The Nested Genetic Algorithms for Distributed Optimization Problems

Jan Roupec and Pavel Popela

Abstract—**In the first part, we review basic principles of the distributed modeling approach in optimization and present introduction to the formal framework based on the concept of a distributed optimization program. The framework is a general one and may be utilized for various classes of decision problems. The DOPs (distributed optimization programs) are introduced as syntactical entities containing certain optimization elements and based on composition rules. They may describe both basic and advanced mathematical programs (e.g., dynamic, stochastic, multistage, and hierarchical) and also game theory models. In addition, more complicated models can be derived from these building stones and further transformed in the syntactical correct way. Although the introduced descriptions are particularly designed for manipulations of programs' structures, semantics for certain DOPs can also be defined. Hence, the next challenge is to search promising solutions in the feasible sets of optimization elements of DOPs. Therefore, several genetic algorithms (GAs) are chosen to search in separate feasible sets and they may also exchange information about different populations for achieved solutions of DOP elements in various ways. The general inspiration comes from decomposition techniques in scenario-based multistage programs, so the name nested GAs is used in our case. The computational results and implementation description are presented for the specific min-max problems that are chosen as elementary prototype instances.**

Index Terms—**Genetic algorithms, minmax problems, distributed optimization programs, nested decompostion**

I. Introduction

Recent engineering optimization problems require further development of suitable modeling and algorithmic tools; see, e.g., [1]. By experience of our collaborators at the Brno University of Technology, decomposability of models and algorithms is asked, see, e.g., [2] and [3] for continuous casting problems, [4], [5], and [6] for concrete design problems, [7], [8], and [9] for power plant policies, [10] for statistical estimation problems, and [11], [12] for stochastic programming engineering applications.

Complexity of optimization models has its quantitative side, related to the growing size of data, and its qualitative side, related to the program structure. We deal mainly with qualitative complexity. In this case, well-known decomposition models and methods characterized by a splitting of the original program into subprograms are widely employed, see [1]. These subprograms must interact, and the goal is to coordinate their actions in such a way that the solution of the original program is obtained.

We may see that several mathematical programming approaches as in [1] give a direct motivation for detailed studies of relations among optimization programs (see also minmax objective functions, two-stage programs, multistage programs, and optimal control and dynamic programming problems). The additional motivation may be provided by other structural approaches to optimization as optimization of multilevel systems and hierarchical optimization (see, e.g., [13] and [14]).

The challenge is to develop tools supporting easy manipulations with large decomposable programs and the solution procedures based on linked GAs.

We have recognized that these models are based on the graph-related decision stage structure, see [1]. Therefore, we have developed an algebraic framework [15], [16] and proposed an object-oriented shell, see [17], [18] that can help the applicability of particular heuristic algorithms, see, e. g. [19] and [20]. The paper shows how to enrich the algebraic framework for so called DOPs (Distributed Optimization Problems) with the use of the set of linked genetic algorithms that we name nested genetic algorithms.

An important requirement that must be considered with further syntax development is to keep developed syntactical rules open also for multicriteria programs, models of conflicts by game theory, and future enhances. Although such requirement may look too ambitious, at the same time we restrict ourselves to the most general and descriptive level. We realize the first attempt in the development of environment that will allow combination of different complex decision problems.

II. Distributed Optimization Problems

Generalizing the experience mentioned in Section 1, we have found that static programs may be understood as certain optimization elements with similar structure: O = (C, F, G). Symbols C, F, and G define the internal structure of any optimization element. The symbol C defines the decision variable $\mathbf{x}$ and related feasible set *C*. With F the evaluation rule for $\mathbf{x}$ is given (e.g., objective function f($\mathbf{x}$)). The symbol G specifies the goal for decision $\mathbf{x}$ if any exists. We may see that this optimization notation is useful for the

Manuscript received July 16, 2011; revised August 10, 2011. This work was supported in part by the projects of Ministry of education of the Czech Republic OPVK CZ.1.07/2.2.00/07.0406 and MSM 0021630529.

