2019 International Russian Automation Conference (RusAutoCon)

Solving Design Optimization Problems Based on Computational Intelligence Methods

Leonid A. Gladkov
Southern Federal University
Rostov-on-Don, Russia
leo_gladkov@mail.ru

Nadezhda V. Gladkova
Southern Federal University
Rostov-on-Don, Russia
nadyusha.gladkova77@mail.ru

Sergey N. Leyba
Southern Federal University
Rostov-on-Don, Russia
lejba.sergej@mail.ru

***Abstract*—A new approach to solving computer-aided design problems based on the integration of various approaches to the organization of the search and optimization process is presented. The statement of the problem is given, constraints of acceptable solutions and formulated quality criteria are chosen. A scheme for organizing the work of a parallel optimization algorithm is proposed. New modifications of evolutionary operators for use in the process of the parallel hybrid algorithm are developed. The implementation of the fuzzy control module based on the use of a multilayer neural network and Gaussian functions is proposed. The main characteristics of the neural network used for training and tuning the parameters of the hybrid algorithm are described. The basic principles of the fuzzy control unit are formulated. The main results of testing the proposed hybrid algorithm and their evaluation are given.**

Keywords—computer-aided design, optimization tasks, design tasks, evolutionary computating, neural network training, hybrid intelligent systems, parallel genetic algorithm

I. Introduction

The tasks of design engineering, as a rule, are characterized by great computational complexity. This is due to the need to search through a huge number of different solutions. Moreover, the exact solution can be obtained only through complete exhaustive search, which is impossible. The most important tasks of design engineering include the tasks of partitioning, placement, routing, etc.

To ensure finding the best solution (global extremum) in such problems, it is necessary to perform a complete search, which is not possible for their large dimension [1–3]. Therefore, to solve such problems, meta-heuristic algorithms are developed, which make it possible to find solutions that are close to optimal.

The reported study was funded by RFBR, project number 17-01-00627.

At present, hybrid approaches are often used to solve complex multi-criteria optimization problems. The idea of hybridization is to try to create systems that combine different approaches. For example, neural networks and evolutionary algorithms, ideas of fuzzy logic and principles of control theory, etc. Such integration is a characteristic feature of many areas of computing intelligence [4–6].

A hybrid algorithm based on the integration of evolutionary search methods and fuzzy control was proposed for solving optimization design problems. Also, the mechanisms of parallelization of calculations is proposed to use to improve the efficiency of the organization of the search process [7–16]. Distributed genetic algorithms are a subclass of parallel genetic algorithms (ParGA). They are used to contend the premature convergence of the process of finding and stimulating the diversity of a population of alternative solutions. Distributed genetic algorithms are based on dividing a population into several separate subpopulations, in each of which calculations are made independently of other half populations.

II. Formulation of the Problem

The tasks of placement and tracing occupy a special place among the optimization problems of computer-aided design.. Traditionally, they are solved at different stages by various methods, which leads to an increase in the cost of time and computational resources. Therefore, it seems expedient to develop integrated methods for solving location and tracing problems that allow these tasks to be performed in one cycle with mutual consideration of existing constraints and current results [17].

The number of circuit elements

$$E = \{e_i \mid i = 1, \ldots, N\},$$

that must be placed in a given set of positions P is the initial data of the task. Each element

$$e_i = (l_i, h_i, T_i),$$

is characterized by a specific set of parameters (overall dimensions), such as length l_i, height h_i and contact list T_i

$$T_i = \{t_j \mid j = 1, \ldots, K\}$$

$$t_j = (x_j, y_j)$$

where t_j is the contact, x_j, y_j are the contact coordinates relative to the element base point; K is the number of contacts of the element.

The optimal placement of elements on the installation space at which the total area of overlapping of the placed elements was zero, and the sum of the remaining minimum criteria values must be found.

Also for each chain, it is necessary to determine the list of positions on the installation space through which this chain passes:

$$W_h = \{(x_q, y_q) \mid i = 1,\ldots,Q\}$$

where Q – the number of positions through which the h circuit passes.

III. Description of the Algorithm

At the beginning of the hybrid algorithm, a population of alternative solutions is created, each of which can be represented as a H_i chromosome (numerical sequence). Each digit of this vector corresponds to the position number, and its value corresponds to the number of the element to be placed. The length of the chromosome (vector) is determined by the number of placed elements. The set of positions is represented in the form of a regular structure (lattice). Each position p_i has coordinates x_i, y_i. The positions are numbered in ascending order of the x_i coordinate within the string from left to right, and the rows are in turn ordered in ascending order of the y_i coordinate from top to bottom.

