Hybrid Method for Solving the Placement Problem

Muhanad J. Yaser [1], Leonid A. Gladkov[1] and Nadezhda V. Gladkova[1]

[1] *Southern Federal University, 44 Nekrasovsky lane, Taganrog, 347928, Russia*

Abstract

The article discusses the problem of placing elements of digital computing technology. The analysis of the current state of research on this topic is carried out, the relevance of the problem under consideration is noted. A complex formulation of the problem of placing elements of digital computing technology is presented. Perspective approaches to solving design problems are analyzed, hybrid methods and models for solving complex multicriteria optimization and design problems are described. The principles and scheme of operation of a fuzzy logic controller are described. The description of the used model of the control unit is given. The scheme of interaction of blocks of a fuzzy genetic algorithm is described. A model of a hybrid algorithm for solving the placement problem is proposed. The control parameters of the fuzzy logic controller are determined. The proposed hybrid algorithm is implemented as an application program. To determine the effectiveness of the developed algorithm and select the optimal values of the control parameters, a series of computational experiments was carried out.

Keywords

computer-aided design, placement problem, optimization tasks, bioinspired algorithms, hybrid methods, genetic algorithms, fuzzy control

1. Introduction

The cycle of designing elements of digital computing equipment includes the following stages: system specification, functional design, logical design, circuit design, physical design, manufacturing, assembly, testing and control [1-3]. In turn, the design stage of the design includes tasks: partitioning, placement, routing, etc.

Placement is one of the most important tasks of the physical design stage in the process of solving which a layout of the designed circuit is built on the basis of a given list of connections, as well as an estimate of the signal transit time and possible interconnection delays. The importance of this task is determined by the fact that at this stage a spatial model of the arrangement of elements is being built, which is the basis for the implementation of all subsequent design tasks. In modern electronic devices, interconnection delays become a determining factor, and since the mutual arrangement of circuit elements is determined at the placement stage, this has a significant impact on the design quality.

In the process of solving the placement problem, the existing network list at the block / gate / transistor level is converted into the actual circuit in a finite time. The main building blocks are formed on the basis of a logical list of nets, and after determining the exact location of the circuit elements in each region of the crystal, a general assessment of the temporal characteristics of the designed object is performed. In modern VLSI circuits, the complexity and dimension of the designed circuits are constantly increasing, the required clock frequency continues to grow due to higher performance and more complex functional requirements for a single chip. Moreover, with technology scaling actively in the submicron era, interconnect latency is becoming a dominant factor in overall

Russian Advances in Fuzzy Systems and Soft Computing: Selected Contributions to the 10th International Conference «Integrated Models and Soft Computing in Artificial Intelligence» (IMSC-2021), May 17–20, 2021, Kolomna, Russian Federation
EMAIL: yasir_82@mail.ru (A. 1); leo_gladkov@mail.ru (A. 2); nadyusha.gladkova77@mail.ru (A. 3)
ORCID:

chip performance. Since the layout of the circuit elements and the corresponding interconnection delays are determined during the placement phase, this has a significant impact on the final characteristics of the project. Moreover, if the placement problem is poorly solved, it is almost impossible to perform synchronization, no matter how successfully other physical methods of routing synthesis and optimization are applied. Consequently, placement is considered one of the most important and effective optimization methods in the physical synthesis process. Most of the existing optimization algorithms are aimed at reducing the design time and obtaining a legal solution to the placement problem.

After the main blocks are fixed in place and the boundaries of the regions are determined, the problem of the global placement of the remaining cells is solved, and then the detailed placement is performed to improve the obtained solution In the process of placing elements on the working area, the longest connections are assigned, which can increase delays in problem areas. These delays are then attempted to be reduced by using buffering and connection sizing techniques. At the same time, the solution of the problem of placing elements and synthesizing a working version of laying connections, taking into account the existing time constraints in modern design systems, occurs in a single process called physical synthesis. Physical synthesis includes almost all traditional physical design processes: chip planning, placement, global and detailed routing, while adding the ability to account for design time requirements. Of course, the poor quality of solving planning and placement problems automatically affects the quality of physical synthesis, therefore, designers perform this process in the form of separate iterations in order to timely identify and correct possible problems that arise when solving problems at individual stages of design.

Successful completion of the physical synthesis process still requires timely correction of errors and consideration of problems with noise, variability, and manufacturability. Unfortunately, in order to make such fixes, the developer sometimes has to go back to earlier stages of the process.