J. Roupec is with the Institute of Automation and Computer Science, Brno University of Technology, Technicka 2, 616 69 Brno, Czech Republic (phone: +420 54114 3346; e-mail: roupec@fme.vutbr.cz).

P. Popela is with The Institute of Mathematics, Brno University of Technology, Brno, Czech Republic. (e-mail: popela@fme.vutbr.cz).

specification of different goals. A simple graph notation is developed to describe optimization elements visually. Optimization elements representing deterministic mathematical programs are denoted by nodes. Further elements together with model transformations are discussed in [15].

In this paper we focus on introduction of specialized heuristic algorithms linked to optimization elements. Up to now, all optimization elements have represented separate programs, so the relations between them must be reviewed.

Having two different optimization elements, we see that they may generate optimal and feasible solutions. The relation between two optimization elements is based on the principal idea that the solution of one element may influence the behavior of another one. For example, the solution of the first program may change the feasible set (e.g., coefficients in the constraints) and/or the objective function value (by the additional cost or profit) of the second program.

From the mathematical point of view, the difference between modification of the objective and feasible set is not so important, because the feasibility check may be incorporated into the objective when infinite values are assigned to the objective for infeasible solutions.

The relation between two elements is represented by arc. The relations can be further classified by the impact of the solution of one element on the other one, see [15]. Described relations may also be symmetric; this situation is denoted with two arcs of the opposite direction or with one edge without orientation.

Using sequence of arcs, we identify how the actions discussed with description of relations between programs are scheduled in time. Therefore, for each optimization element the algorithm can be chosen and they may communicate following the arc related information.

In the common case of more optimization elements O_t, t=1,…,T, we may choose them in the form

$$? \in \arg\min\left\{ f_t\left(\mathbf{x}_t; \mathbf{v}_t, \mathbf{u}_t\right) \middle| \mathbf{x} \in C_t(\mathbf{u}_t) \right\}, \qquad (1)$$

where the influence of other elements is described by import parameters $\mathbf{u}_t$ and $\mathbf{v}_t$, and $? \in \arg\min\{...\}$ means that we search for any optimal solution.

Without assigning values to $\mathbf{u}_t$ and $\mathbf{v}_t$, this description has no reasonable semantics. Using our syntax O = (C, F, G), we introduce transformation elements as those that may connect optimization elements (e.g., $T_t(\mathbf{x}_t; \mathbf{v}_t)$ where T_t is a suitable mapping).

The transformation elements may be combined to design even more complex transformation elements. All considered elements (optimization, transformation, random) are called DOP elements.

To link O_t to other elements, we have to specify values of $\mathbf{u}_t$ and $\mathbf{v}_t$. Therefore, for so called immediate ancestor elements with indices from A(t), we have

$$\mathbf{u}_t = \Phi_t(\mathbf{x}_{A(t)}) = \Phi_t((\mathbf{x}_a)_{a \in A(t)}), \qquad (2)$$

where $\mathbf{x}_{A(t)}$ are variables and parameters of O_t immediate ancestors chosen by a modeler, and Ψ_t is a suitable vector (or scalar) function. For so called immediate successor elements with indices from S(t), we have

$$\mathbf{v}_t = \Psi_t(\mathbf{x}_{S(t)}) = \Psi_t((\mathbf{x}_s)_{s \in S(t)}), \qquad (3)$$

where $\mathbf{x}_{S(t)}$ are variables and parameters of O_t immediate successors chosen by a modeler, and Ψ_t is a suitable vector

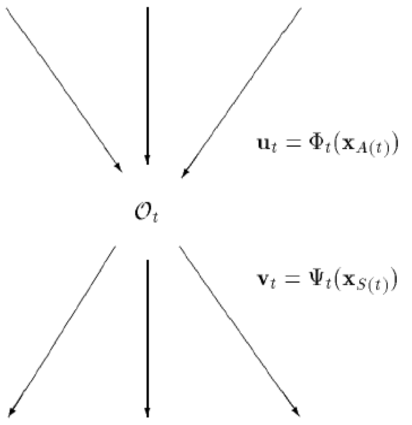
$\mathbf{u}_t = \Phi_t(\mathbf{x}_{A(t)})$
O_t
$\mathbf{v}_t = \Psi_t(\mathbf{x}_{S(t)})$

Fig. 1. DOP element and related arcs, see (4).