Each element has a base point relative to which the contour description of the element e_i is given. In our case, the lower left corner of the element is taken as the base point. Outline description has a rectangular shape.

The final value of the objective function is calculated by the formula:

$$F_p = k_1 * S + k_2 * L + k_3 * T + k_4 * J + k_5 * Q$$

This optimization problem is multicriteria, therefore when calculating the value of the objective function, the following are taken into account:

a normalized estimate of the amount of the penalty for overlapping the areas of the placed elements S;

estimating the lengths of possible relationships between elements L;

percentage of unconnected connections (routing index) T;

evaluation of the thermal Q and electromagnetic J compatibility of the elements;

k_1, k_2, k_3, k_4, k_5 are the coefficients, determining the influence of each component of the fitness function on the overall assessment.

To estimate the length of the connections, it is advisable to use the semi -perimeter of the describing rectangle of the circuit.

To assess the quality of tracing, an indicator is used that determines the percentage of unprovoked connections. As an additional criterion, the total area of intersection of the areas describing rectangles of all chains can also be used.

To solve the tasks, a parallel multi-population genetic algorithm has been developed [11–13]. The main idea of this algorithm is to organize the parallel execution of the evolution process on several populations at once.

The possibilities of evolutionary algorithms are expanded by using the principles of distributed computing. Principles of the organization of distributed computing can be successfully used in the process of evolution. In genetic algorithms, the motive force of evolution is random crossbreeding, that is, the intention of each individual to cross with other individuals in the same population. It is obvious that it is rather difficult to manage the evolution when randomly crossed. In parallel genetic algorithms, evolution takes place within separate subpopulations, which are characterized by a certain structural proximity. Genetic operators are also performed mainly within each subpopulation. This property significantly expands the possibilities of managing the process of evolution.

There are three main groups of parallel genetic algorithms: global single-population algorithms (Master-Slave GAs); single-population algorithms (Fine-Grained GAs); multi-population algorithms (Coarse-Grained GAs). The main advantage of the algorithms of the first type is the simplicity of implementation, because parallelization occurs only at the level of calculation of the fitness-function (FF). The consequence of this organization is the low efficiency of the computational process.

In Fine-Grained GAs, one spatial-structured population is used. Selection and crossing operators are limited to close relationships. The migration operator appears in these algorithms. In the process of evolution, regions of populations are formed with a similar value of FF. As the algorithm works, these areas grow and compete with each other. The disadvantage of this strategy is the need for constant synchronization of subpopulations due to asynchronous operation and the presence of a common address space.

Of greatest interest for solving the problems under consideration are Coarse-Grained GAs. They use several large subpopulations and exchange individuals between them through migration. The transition from a genetic algorithm with the sequential organization of calculations to a multi-population parallel GA does not require significant efforts. To do this, it is only necessary to add a mechanism for implementing the exchange between subpopulations.

In the development process, an analysis and assessment of the capabilities of existing models of the organization of parallel computing was carried out. The main types of models for the organization of parallel genetic algorithms include the following: insular, buffer, cellular.

The “island” model is based on the division of the main population into a certain number of subpopulations (“islands”), each of which is in development. In the process of evolution there is an exchange of individuals between the "islands" periodically. When the “island” model are using it is very important to set the correct migration frequency. Too frequent exchanges will lead to a reduction in the quality of solutions and the degeneration of the parallel algorithm into a regular sequential one. Insufficient migration frequency can lead to premature convergence of the computational process.

In the cell model, each individual can interact only with his neighbors (above, below, left, right). Each cell corresponds to exactly one individual. All individuals have a random value of the function of fitness at the beginning of the process. Regions consisting of individuals with similar values of the fitness function are formed after performing several generations of the algorithm. In the process of the algorithm, the size of these areas increases and competition arises between different areas.

The buffer model can be represented as a “star” (Fig. 1). In this case, the interaction between different subpopulations is carried out through a buffer. In the process, the buffer is filled with individuals from different populations. When a certain situation arises, the subpopulation goes into the buffer and takes part or all of the chromosomes from there, and then adds some of its individuals to it.