2. Formulation of the problem

The standard goal of solving the placement problem is to specify such an arrangement of circuit elements on the working field that leads to the minimization of the total length of circuit connections. This is because the length of the connections can be easily modeled and serves as a good first-order approximation to real-world target functions such as timing, power, and project manageability. There are also various forms of estimating the length of joints. The most popular scoring models are square link length, linear link length, or some approximation of link length, which are used in many placement algorithms. Recently, the Steiner problem model has been actively used to estimate the length of connections, which is considered the most accurate estimate of the laid length of wires, and has also been used as an objective function of placement in some scientific studies. The task of placing elements and determining the minimum length of joints is critical for the design of modern microelectronic products, since the length of the joints directly affects the delays of electrical signals. The length of the connections also affects the quality of the routing solution. The length of the joints also has a significant impact on the technology used in the design, which is another important aspect of physical synthesis.

The main task of placement is to determine the location of the circuit elements in the on-chip. Therefore, first the placement area must be defined, as a rule, it is a rectangular area, the boundaries of which are set by the coordinates of the corner points (x_{low}, y_{low}, x_{high}, y_{high}). However, this condition is optional. Recently, there has been a variety of shapes of placement areas, for example, the use of L-shaped or T-shaped placement areas. However, using a rectangular nesting area is still the norm for the global placement task. The initial data for starting the solution of the placement problem is the list of circuit connections, which can be specified as a graph G = (X, U) [4], where the set X is the elements of the designed circuit, and the set U is the set of connections of the circuit elements [5-7]. In turn, the set of vertices X consists of two disjoint subsets: $X = X1 \cup X2$, where X1 is the set of elements that have not yet been assigned positions, and X2 is the set of elements whose positions have already been determined. The location of each element xi of the circuit must be within the boundaries of the specified location area (working area).

The task of placing different-sized elements in space can be set as follows:

1. Limits are set on the possible area on which elements can be placed. The placement area is usually specified as a rectangle. In this case, we can talk about the problem of placing identical elements on a field with multiple dimensions or about solving a complex problem of placing elements with different installation areas.

2. The dimensions of the elements that need to be placed on the working area are set, for which it is enough to define, for example, two dimensions: the length and width of each type of element.

So, the initial data of the problem are: a, b - dimensions (length and width) of the working field; $\{e_1, e_2, ..., e_n\}$ - set of circuit elements; U - a list of connectivity, reflecting the relationship of elements.

It is necessary to find such a variant of the placement of elements on the working area so that there is no overlapping of elements and the total length of the connections is minimal.

$$E = \{(x_1, y_1), ..., (x_i, y_i), ..., (x_n, y_n)\},$$

where (x_i, y_i) – coordinates of the center of gravity of the footprints intended for placing the elements.

From the point of view of optimization theory, the placement problem can be interpreted as an optimization problem for an additive objective function, which includes the normalized value of the penalty for overlapping the areas of the placed elements and an estimate of the total length of joints:

$$F = \min_{z_j \in Z}(k \cdot O(L(z_j)) + T(S_{sum}(z_j))),$$

where z_j – current accommodation option; k – weight coefficient; S_{sum} – total overlapping area of elements; $O(L(z_j))$ – connection length estimation; $T(S_{sum}(z_j))$ – amount of penalty for overlapping areas.

The total length of the joints is calculated by the formula

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}c_{ij},$$

where d_{ij} - distance between mounting positions on the working area $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$;

c_{ij} - number of links between elements i and j.

To normalize the total length of the connections, calculate the ratio $L(z_j)$ to L_{max}, where

$$L_{max} = n^2 \cdot \sqrt{a^2 + b^2};$$

$$O(L(z_j)) = L(z_j) / L_{max}.$$

The total overlap area is calculated using the following formula:

$$S_P = \sum_{i=1}^{n}\sum_{j=1}^{n} S_{ij},$$

where S_{ij} – overlapping area of elements z_i and z_j.

$$S_{ij} = [0{,}5(a_2 + a_1) - |x_2 - x_1|]\,[0{,}5(b_2 + b_1) - |y_2 - y_1|].$$

Penalty for overlapping areas:

$$P(S_{общ}) = S_{общ} / nab.$$

3. Algorithm structure

From a conceptual point of view, the created intelligent computer-aided design systems can be classified as mixed artificial goal-oriented systems, i.e. systems created by man and combining artificial and natural subsystems, the basis of the functioning of which is the factors of expediency [5].