(or scalar) function, see Figure 1.

Therefore, the DOP elements may be understood as being composed of optimization, transformation, and further elements indexed by t with a similar structure

$$O_t = (C_t, F_t, G_t, \Phi_t, \Psi_t). \qquad (4)$$

For the given set of indices t, the set of O_t's is called a distributed optimization program (DOP). The DOP is closed (CDOP) when for all t: $\mathbf{u}_t$ and $\mathbf{v}_t$ parameters are all specified with Φ_t and Ψ_t. We may assume that variables, functions, and parameters are specified in the manner that satisfies necessary dimensional requirements. Therefore, the DOP contains nodes that are connected by arcs. Graphical representation of DOP is called a support graph.

In certain cases, $\mathbf{u}_t$ and $\mathbf{v}_t$ parameters are not fully specified for some Φ_t and Ψ_t. The DOP with this property is called an open DOP (ODOP). The ODOP may be understood as representing a certain form of uncertainty. When we specify the uncertainty including random elements and further deterministic reformulations, we again obtain closed DOP (CDOP), see [16].

We have syntactically correct description with graphical visualization based on relations among DOP elements. Under circumstances introduced in [16] i.e. the CDOP without circuits, having one forefather optimization element, may be shrunken into one-node compressed description.

In this case, the forward substitution procedure applied to the DOP replaces all occurrences of $\mathbf{u}_j$ with Φ_j in successive forward steps, and then, the backward procedure replaces $\mathbf{v}_j$ with Ψ_j and collects constraints and objective functions. This backward procedure is similar to the substitution used for composed functions.

We have presented DOP syntax and we now mention

DOP semantics. Usually, the obtained one-node compressed description is rather complex, but it may be often interpreted as the mathematical program that defines the DOP semantics.

When objective functions of all node programs are contained in the final composed objective function of the forefather node, we may understand it as a generalized utility function that replaces multiple criteria with a single criterion.

However, in certain cases, this compact description is not available and simulation techniques and heuristics to study behavior of the DOP are welcome. So, we focus on this problem in the next section.

III. Nested Genetic Algorithms

Many optimization models with a complex structure (e.g., dynamic programming, multistage stochastic programming, bilevel programming, Nash-Stackelberg games, minmax problems, etc.) can be studied within the framework of DOP.

Therefore, it is a challenge to develop algorithms that are enough robust to be suitable for the large set of DOPs. We may find that the aforementioned classes of programs and games often deal with specialized algorithms. In addition, many of them also involve specialized heuristic algorithms.

Hence, we have decided to utilize our previous experience, see [16] and [20] and extend the developed GAs in such a way that may serve for the DOPs as well. However, there are certain methodological challenges that must be solved. Therefore, to tackle them, we firstly focus on the particular class of minmax problems that can be applied in engineering robust models, see also [20].

Firstly, we review the fundamental ideas of genetic algorithms (GAs) that is quite understandable and seems to be widely known. Genetic algorithms belong in stochastic heuristic optimization methods. The main usage of GA is the solving of problems of complicated multi-dimensional optimization where classical analytic and numerical techniques fail.

We denote an optimization problem as

$$\mathbf{x}_{\text{opt}} \in \arg\min\left\{ f(\mathbf{x}) \middle| \mathbf{x} \in C \right\}, \tag{5}$$

where *f* is an objective function, *C* is a set of feasible solutions, $\mathbf{x}$ is a feasible solution and $\mathbf{x}_{\text{opt}}$ is the optimum solution to be found.

