
A serious limitation of GPUs is that the cores must be run in a Single Instruction Multiple Data regime. If the problem definition allows to work in this mode then the user can count on very interesting speedup values of several tens up to several hundreds when comparing the total execution time of a program working with and without the GPU card. A convenient software platform to start from is called EASEA and has been maintained by the team from the Strasbourg University [39]. The platform uses CUDA and it allows the user to concentrate on programming the problem definition since many standard EA version have been preprogrammed as EASEA templates. The programming language is a dialect of `C++`. An alternative attempt to the use of openCL to implement an EA on GPU can be found e.g. in [40].

4. Concluding Remarks

Analysis of the dynamics of the EA population reveals two interleaving phases: the quasi-stability of the sampling distribution, when the population oscillates in an area of the domain for many iterations, and the population drift when the population changes its location in a more systematic way, which takes only few generations. When looking at the dynamics of the EA development, a similar pattern can be recognized. For this reason it is quite easily predictable that the current trends of development will remain current

for several years, but it seems also quite possible that it will be not much to win by continuing them.

In the author's personal opinion, what really lacks in EA is a tighter link between theory and practice. It seems that the bottleneck is definitely on the side of theory. Theoreticians are used to consider optimization problems where the stress is on the convergence. This kind of analysis may imply that the EA should possess some element of a "pure chance" that will allow to reach any nonempty neighborhood of every point by transformation of any point from the domain. In the light of this criterion the EA is a method which is worse than the random walk or the uniform sampling since for these methods the probability of such an event is highest! Similarly, when looking at the rate of progress obtained for the parabola function, EAs cannot compete against pseudo-Newton methods.

It seems that the focus of the analysis should be changed. One hopeful direction seems to consider the ability of the population to escape from the neighborhood of a local optimum [41]. This ability relates to the maintenance of the diversity level. Only few works on this issue can be found in the literature (see [30] for an overview). The other possible direction is to observe that EA adapts the populations' location such that better points are more likely to appear. This does not necessary imply that the most interesting area for an EA is the neighborhood of the global optimum, since it may too "narrow" in comparison to the populations' diversity. This effect is sometimes called the "survival of the flattest".

It should be also stressed that if an EA is used as a tool to solve e.g., some engineering problem, it uses an objective function which is the problem model. For this reason, even if the global optimization has succeeded, this means that the problem model, rather the problem itself, has been really solved. There is a need for systematic treatment of such situations.

It seems that the hot topic for the next few years will become the use of massively parallel computing offered by programmable GPUs. The ability to evaluate in parallel a huge number of points may shift the interest from sophisticated methods, which use a small number of points but need a large number of iterations (a typical situation for adaptive EAs), towards simpler methods that can effectively use massive parallelism. Perhaps this will change the criteria of benchmarking various EA methods, since the number of generations will influence the execution time rather than the number of the objective function evaluations.

Acknowledgements

The author is grateful to Prof. K. Trojanowski for comments on the first version of the article.

References

[1] *Evolutionary Computation: The Fossil Record*, D. Fogel, Ed. IEEE Press, 1998.

[2] K. de Jong, *Evolutionary Computation*. MIT Press, 2002.

[3] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[4] S. Wierzchoń, *Sztuczne Systemy Immunologiczne. Teoria i Zastosowania*. EXIT, Warszawa 2001 (in Polish).

[5] *Estimation of Distribution Algorithms: a New Tool for Evolutionary Computation*, P. Larranaga and J. Lozano, Eds. Kluwer, 2002.

[6] M. Clerc, *Particle Swarm Optimization*. Wiley, 2006.

[7] S. Luke, "Essentials of metaheuristics", 2009 [Online]. Available: http://cs.gmu.edu/∼sean/book/metaheuristics/

[8] K. Trojanowski, *Metaheurystyki Praktycznie*. 2nd ed. Wydawnictwo WIT, 2008 (in Polish).

[9] J. Arabas, *Wykłady z Algorytmów Ewolucyjnych*. Warszawa: WNT, 2004 (in Polish).

[10] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. 2nd ed. Springer, 2007.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1998 (3rd ed.)

[12] R. Horst and P. M. Pardalos, *Handbook of Global Optimization*. Kluwer, 1995.

[13] P. M. Pardalos and H. E. Romeijn, *Handbook of Global Optimization, Volume 2*. Kluwer, 2002.

[14] A. Torn and A. Zilinskas, *Global Optimization*. Springer, 1989.

[15] K. A. de Jong, "Genetic algorithms are not function optimizers", in *Proc. Worksh. Found. Genetic Algorithms FOGA 1992*, Vail, Colorado, USA, 1992.