Hybrid systems are heterogeneous systems consisting of dissimilar components combined to achieve their goals [6, 7]. Integration and hybridization of methods and technologies of different physical nature allows solving problems that cannot be solved on the basis of traditional approaches. Hybrid architectures that combine several paradigms help to overcome the disadvantages inherent in individual methods, while there is a so-called synergistic effect when the advantages of one method compensate for the disadvantages of another [8].

Unfortunately, unlike natural ones, "artificial" systems, as a rule, do not have the ability to develop, self-organize, and adapt to changing external conditions. Therefore, the main task of developers is the need to ensure the presence of such properties in the designed systems.

One of the promising tools for constructing effective design and optimization algorithms is the hybridization of a population algorithm with two or more population and / or nonpopulation algorithms [9 - 12]. The structure of the hybrid algorithm allows preserving the advantages of population algorithms, using them at the initial stage to effectively narrow the search space, and then apply one of the "classical" optimization methods to find the global extremum.

One of the hybrid models is a fuzzy genetic algorithm [13, 14]. It combines the search capabilities of genetic algorithms and the capabilities of fuzzy inference systems for evaluating and changing control parameters. For this, a special block is used - a fuzzy logic controller (FLC), which uses the available fuzzy rules, evaluates the course of the evolution process and the diversity of the current population of solutions and, if necessary, changes the values of free variables, such as the probabilities of executing genetic operators.

The FLC circuit (Fig. 1) is based on feedback control. FLC converts the current information about the course of evolution and the state of the population of solutions to a fuzzy form, then, based on the specified fuzzy rules, it evaluates it, determines the control action and returns the adjusted values of the control parameters.

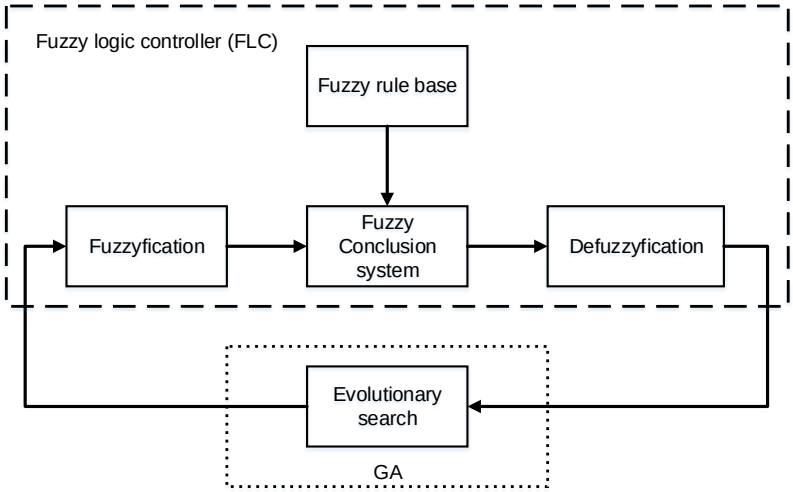

Figure 1: Fuzzy logic controller structure

The basic control law can be written as follows [15 – 17]:

$$u(t) = f(e(t), e(t-1), \ldots, e(t-r), u(t-1), \ldots, u(t-r)),$$

where t is discrete time; e is the error between the model value of the quantity y * and the real value of the output parameter of the control object; f - non-linear function, which is defined as the ratio between the input and output of the FLC.

The functioning of the FLC is determined by a set of linguistically represented rules based on expert knowledge, which are written in the following form: If IF (set of conditions), then THEN (conclusion). FLC operates with fuzzy sets. Therefore, the input values in the fuzzification process are converted into linguistic variables. They are transferred to the block for generating a solution, where one or several fuzzy sets with the corresponding membership functions are formed. After that, the defuzz function is performed, i.e. transforming these fuzzy sets into a control action.

The knowledge that forms the basis for the correct functioning of the fuzzy control module is written in the form of a fuzzy rule:

$$R^k: \textbf{IF}(x_1 \text{ это } A_1^k \textbf{ AND} \ldots \textbf{AND } x_n \text{ это } A_n^k) \textbf{THEN}(y \text{ это } B^k).$$

You can also represent these rules in the form of fuzzy sets with a membership function given by:

$$\mu_{R^{(k)}}(x, y) = \mu_{A^k \to B^k}(x, y).$$

Therefore, if the operation of multiplication is used as a fuzzy implication, then we obtain the formula:

$$\mu_{A^k \to B^k}(x, y) = \mu_{A^k}(x)\mu_{B^k}(y).$$

The Cartesian product of fuzzy sets can be represented as:

$$\mu_{A^k}(x) = \mu_{A_1^k * \ldots * A_n^k}(x) = \mu_{A_1^k}(x) \ldots \mu_{A_n^k}(x_n).$$