The strategy of searching through the set *C* used by the GA is inspired by natural evolution, where the best individuals have the biggest chance to survive and to become parents of new offspring. In addition, the GA uses another mechanism existing in the nature - mutation. Mutation is based on a random change of genetic information. Using mutation individuals can adapt to changing living conditions easier. Mutation can also prevent degeneration (in optimization the deadlock in a local extreme can be supposed to be an analogy to degeneration).

The GA has an iterative character. It works not only with one solution in time but with the whole population of solutions. The population contains many (from tens to hundreds) individuals – bit strings representing solutions.

The mechanism of GA involves only elementary operations like strings copying, partially bit swapping or bit value changing. The GA starts with a population of strings and thereafter generates successive populations using the following three basic operations: reproduction, crossover, and mutation.

Reproduction is the process by which individual strings are copied according to an objective function value (fitness). Copying of strings according to their fitness value means that strings with a higher value have a higher probability of contributing one or more offspring to next generation. This is an artificial version of natural selection.

Mutation is an occasional (with a small probability) random alteration of the string position value. Mutation is needed since, in spite of reproduction and crossover effectively searching and recombining the existing representations, they occasionally become overzealous and lose some potentially useful genetic material. The mutation operator prevents such an irrecoverable loss.

The recombination mechanism allows mixing of parental information while passing it to their descendants, and mutation introduces innovation into the population. In spite of simple principles, the design of GA for successful practical usage is surprisingly complicated. The GA may have many parameters that depend on the problem to be solved.

The description of GA implementation details can be found in [21] and [22]. The basic GA that we use for particular optimization elements in the examples described in the paper utilizes shadows i.e. a type of redundancy introduced by [23], and the death operator, see [24].

As an example of the DOP that we use for our tests we consider the following minmax problem

$$? \in \min_{\mathbf{x}\in X} \max_{\mathbf{y}\in Y} \{ f(\mathbf{x},\mathbf{y}) \}, \tag{6}$$

where f is an objective function, $\mathbf{x}$ and $\mathbf{y}$ are decision vectors from spaces X and Y respectively.

In this case, there are two optimization elements. A maximization-based (outer) element should be solved for fixed values of $\mathbf{x}$ set by Ψ, and then the minimization-based (inner) element has to be solved for obtained optimal values of $\mathbf{y}$ that are passed from maximization-based element by identity transformation Φ.

Hence, the both elements are represented by nodes and they are connected by arcs. Our test computations have shown that the single GA is not suitable for solving the minmax problem. Thus we have developed and tested the population-based method with the nested arrangement of genetic algorithms. The basic idea was inspired by [25] and further generalized.

We use the matrix of three columns and *n* rows. The first and second columns contain the values of $\mathbf{x}$ and $\mathbf{y}$ variables and the third one holds the corresponding objective function values. Above that the auxiliary data structure is used for storing the triplet $\left(\mathbf{x}_i, \mathbf{y}_i, f(\mathbf{x}_i, \mathbf{y}_i)\right)$ representing the best known solution at the moment.

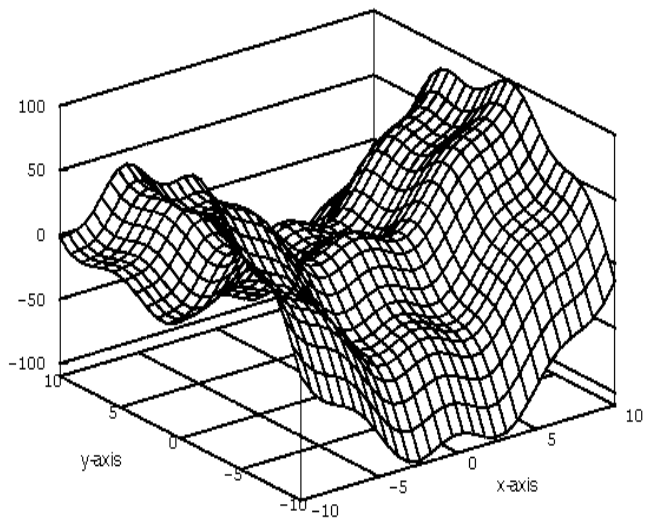

Fig. 2. The graph of the 1st test function (7).