The evolution of the subpopulations is performed separately. At each iteration, the condition for applying the migration is checked. If the condition is met, an exchange occurs between the subpopulation and the buffer. Then the buffer size is checked. If the current size exceeds the allowable, then the reduction is performed. Migration conditions and types of genetic operators may be different for different subpopulations. The advantage of this model lies in its flexibility [12, 13].

The island and buffer models are used in the genetic algorithm to organize a parallel computational process. The choice of the type of model is carried out in the process of the algorithm. The buffer model is based on the exchange of chromosomes between populations through an intermediate buffer. In the process of evolution, the intermediate buffer is filled with solutions. And, upon the occurrence of a specific event, the population can exchange chromosomes with a buffer (Fig 2). The island model is based on the direct exchange of chromosomes between populations. At a certain point in time, synchronization of the development of parallel populations is performed. Solutions are exchanged between populations using a modified migration operator. In the developed algorithm, each population can exchange with two neighboring populations (Fig 3). In this case, for the exchange in each population, a given number of solutions are selected that have the best value of the fitness function (FF).

An important point in the development of a parallel algorithm is the selection of the frequency of migration. An increase in the frequency of migrations leads to a degeneration of populations, and its decrease, on the contrary, to a decrease in convergence. To regulate the frequency of migration is proposed to use a fuzzy logic controller (FLC). At the input of the controller receives data on the effectiveness of the evolutionary process. Based on the results of the evaluation of these data, a decision may be made to change the main parameters of the genetic algorithm. The hybrid migration operator allows migration only when it is necessary, for example, when there is a threat of premature convergence.

For each placement option, an estimated trace is performed using the wave algorithm. Then, from one population to another, a certain number of chromosomes with the best traceability index are copied. The same number of chromosomes with the worst value of the objective function is removed from the populations. Figure 4 shows a diagram of the model of a parallel genetic algorithm performed on 2 populations. In practice, the number of populations can be much larger.