As a result of the transformations, we obtain the following expression for the membership function of a fuzzy set $\overline{B}^k$:

$$\mu_{\overline{B}^k}(y) = \sup_{x_1,\dots,x_n \in X} \{\mu_{B^k}(y) \prod \prod_{i=1}^{n} \mu_{A_i'}(x_i)\mu_{A_i^k}(x_i)\}.$$

As a fuzzification operation, we use a singleton-type operation, let

$$A'(x) = \begin{cases} 1, if\ x = \overline{x}; \\ 0, if\ x \neq \overline{x}. \end{cases}$$

Note that the supremum of the membership function is achieved only in the case when $x = \overline{x}$, i.e. for $\mu_{A^k}(\overline{x})$ = 1. In this case, we obtain

$$\mu_{\overline{B}^k}(y) = \mu_{B^k}(y) \prod_{i=1}^{n} \mu_{A_i^k}(\overline{x_i}).$$

Let us apply the center average defuzzification method, according to which

$$\overline{y} = \frac{\sum_{k=1}^{N} \overline{y}^k \mu_{\overline{B}^k}(\overline{y}^k)}{\sum_{k=1}^{N} \mu_{\overline{B}^k}(\overline{y}^k)}.$$

In the above formula, $\overline{y}^k$ this is the center of the fuzzy set B_k, i.e. the point at which $\mu_{B^k}(y)$ it reaches its maximum value.

Based on this, we get the equality:

$$\overline{y} = \frac{\sum_{k=1}^{N} \overline{y}^k (\prod_{i=1}^{n} \mu_{A_i^k}(\overline{x_i}))}{\sum_{k=1}^{N} (\prod_{i=1}^{n} \mu_{A_i^k}(\overline{x_i}))}.$$

The final stage in the process of designing a fuzzy control module is to determine the form of representation of fuzzy sets A_i^k, 1, ..., n; k = 1,…, N. For example, it can be a Gaussian function

$$\mu_{A_i^k}(x_1) = \exp(-(\frac{x_i - \overline{x}_i^k}{\sigma_i^k})^2),$$

where the parameters $\overline{x}_i^k$ and σ_i^k have a physical interpretation: $\overline{x}_i^k$ is the center, and σ_i^k is the width of the Gaussian curve [18].

As will be shown below, these parameters can be modified during training, which allows changing the position and structure of fuzzy sets.

Now we combine all the presented elements, and the fuzzy control module takes the final form as a result.:

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

Each element of this expression can be specified in the form of a function block (sum, product, Gaussian function), which, after the appropriate combination, allows you to create a multi-layer network. (fig. 2).

The diagram shows a control module with four inputs (n = 4). Layers are designated L1 through L4 and are highlighted in gray. Elements denoted by P (multipliers) multiply all input signals, elements denoted by ∑ (adders) add them, and element (a / b) divides one signal by another. Black named dots placed on links represent the weights of those links. Elements of layer L1 implement the Gaussian function with parameters $\overline{x}_i^k$ and σ_{ik}. Expressions and arrows placed above the diagram determine the direction of propagation of the signal and its interpretation.

In genetic algorithms (GA), the control parameters are, as a rule, the values of the probability of execution of the genetic crossing-over (Pc) and mutation (Pm) operators, as well as the population size [19, 20]. The interaction of the blocks of the fuzzy genetic algorithm is shown in Fig. 3.

In addition, it is obvious that the efficient operation of a genetic algorithm depends on the correct selection of its parameters, which can be an extremely difficult task. Statically specified constraints can lead to the omission and reduction of solutions with a high potential for further transformations, which will certainly affect the result of the algorithm. An excessively high or low probability of using the mutation operator often leads to an early hit of the algorithm in a state of local optimum.