The nested combination of GAs works as follows:

1. Initialization
   1.1. Generating values $\mathbf{x}_i^0, i \in \{1,...n\}$ (randomly). The subscript denotes the line number in the matrix and the superscript denotes the number of iteration.
   1.2. Computing $\mathbf{y}_i^1, i \in \{1,...n\}$, such that

$$f(\mathbf{x}_i^0, \mathbf{y}_i^1) = \max_{\mathbf{y} \in Y}\{f(\mathbf{x}_i^0, \mathbf{y})\}$$

   using GA selected for **y** related DOP element (the inner nested GA).
   1.3. Sorting of matrix rows by the third column (ascending). Storing of the first line values to the auxiliary structure (least known maximum). Hence, the obtained solution of the inner DOP element is transformed to the outer DOP element.
2. Main cycle
   2.1. Computing $\mathbf{x}_i^k, i \in \{1,...n\}$, such that

$$f(\mathbf{x}_i^k, \mathbf{y}_i^k) = \min_{\mathbf{x} \in X}\{f(\mathbf{x}, \mathbf{y}_i^k)\}$$

   using GA selected for **x** related DOP element (the outer nested GA).
   2.2. Computing $\mathbf{y}_i^{k+1}, i \in \{1,...n\}$, such that

$$f(\mathbf{x}_i^k, \mathbf{y}_i^{k+1}) = \max_{\mathbf{y} \in Y}\{f(\mathbf{x}_i^k, \mathbf{y})\}$$

   using GA selected for **y** related DOP element (the outer nested GA).
   2.3. Sorting of matrix rows by the third column (ascending). When better solution was obtained it is stored into the auxiliary structure (new least known maximum).
   2.4. Go to step 2.1 until the number of iterations is satisfied.

We have realized several tests (see Figure 2 and 3 for examples of two graphs of the further discussed test objective functions). For the first test case of the objective function

$$f(x, y) = x^2 - y^2 + 10\sin x - 10\cos y\,, \tag{7}$$

we have obtained

$$f\left(x^*, y^*\right) = \min_{x \in X}\ \max_{y \in Y}\{f(x, y)\} = f(-2.6, -1.4) = 6.08\,.$$

For the next more complicated test objective function

$$f(x, y) = x^2 - y^2 + 15\sin\left(\sqrt{x^2 + y^2}\right), \tag{8}$$

we have computed

$$f\left(x^*, y^*\right) = \min_{x \in X}\ \max_{y \in Y}\{f(x, y)\} = f(4.13, 0.00) = 4.53\,.$$

For presented instances, we have chosen $X = \langle -10, 10 \rangle$ and $Y = \langle -10, 10 \rangle$. We have used the population size of 30 individuals and the number of iterations of 30 for all genetic algorithms in the arrangement described above.

We did not realize any statistical evaluation (some principal ideas to follow in the future can be found in [22]. The results tended to stabilize usually after 10 iteration of the main cycle. Results were verified using software Octave.