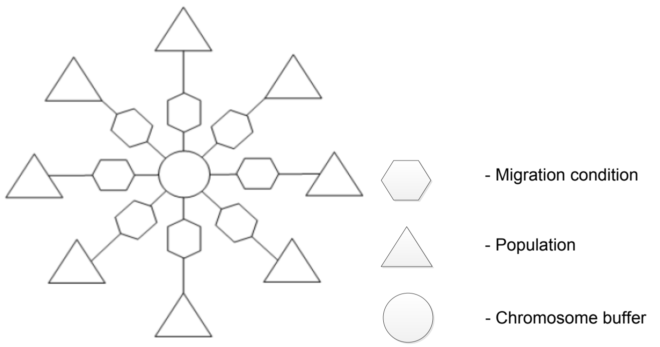

Fig. 1. Buffer Model Structure

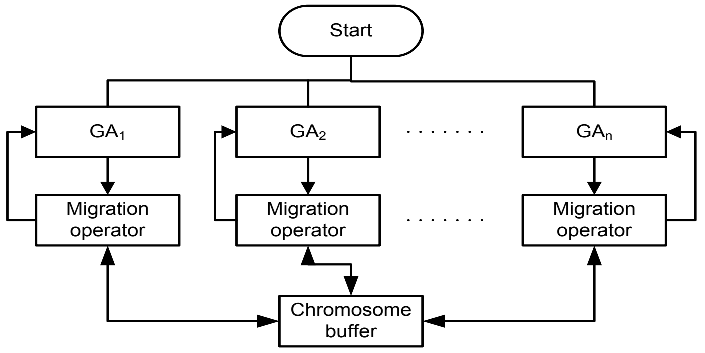

Fig. 2. Buffer Model of ParGA

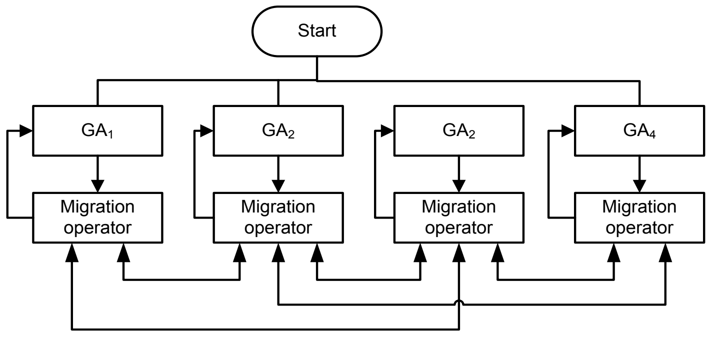

Fig. 3. Island Model of ParGA

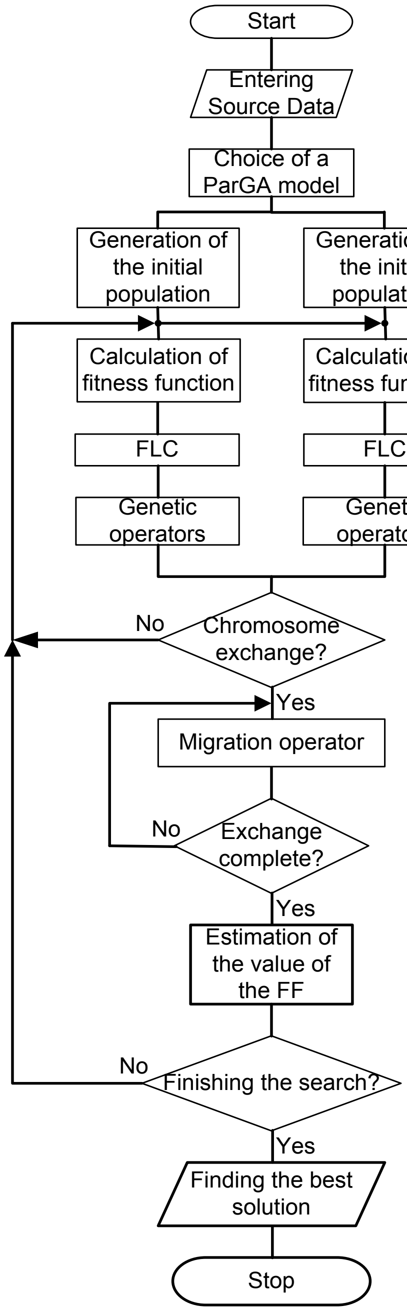
Start
Entering Source Data
Choice of a ParGA model
Generation of the initial population
Generation of the initial population
Calculation of fitness function
Calculation of fitness function
FLC
FLC
Genetic operators
Genetic operators
Chromosome exchange?
No
Yes
Migration operator
Exchange complete?
No
Yes
Estimation of the value of the FF
Finishing the search?
No
Yes
Finding the best solution
Stop

Fig. 4. The scheme of the parallel hybrid algorithm

Inclusion in the proposed search scheme for a fuzzy logic controller allows to improve the quality of the solutions obtained by adjusting the parameters of the genetic algorithm. In this case, a fuzzy logic controller provides the possibility of a reverse control action on the parameters of the genetic algorithm in order to correct them promptly [15–18].

The following values are used as estimates for the operation of a fuzzy logic controller: the best (f_{best}), worst (f_{worst}) and average (f_{ave}) values of the FF at the current iteration (t), as well as the comparison of these values with similar values at previous iterations (t - 1) [10, 11, 18–20]:

$$e_1(t)=\frac{f_{ave}(t)-f_{best}(t)}{f_{ave}(t)};\ e_1(t)=\frac{f_{ave}(t)-f_{best}(t)}{f_{worst}(t)-f_{best}(t)};$$

$$e_3(t)=\frac{f_{best}(t)-f_{best}(t-1)}{f_{best}(t)};\ e_4(t)=\frac{f_{ave}(t)-f_{ave}(t-1)}{f_{ave}(t)}.$$

The variables e_1, e_2, e_3, e_4 are defined on the following intervals:

$$e_1\in[0;\ 1];e_2\in[0;\ 1];e_3\in[-1;\ 1];e_4\in[-1;\ 1].$$

The output parameters are the probabilities of crossover, mutation and migration, respectively - $Pc(t)$, $\Delta Pm(t)$, $\Delta Ps(t)$:

$$Pc(t)\in[0;\ 1];Pm(t)\in[0;\ 1];Ps(t)\in[0;\ 1].$$

To improve efficiency, a learning module based on a multilayered neural network is introduced into the structure of a fuzzy controller [10, 11, 18–20]. The output function can be written based on the Gauss function. After combining all the elements, the resulting output function takes the following form:

$$\bar{y}=\frac{\sum_{k=1}^{N}\bar{y}^{k}\left(\prod_{i=1}^{n}\exp\left(-\left(\frac{\bar{x}_i-\bar{x}_i^{k}}{\sigma_i^{k}}\right)^2\right)\right)}{\sum_{k=1}^{N}\left(\prod_{i=1}^{n}\exp\left(-\left(\frac{\bar{x}_i-\bar{x}_i^{k}}{\sigma_i^{k}}\right)^2\right)\right)}.$$

In the developed hybrid algorithm, the neural network contains 4 layers. Each element of the first layer implements the fuzzy set membership function A_i^k, 1, …, n; k = 1, …, N. In this layer the input signals $\overline{\text{xi}}$ come, and at its output the values of the membership function for these signals are formed. At the output of the first layer, the values of the membership function of fuzzy sets are formed. The configuration of the links of layer 2 corresponds to the rule base, and the multipliers to the output block. The use of multipliers as nodes of layer 2 is due to the fact that in fuzzy operations the multiplication operation is used. The number of elements in this layer is equal to the number of rules stored in the database. Layers 3 and 4 realize the functions of the defuzzification block. As a result, we get a multilayered neural network with a fuzzy inference function. The peculiarity of this network is that, each layer, all parameters and weights of network elements have an obvious physical interpretation.

IV. Results Of Computer Experiments

To assess the performance of a fuzzy logic controller, graphs of dependencies are displayed, reflecting the dynamics of changes in the input and output parameters of the controller. (Fig. 5).

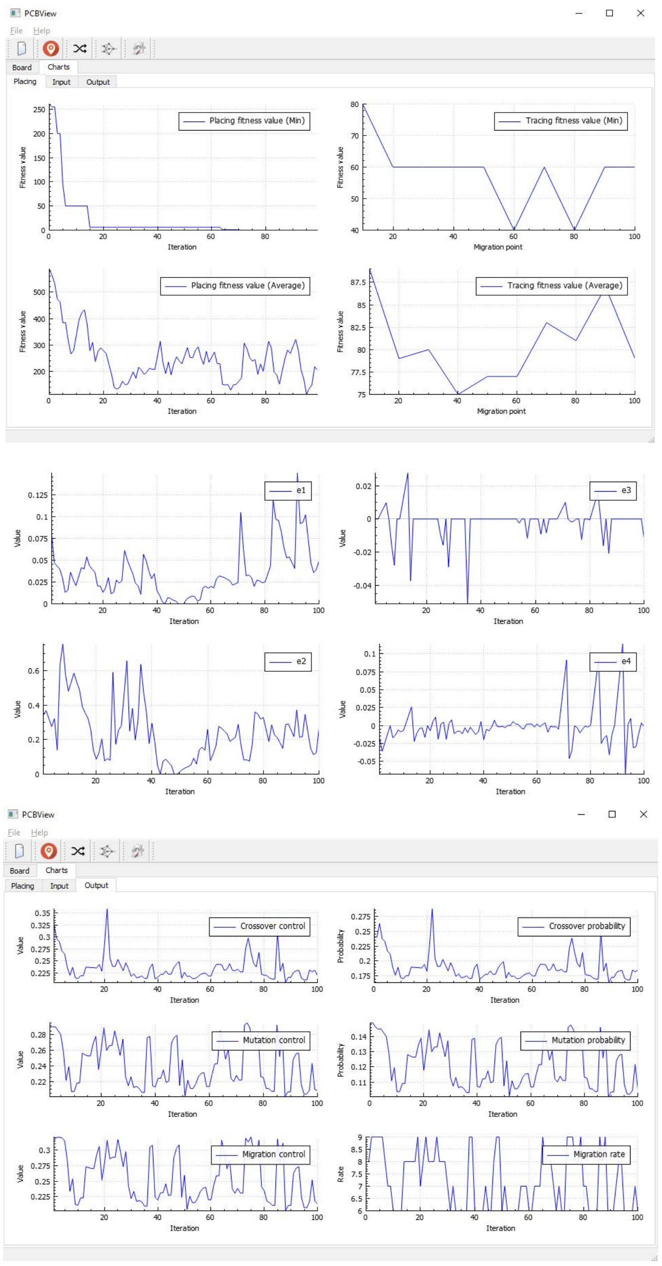

Fig. 5. Graphs of changes in input and output parameters of a fuzzy logic controller

Analysis of the experimental data allows us to conclude that the use of the fuzzy logic controller does not increase the execution time of the hybrid algorithm. At the same time, using the fuzzy logic controller allows you to improve the results of solving the problem by an average of 25% with the same number of iterations.