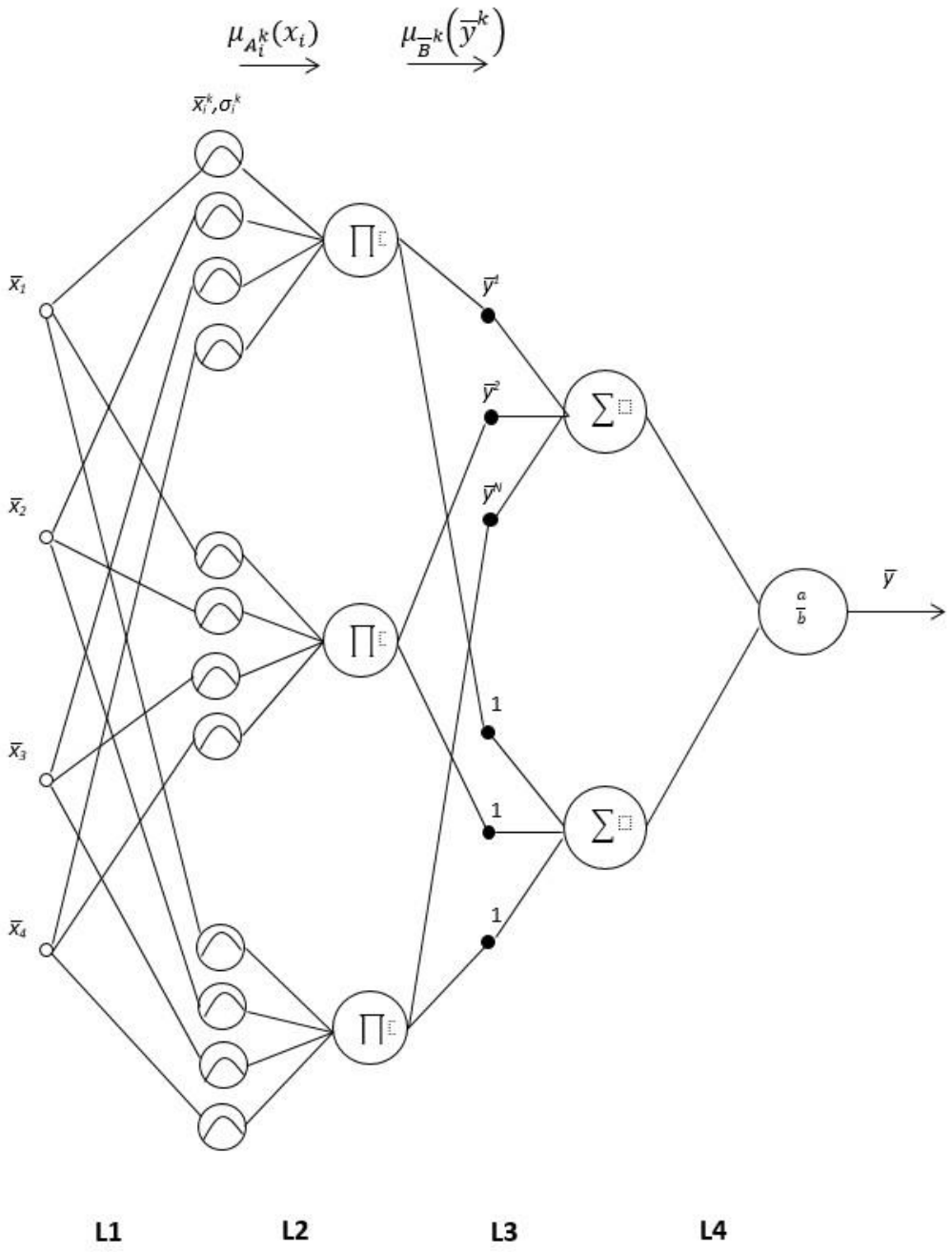

Figure 2: Neural network structure

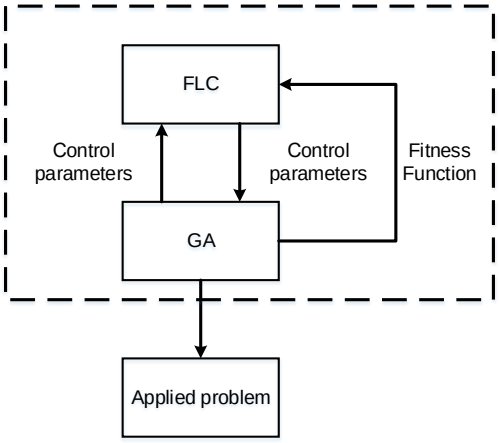

Figure 3: The scheme of interaction of blocks of a fuzzy genetic algorithm

One of the ways to implement a flexible adaptive system of constraints is the use of a fuzzy logic mechanism in the form of a controller - a module that controls dynamic variables during the operation of the system and directs the flow of execution of the software implementation of the algorithm.

Dynamically changing constraints can help avoid this situation, however, the creation of such an adaptive system is a very complex task, since there is a problem of the connection of these constraints with the subject area, and the impossibility of their application for a wide class of problems.

In this paper, we propose the use of a hybrid algorithm based on the previously described approaches. The general scheme of the proposed hybrid algorithm is shown in Figure 4.