[16] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization", *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, 1997.

[17] T. M. English, "Optimization is easy and learning is hard in the typical function", *Proc. Congr. Evol. Comp.*, La Jolla, CA, USA, 200, pp. 924-931.

[18] J. Arabas and P. Cichosz, "Searching for intelligent behavior", in *Proc. 16th Int. Conf. Intel. Inform. Sys. IIS 2008*, Zakopane, Poland, pp. 3–22.

[19] G. Rudolph, "Convergence of evolutionary algorithms in general search spaces", in *Proc. Third IEEE Conf. Evol. Comp. CEC 1996*, Nagoya, Japan, 1996, pp. 50–54.

[20] R. Schaefer, *Foundations of Global Genetic Optimization*. Springer, 2007.

[21] L. Dixon and G. Szegö, *Towards Global Optimization*. North- Holland, 1975.

[22] N. Hansen, A. Auger, R. Ros, S. Finck, P. Posik, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009", in *Proc. GECCO Genetic and Evol. Comput. Conf.*, Portland, Oregon, USA, 2010.

[23] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization", Tech. Rep., Nanyang Technological University, Singapore, 2005.

[24] H.-G. Beyer, *The Theory of Evolution Strategies*. Springer, 2001.

[25] G. Rudolph, "Local convergence rates of simple evolutionary algorithms with Cauchy mutations", *IEEE Trans. Evol. Comput.*, vol. 1, no. 4, pp. 249–258, 1997.

[26] M. Vose, *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, 1999.

[27] X. Qi and F. Palmieri, "Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. Part I: Basic properties of selection and mutation", *IEEE Trans. Neural Netw.*, vol. 5, pp. 102–119, 1994.

[28] I. Karcz-Dulęba, "Dynamics of two-element populations in the space of population states", *IEEE Trans. Evol. Comput.*, vol. 10, pp. 199–209, 2006.

[29] I. Karcz-Dulęba, *Dynamika Adaptacji Ewolucyjnych Metod Fenotypowych*. Wrocław University of Technology Press, 2008 (in Polish).

[30] J. Arabas, "Approximating the genetic diversity of populations in the quasi-equilibrium state", *IEEE Trans. Evol. Comput.*, 2011 (to appear).

[31] A. Obuchowicz and P. Pretki, "Evolutionary algorithms with stable mutations based on a discrete spectral measure", in *Proc. 10th Conf. Artif. Intell. Soft Comput. LNAI*, vol. 6114, part 2, pp. 181–188, Springer, 2010.

[32] R. Storn, "Differential evolution research – trends and open questions", in *Advances in Differential Evolution*, U. K. Chakraborty, Ed. Springer, 2008, pp. 1–31.

[33] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies", *Evol. Comput.*, vol. 9, pp. 159–195, 2001.

[34] N. Hansen, "Compilation of results on the 2005 CEC benchmark function set", Tech. Rep., Institute of Computational Science ETH Zurich, 2006.

[35] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms", *IEEE Trans. Evol. Comput.*, vol. 3, pp. 124–141, 1999.

[36] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms", *Swarm and Evol. Comput.*, vol. 1, 2011 (to appear).

[37] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 16, pp. 122–128, 1986.

[38] K. Karimi, N. G. Dickson, and F. Hamze, "A Performance Comparison of CUDA and OpenCL", arXiv:1005.2581v3, 2011.

[39] O. Maitre, F. Krüger, S. Querry, N. Lachiche, and P. Collet, "EASEA: Specification and execution of evolutionary algorithms on GPGPU", *Soft Comput.*, vol. 1, pp. 1–19, 2011.

[40] T. Puzniakowski and M. Bednarczyk, "Towards an OpenCL implementation of genetic algorithms on GPUs", in *Proc. 19th Int. Conf. Intel. Inf. Systems*, Warsaw, Poland, 2011 (to appear).

[41] R. Galar, *Miękka Selekcja w Losowej Adaptacji Globalnej w R^n*. Wrocław University of Technology Press, 1990 (in Polish).

Jarosław Arabas received the M.Sc., Ph.D., and D.Sc. degrees from the Warsaw University of Technology (WUT), Warsaw, Poland, in 1993, 1996, and 2005, respectively. Since 1993, he has been with the Faculty of Electronics and Computer Engineering, WUT, where he is currently a Professor. Since 1995, he has been consulting on applications of these methods for power plant control, decision making, and risk management in electricity markets. His main areas of research interests include evolutionary computation optimization methods, and learning systems.

E-mail: jarabas@elka.pw.edu.pl
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland