Received: 4 September 2019 | Revised: 31 January 2020 | Accepted: 2 March 2020

DOI: 10.4218/etrij.2019-0412

ORIGINAL ARTICLE

ETRI Journal WILEY

# Hybrid genetic-paired-permutation algorithm for improved VLSI placement

**Vladimir V. Ignatyev**[1] | **Andrey V. Kovalev**[2] | **Oleg B. Spiridonov**[1] | **Viktor M. Kureychik**[3] | **Alexandra S. Ignatyeva**[3] | **Irina B. Safronenkova**[4]

[1]Department of the Design Bureau of Modeling and Controlling Systems, Southern Federal University, Rostov Oblast, Russia

[2]Engineering Center of Radio and Microelectronic Instrument Making, Southern Federal University, Rostov Oblast, Russia

[3]Department of Computer-Aided Design Systems, Southern Federal University, Rostov Oblast, Russia

[4]Federal Research Center, The Southern Scientific Center of the Russian Academy of Sciences, Rostov-on -Don, Russian Federation

**Correspondence**
Vladimir V. Ignatyev, Department of the Design Bureau of Modeling and Controlling Systems, Southern Federal University (SFU), Rostov Oblast, Russia.
Email: vova3286@mail.ru

**Funding information**
This work was supported by the project "Creating a high-tech production of hardware and software systems for processing agricultural raw materials based on microwave radiation" (Agreement with the Ministry of Education and Science of the Russian Federation, No 075-11-2019-083, dated 20.12.2019; Agreement Southern Federal University, No 18, dated 20.09.2019; work number in Southern Federal University, No HD/19-25-RT).

This paper addresses Very large-scale integration (VLSI) placement optimization, which is important because of the rapid development of VLSI design technologies. The goal of this study is to develop a hybrid algorithm for VLSI placement. The proposed algorithm includes a sequential combination of a genetic algorithm and an evolutionary algorithm. It is commonly known that local search algorithms, such as random forest, hill climbing, and variable neighborhoods, can be effectively applied to NP-hard problem-solving. They provide improved solutions, which are obtained after a global search. The scientific novelty of this research is based on the development of systems, principles, and methods for creating a hybrid (combined) placement algorithm. The principal difference in the proposed algorithm is that it obtains a set of alternative solutions in parallel and then selects the best one. Nonstandard genetic operators, based on problem knowledge, are used in the proposed algorithm. An investigational study shows an objective-function improvement of 13%. The time complexity of the hybrid placement algorithm is $O(N^2)$.

**KEYWORDS**
evolutionary algorithm, genetic algorithm, multi-objective optimization, VLSI design, VLSI placement

## 1 | INTRODUCTION

Very large-scale integration (VLSI) is one of the most widely used technologies for microchip processors, integrated circuits (IC), and component design. In today's world, VLSI chips are widely used in various branches of engineering; for example, voice and data communication networks, digital signal processing, computers, commercial electronics, automobiles, medicine, etc.

In the present stage of technological development, VLSI designs are very complicated and have high dimensionalities. In this context, applying traditional computer-aided design algorithms is ineffective because of the length of the computational process. Moreover, new trends in VLSI manufacturing technology restrict the usage of traditional design methods and approaches. VLSI technologies have multiple advantages, such as compactness, low cost, low power consumption, high reliability, and wide functionality [1,2]. As modern VLSI design technologies play a major role in the manufacturing of high-tech electronic circuit boards, developing new computer-aided design methods and algorithms is an urgent task [3].

VLSI placement is a crucial step in modern physical VLSI design. Formally, placement involves computing an optimal variant for arranging the components on the board, in accordance with a set of criteria [4].

The latest research in VLSI design shows that a great many scientists are focused on developing methods and algorithms for multi-objective VLSI placement optimization. This is because the VLSI placement determines the speed and quality of the routing. Consequently, an optimal element arrangement on a chip increases the reliability of the entire designed product. It also reduces the size of the structural units and minimizes mutual pickups, signal delays, and the lengths of the interconnections [5].

Classical methods for multi-objective problem-solving in computer-aided design (CAD) are branch and bound methods, scalarizing objective functions by using weighting coefficients, and sequential optimization [6,7]. Reducing a multi-objective problem to a single-objective problem with sequential optimization has several disadvantages.

First, the solution to a single-objective optimization problem depends on the parameter restrictions, which are imposed by an engineer. If the restrictions are incorrect, a solution will have no practical meaning, and/or will not produce an optimal result. Moreover, better solutions may exist, which will not be considered.

Second, many optimization algorithms used in modern industrial CAD systems are based on single-criterion optimization methods, whereby a set of criteria are optimized sequentially. Therefore, the result largely depends on the order of the optimization criteria. When sequential optimization takes place, multi-objective optimization (MOO) loses its meaning.

In CAD, the broadcast method of MOO problem-solving is a natural process with system simulations. It generally involves evolutionary and genetic algorithms (GAs), annealing modeling algorithms, various swarm-intelligence methods, and others [8,9]. Hybrid algorithms combine GAs with other search methods. With this method, an alternative solution is optimized in each generation. Then, the best solutions are exchanged. This mechanism is relatable to Lamarck evolution, when each alternative solution can learn, and then transfer new "skills" to a new generation [10,11].

For example, in [12,13], evolutionary and co-evolutionary methods were developed to reduce the soft error rate, which applies to chips in aerospace applications. The proposed methods were based on a GA and particle swarm optimization (PSO) and improved the convergence rate and cost of the former methods.