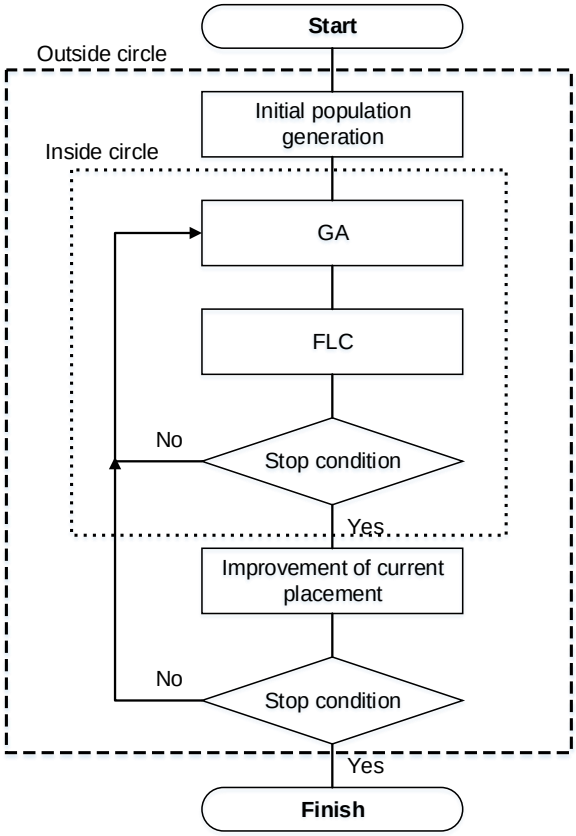

Figure 4: General scheme of the hybrid algorithm

The hybrid algorithm starts by specifying the initial data, determining the optimization criterion and problem constraints. After that, a starting population of solutions is created (initial placement) and the quality of the initial population (both individual individuals and the average value for the population) is assessed from the point of view of the chosen optimization criterion.

After that, the genetic algorithm is executed, a new population of solutions is created, and the control parameters of the algorithm can be adjusted using a fuzzy logic controller.

The next block is a modified ant colony algorithm. Its task is to try to improve the resulting placement.

At the end, the execution of the algorithm stopping criterion is checked, the conditions are not met, the algorithm proceeds to the next iteration, otherwise the execution of the algorithm ends.

Changing the arrangement of elements on the working area continues until the current value of the objective function improves.

Thus, the algorithm consists of three main modules - genetic, ant colony and fuzzy logic controller (FLC) module.

4. Results of computational experiments

The LEF / DEF specification is used to store PCB layout data. LEF (Library Exchange Format) is a specification for representing the physical structure of an integrated circuit in ASCII format. It includes layout rules and abstract information about elements. LEF is used in conjunction with the

Design Exchange Format (DEF) specification, which is used to represent the complete layout of an integrated circuit.

The studies were carried out on two hardware configurations: Intel® Core (TM) i7-3630QM CPU @ 2.40 GHz, RAM - 8GB (configuration 1) and Intel® Core (TM) 2 Quad CPU Q8200 @ 2.40 GHz, RAM - 4 GB (configuration 2). During the study, 5 experiments were carried out with the number of elements from 100 to 3000 with a step of 100. With a constant number of chains equal to 50, the number of iterations equal to 50, the chromosomes of the population equal to 20 and 2 evolutionary processes. When executing the algorithm, a fuzzy logic controller was used.

The results of the experiments are shown in Figure 5. They show the dependences of the average execution time of the algorithm on the number of elements to be placed.