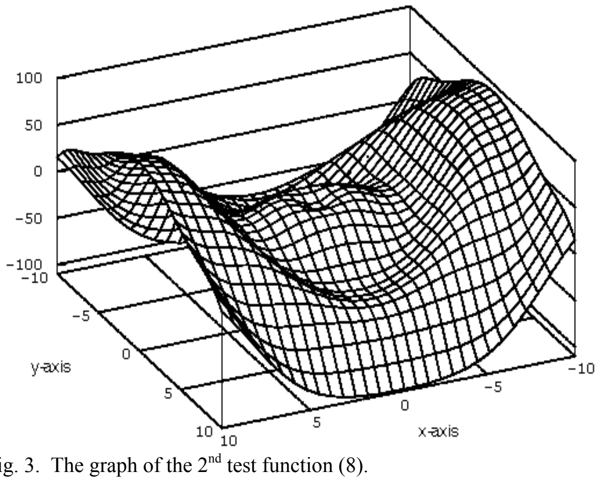

Fig. 3. The graph of the 2nd test function (8).

IV. Conclusion

We have reviewed basic concepts of DOPs, focusing on syntax and mentioning semantics. For the simple case of the minmax problem, the nested genetic algorithms are introduced. Two computational examples are presented and visualized.

ACKNOWLEDGMENT

The paper was supported by project from MSMT of the Czech Republic no.1M06047, by a research plan from Ministry of Education of the Czech Republic no. MSM0021630519 and by the Czech Ministry of Industry and Trade within the framework of research tasks FR-TI1/357. The support from Strategiskhoyskoleprosjekt "Supply chain management og optimierungsmodeller" is also gratefully acknowledged.

REFERENCES

[1] P. Popela, "An Objected-Oriented Approach to Multistage Stochastic Programming," Ph.D. dissertation, Charles University, Prague, 1998.

[2] F. Kavička, J. Štětina, B. Sekanina, et al. "The optimization of a concasting technology by two numerical models," *Journal of Materials Processing Technology,* vol. 185, no.1–3, pp. 152–159, 2007.

[3] T. Mauder, Č. Šandera, M. Šeda, J. Štětina "Optimization of Quality of Continuously Cast Steel Slabs by Using Firefly Algorithm," in *Proc. of Conference on Materials and Technology*, Portoroz, 2010, pp. 45–53.

[4] J. Plšek, P. Štěpánek, P. Popela, "Deterministic and Reliability Based Structural Optimization of Concrete Cross-section," *Journal of Advanced Concrete Technology* vol. 5, no. 1, pp. 63–74, 2007.

[5] P. Štěpánek, J. Plšek, I. Laníková, F. Girgle, and P. Šimůnek, "Optimization of Concrete Structures Design," In *Proc. of the 16th International Conference on Soft Computing MENDEL 2010*, Brno, 2010, pp. 434–440.

[6] I. Laníková, P. Štěpánek, P. Šimůnek, "Fully Probabilistic Design of Concrete Structures," In *Proc. of the 16th International Conference on Soft Computing MENDEL 2010*, Brno, 2010, pp. 426–433.

[7] M. Touš, M. Pavlas, P. Stehlík, P. Popela, "Effective biomass integration into existing combustion plant," *Energy*, vol. 36, no. 8, pp. 4654 – 4662, 2011.

[8] M. Pavlas, M. Touš, P. Klimek, L. Bébar, "Waste incineration with production of clean and reliable energy," (10 p.). *Clean Technologies and Environmental Policy,* vol. 13, no. 4, pp. 595–605, 2011.

[9] M. Pavlas, M. Touš, L. Bébar, P. Stehlík, "Waste to Energy - An Evaluation of the Environmental Impact," *Applied Thermal Engineering*, vol. 30, no. 10, pp. 2326–2332, 2010.

[10] R Matoušek, Z. Karpíšek, "Exotic Metrics for Function Approximations," in *Proc. of the 17th International Conference on Soft Computing MENDEL 2011*, Brno, 2011, pp. 560–566.

[11] T. Mauder, J. Novotný, "Two Mathematical Approaches for Optimal Control of the Continuous Slab Casting Process," in *Proc. of the 16th International Conference on Soft Computing MENDEL 2010*, Brno, 2010, pp. 395–400.