Network factors can be looked for by random or directed search. As a directional search tool, it is possible to use a genetic algorithm. As a result of using the training block, the values corresponding to the optimal parameters of the algorithm were determined.

REFERENCES

[1] J. P. Cohoon, J. Karro, and J. Lienig, “Evolutionary Algorithms for the Physical Design of VLSI Circuits,” in Advances in Evolutionary Computing: Theory and Applications, A. Ghosh and S. Tsutsui, Eds. London: Springer Verlag, 2003, pp. 683–712.

[2] Charles J. Alpert, Dinesh P. Mehta, and Sachin S. Sapatnekar, Handbook of algorithms for physical design automation. New York: CRC Press, 2009.

[3] N. Shervani, Algorithms for VLSI physical design automation. USA: Kluwer Academy Publisher, 1995.

[4] L. A. Gladkov, V. M. Kureychik, V. V. Kureychik, and P. V. Sorokoletov, Bioinspired methods in optimization. Moscow: Fizmatlit, 2009.

[5] L. A. Gladkov, V. V. Kureychik, and V. M. Kureychik, Genetic algorithms. Moscow: Fizmatlit, 2010.

[6] A. Michael and H. Takagi, “Dynamic control of genetic algorithms using fuzzy logic techniques,” Proc. of the 5th Int. Conf. on Genetic Algorithms, pp. 76–83, 1993.

[7] M. A. Lee and H. Takagi, “Integrating design stages of fuzzy systems using genetic algorithms,” Proc. of the 2nd IEEE Int. Conf. on Fuzzy System, pp. 612–617, 1993.

[8] F. Herrera and M. Lozano, “Fuzzy Adaptive Genetic Algorithms: design, taxonomy, and future directions,” Soft Computing, 7, pp. 545–562, 2003.

[9] R. T. F. A. King, B. Radha, and H. C. S. Rughooputh, “A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration,” Proc. of 2004 IEEE Int. Conf. on Networking, Sensing and Control, pp. 577–582, 2004.

[10] N. G. Yarushkina, Fundamentals of the theory of fuzzy and hybrid systems. Moscow: Finance and Statistics, 2004.

[11] I. Z. Batyrshin and A. O. Nedosekin, Fuzzy hybrid systems. Theory and practice. Moscow: Fizmatlit, 2007.

[12] E. Cantu-Paz, A Survey of Parallel Genetic Algorithms. IlliGAL Report, 1997.

[13] D. S. Knysh and V. M. Kureychik, “Parallel genetic algorithms: Problems, overview and status,” News of RAS. Theory and Management Systems, no. 4, pp. 72–82, 2010.

[14] M. A. Rodriguez, D. M. Escalante, and A. Peregrin, “Efficient distributed genetic algorithm for rule extraction,” Applied Soft Computing, vol. 11, pp. 733–743, 2011.

[15] E. Alba and M. Tomassini, “Parallelism and evolutionary algorithms,” IEEE T. Evolut. Comput., vol. 6, pp. 443–461, 2002.

[16] X. Zhongyang, Y. Zhang, L. Zhang, and S. Niu, “A parallel classification algorithm based on hybrid genetic algorithm,” Proc. of the 6th World Congress on Intelligent Control and Automation, pp. 3237–3240, 2006.

[17] A. Pegat, Fuzzy modeling and control. Moscow: BINOM. Knowledge Lab, 2009.

[18] L. A. Gladkov, N. V. Gladkova, and S. N. Leiba, “Manufacturing Scheduling Problem Based on Fuzzy Genetic Algorithm,” Proc. of IEEE East-West Design and Test Symposium, pp. 209–213, 2014.

[19] L. A. Gladkov, N. V. Gladkova, and A. A. Legebokov, “Organization of Knowledge Management Based on Hybrid Intelligent Methods,” Software Engineering in Intelligent Systems. Proc. of the 4th Computer Science On-line Conf., vol. 3, pp. 107–113, 2015.

[20] L. A. Gladkov, N. V. Gladkova, S. N. Leiba, and N. E. Strakhov, “Development and research of the hybrid approach to the solution of optimization design problems,” Intelligent Information Technologies for Industry, pp. 246–257, 2018.