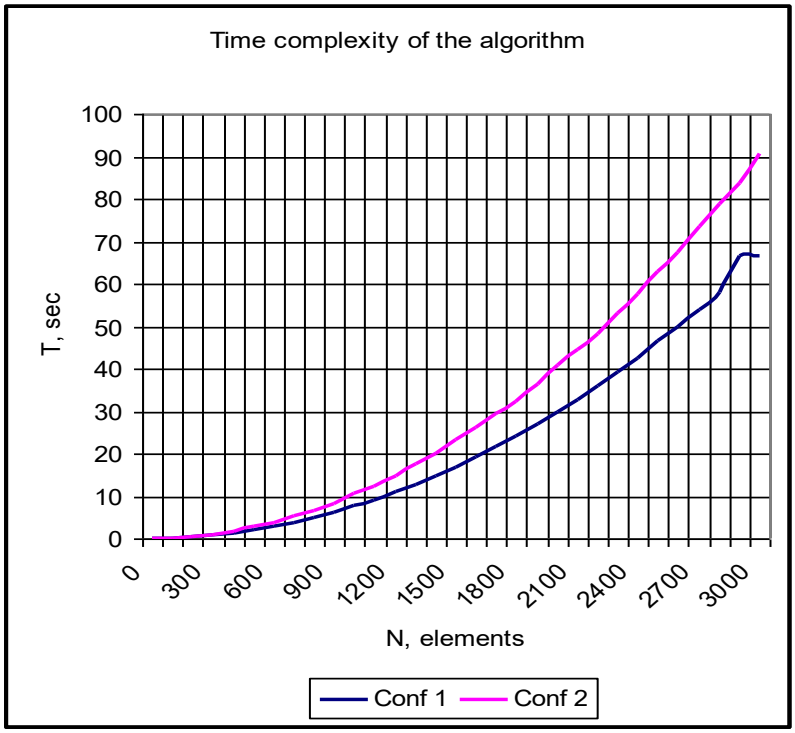

Figure 5: Dependence of the execution time of the algorithm on the number of elements

The results of the proposed algorithm were compared with the use of a fuzzy logic controller and without it. The results are shown in Figure 6, 7.

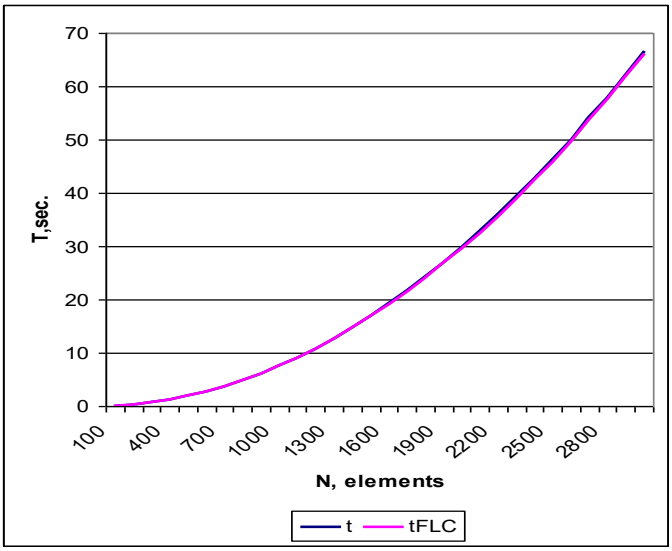

Figure 6: Dependence of the execution time of the algorithm on the number of placed elements with and without a controller

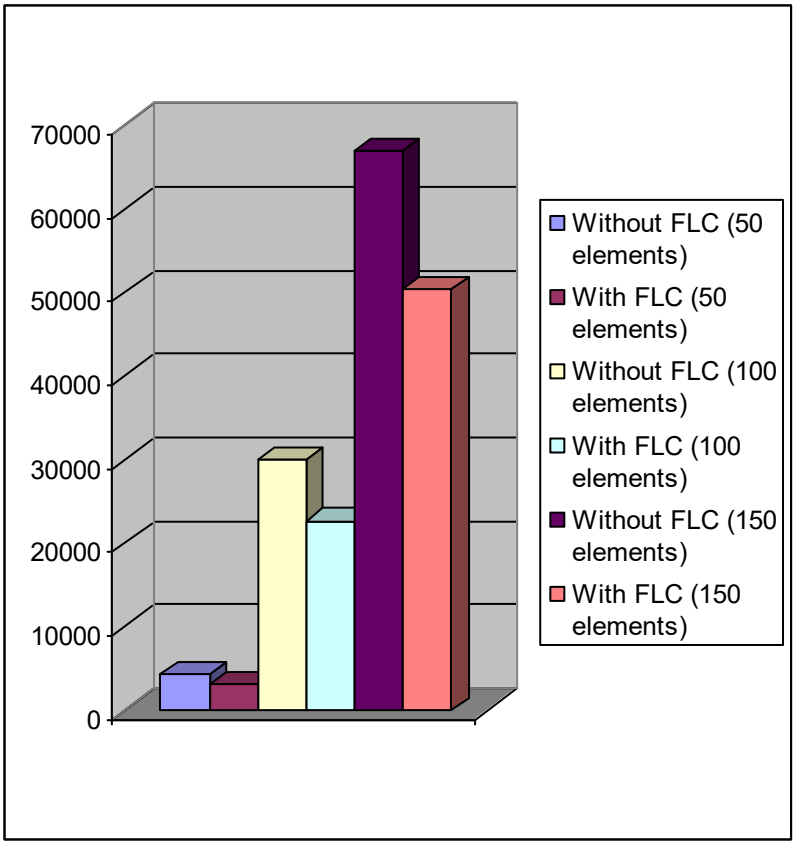

Figure 7: Comparison of the quality of solutions with and without a controller

After analyzing the data presented above, we can conclude that the use of FLC makes it possible to improve the result of solving the problem with the same number of iterations by an average of 25%. The efficiency of using the controller increases after the introduction of a learning unit based on an artificial neural network model.