[12] E. Žampachová, P. Popela, M. Mrázek, "Optimum beam design via stochastic programming". *Kybernetika* vol. 46, no. 3, pp. 575–586, 2010.

[13] P.N.D. Bukh, "A bibliography of hierarchical production planning," Research report, Institute of Management, University of Aarhus, March 1994.

[14] R. M. Burton and B. Obel, editors. *Design Models for Hierarchical Organizations: Computation, Information and Decentralization.* Kluwer Academic Press, 1995.

[15] P. Popela, R. Matoušek, J. Sklenář, "The Formal Framework for DOP," in *Proc. of the 14th International Conference MENDEL 2008*, Brno, 2008, pp. 235–240.

[16] P. Popela, R. Matoušek, J. Sklenář, E. Žampachová, J. Roupec, "Advances in the Formal Framework for DOP," in *Proc. of the 17th International Conference on Soft Computing MENDEL 2011*, Brno, 2011, pp. 454–469.

[17] P. Popela, R. Matoušek, J. Sklenář, E. Žampachová, "The internal objects for optimization programs," in *Proc. of the 15th International Conference on Soft Computing MENDEL 2009*, Brno, 2009, pp. 247–258.

[18] P. Popela, R. Matoušek, J. Sklenář, E. Žampachová, "The external objects for optimization programs," in *Proc. of the 16th International Conference on Soft Computing MENDEL 2010*, Brno, 2010, pp. 465–470.

[19] R. Matoušek, "GAHC: Improved GA with HC mutation", in *Proc. of the World Congress on Engineering and Computer Science WCECS 2007*, San Francisco, CA, 2007, pp.915–920.

[20] J. Roupec, P. Popela "Scenario generation and analysis by heuristic algorithms", in Proceedings of World Congress on Engineering and Computer Science (WCECS), San Francisco, CA, 2007, pp. 931–935.

[21] J. Roupec "Advanced Genetic Algorithms for Engineering Design Problems," *Engineering Mechanics*, vol. 17, no. 5/6, pp. 407–417, 2011.

[22] J. Roupec, "Design of Genetic Algorithm for Optimization of Fuzzy Controllers Parameters (In Czech)," Ph.D. dissertation, Brno University of Technology, Brno, 2001.

[23] C. Ryan, "Shades. Polygenic Inheritance Scheme," in *Proc. of the 3th International Conference on Soft Computing MENDEL 1997*, Brno, 1997, pp. 140–147.

[24] P. Ošmera, J. Roupec "Limited Lifetime Genetic Algorithms in Comparison with Sexual Reproduction Based GAs," in *Proc. of the 6th International Conference on Soft Computing MENDEL 2000,* Brno, 2000, pp. 118–126.

[25] H. J. C. Barbosa, "A Coevolutionary Genetic Algorithm for Constrained Optimization," in *Proc. of the CEC'99*, Washington, DC, 1999, pp. 1605–1611.

[26] Č. Šandera, P. Popela, J. Roupec, "The Worst Case Analysis by Heuristic Algorithms," in *Proc. of the 15th International Conference on Soft Computing MENDEL 2009*, Brno, 2009, pp.109–114.

[27] P. Popela, J. Roupec, P. Ošmera, R. Matoušek, "The Formal Stochastic Framework for Comparison of Genetic Algorithms," in *Proc. of The 2002 IEEE World Congress on Computational Intelligence*, Honolulu, HI, 2002, pp. 576–581.

[28] R. Matoušek, P. Ošmera, J. Roupec, "GA with Fuzzy Inference System," in *Proc. of the Congress of Evolutionary Computing CEC 2000*, La Jolla, CA, 2000, pp. 646–651.

[29] J. Šťastný, V. Škorpil, "Genetic Algorithm and Neural Network," in *Proc. of the 7th WSEAS International Conference on Applied Informatics and Communications*, Vouliagmeni, Greece, 2007, pp. 347–351.