5. Acknowledgements

The reported study was funded by RFBR according to the research project № 19-01-00715.

6. References

[1] J.P. Cohoon, J. Karro, J. Lienig, Evolutionary Algorithms for the Physical Design of VLSI Circuits. Advances in Evolutionary Computing: Theory and Applications, Ghosh, A., Tsutsui, S. (eds.) Springer Verlag, London, 2003.
[2] Charles J. Alpert, Dinesh P. Mehta, Sachin S. Sapatnekar, Handbook of algorithms for physical design automation. CRC Press, New York, USA, 2009.
[3] N. Shervani, Algorithms for VLSI physical design automation. USA, Kluwer Academy Publisher, 1995.
[4] L.A. Gladkov, V.V. Kurejchik, V.M. Kurejchik, Diskretnaya matematika. M.: Fizmatlit, 2014.
[5] I.V. Prangishvili, Sistemnyj podhod i obshchesistemnye zakonomernosti. – M.: SINTEG, 2000.
[6] N. Yarushkina, Fundamentals of the theory of fuzzy and hybrid systems. Moscow: Finance and Statistics, 2004.
[7] I. Batyrshin, etc. Fuzzy hybrid systems. Theory and practice. Moscow: Fizmatlit, 2007.
[8] H. Haken, Synergetics, an Introduction: Nonequilibrium Phase Transitions and Self-Organization in Physics, Chemistry, and Biology, 3rd rev. enl. ed. New York: Springer-Verlag, 1983.

[9] A.P. Karpenko, Sovremennye algoritmy poiskovoj optimizacii. Algoritmy, vdohnovlennye prirodoj. – M.: izd-vo MGTU im. Baumana, 2016.
[10] L.A. Gladkov, N.V. Gladkova, S.N. Leiba, N.E. Strakhov. ”Development and research of the hybrid approach to the solution of optimization design problems”. Advances in Intelligent Systems and Computing, v. 875. International Conference on Intelligent Information Technologies for Industry IITI'18, vol. 2, Springer Nature Switzerland AG (2019): pp. 246-257.
[11] L. Gladkov, Vl. Kureychik, V. Kureychik, Genetic Algorithms. Moscow: Fizmatlit, 2010.
[12] L. Gladkov, Vl. Kureychik, V. Kureychik, P. Sorokoletov, Bio-inspired methods in optimization. Moscow: Fizmatlit, 2009.
[13] V. Borisov, V. Kruglov, A. Fedulov, Nechetkie modeli i seti (Fuzzy models and networks). Goryachaya liniya – Telekom, Moscow, 2007.
[14] F. Herrera, M. Lozano. ”Fuzzy Adaptive Genetic Algorithms: design, taxonomy, and future directions”. Soft Computing 7, Springer-Verlag, (2003): p. 545 – 562.
[15] A. Michael, H. Takagi. ”Dynamic control of genetic algorithms using fuzzy logic techniques”. Proc. of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann (1993): pp. 76-83.
[16] R.T.F.A. King, B. Radha, H.C.S. Rughooputh. ”A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration”. Proc. of 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan. (2004): p. 577 – 582.
[17] F. Herrera, M. Lozano. ”Adaptation of genetic algorithm parameters based on fuzzy logic controllers”. In: F. Herrera, J. L. Verdegay (eds.) Genetic Algorithms and Soft Computing, Physica-Verlag, Heidelberg (1996): pp. 95-124.
[18] D. Rudkovskaya, Nejronnye seti, geneticheskie algoritmy i nechyotkie sistemy. / D. Rudkovskaya, M. Pilinskij, L. Rutkovskij. – M.: Goryachaya liniya – Telekom, 2006.
[19] L.A. Gladkov, N.V. Gladkova, N.Y. Gusev, N.S. Semushina. ”Integrated approach to the solution of computer-aided design problems”. Proceedings of the 4th International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’19). Advances in Intelligent Systems and Computing, vol. 1156, Springer, Cham (2020): pp. 465-476.
[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, (2018): pp. 246–257.