Algorithm integration allows us to use the advantages of both algorithms. To overcome local-optima problems, an approach based on a combination of genetic global search algorithms and local search algorithms appears to be very promising. The consistent performance of global and local search algorithms suggests that they are the simplest and most effective methods of combining standard cells. A sequential strategy is the frame for the hybrid algorithm proposed in this paper.

## 2 | LITERATURE REVIEW

As VLSI technologies have developed, new design trends have appeared. Because of element size and signal delay reduction, over 80% of the total time delay now corresponds to interconnection delays. In this context, placement is becoming increasingly important, and new methods are required [14,15].

The main advantage of evolutionary methods is that they provide a set of alternative solutions for parallel computing. They are a powerful tool for overcoming local optima [16,17].

One solution for VLSI placement is PSO [18,19]. This approach is very effective when solving for an optimum at the initial placement stage.

Maji and others [20] proposed an evolutionary algorithm called craziness-based PSO (CRPSO) for optimizing the floor planning of a VLSI chip. It was used to speed up local searches and improve the precision of the solution. The main objective of floor planning optimization is to minimize the chip area and the interconnection wirelength.

Laudis and others [21] focused on the multi-objective bat algorithm (MOBA), a biologically inspired metaheuristic, and successfully used it to improve the floor planning in VLSI design. The algorithm's key idea is to simulate bats' echolocation abilities to optimize the placement. The placement was considered as a MOO problem, wherein equal

importance was given to wirelength minimization and dead-space minimization.

To improve the quality of a VLSI implementation, Basir-Kazeruni [22] proposed a new temperature-uncertainty formulation during the clock-tree synthesis process called stochastic perturbation-based clock optimization (SPECO). The study considers the effect of a chip's temperature variation on the synchronization quality, which in turn has a great influence on the speed of modern VLSI chips. Experimental results showed a significant reduction in the dephasing and computational complexity.

Later, Yang and associates [23] proposed a new mixed algorithm that combined the ant-colony algorithm and the tabu search algorithm to improve the net routing design scheme in a VLSI physical design. The results showed that the new algorithm avoided the low convergence rate in the initial stage of a basic ant-colony system. The efficiency of the tabu-ant colony system improved by approximately 16.667%, compared to existing approaches. Moreover, it effectively avoided local optima.

Subhojit and Susovon [24] presented an efficient technique for designing a fixed-order compensator for compensating the current-mode control architecture of DC-DC converters. The proposed algorithm involved combining a population search-based optimization approach, specifically, PSO, and a local search-based method.

Although many scientists have tried to improve VLSI placement by developing new algorithms and techniques, the high computational complexity problem has not been addressed until recently.

Martins and associates [25] focused on the concept of MOO while automating the placement in analog integrated circuit layout design. An innovative archive-based multi-objective simulated-annealing algorithm, operating over an absolute representation, is proposed to optimize the placement of each proximity group. In contrast to traditional single-objective placement approaches, the resulting Pareto fronts, representing the trade-offs between the optimization objectives of each group, are combined, bottom-up, through the design hierarchy, until a final front is obtained. In this manner, the problem's complexity is reduced, and split over multiple executions of the optimization kernel with fewer design variables. Moreover, the analog designer becomes aware of the design trade-offs.

Taher and others [26] presented PASSIOT, a new approach for multi-objective-optimized synthesis of analog circuits, based on computing Sobol' indices for the vectors of their input variable parameters. PASSIOT was simulated using real optimization functions and was proven competitive in terms of runtime and solution quality.

In [27], to efficiently optimize a multi-objective thermally aware nonslicing floor planning method, an adaptive hybrid memetic algorithm was presented to optimize the area, total wirelength, and maximum and average temperatures of a chip. In [27]'s proposed algorithm, a genetic search algorithm is used as a global search method to explore the search space as much as possible. A modified simulated-annealing search algorithm is used as a local search method to exploit the information in the search region. The global exploration and local exploitation are balanced by a death-probability strategy. Experimental results on standard test benchmarks showed that the proposed algorithm efficiently obtained floor plans, while decreasing the average and peak temperatures.

In [28], the authors presented a smart decision-making hybrid PSO-GA that tried to reduce the area, wirelength, and hotspots by distributing the temperature evenly across the chip. A B* tree was used to generate the initial floor plan, and later, a PSO-GA–based hybrid algorithm was used to obtain an optimal placement solution. Temperature-driven floor planning was considered at the perturbation stage to separate the hotspots, thereby reducing the average and maximum temperatures. The experimental results showed that the proposed algorithm created an efficient floor plan, with reduced average and peak temperatures.

In [29], the authors described a two-phase method that combined an ant-colony algorithm and a simulated-annealing method to conduct VLSI placement with a fixed distance between placement rows.

## 2.1 | Genetic algorithms for VLSI placement

Genetic algorithms are stochastic search algorithms. However, they have evolutionary development strategies, which are based on programmable selection and reproduction mechanisms, leading to near-optimal solutions. The effectiveness of a GA depends largely on the specificity of the current task, and applying new and modified search procedures [30]. The specificity of the current problem is considered during the structure development, and while creating the chromosome encoding and decoding principles. When a GA is developed, it is desirable to have homologous chromosomes (chromosomes with an identical structure and length, and containing information on an identical set of solutions). It prevents nonrealistic solutions and simplifies the genetic-operator execution process [31].

Singh and Jain [32] proposed an intellectual approach for VLSI placement, which was based on a heuristic placement strategy and their entropy-based intelligent genetic algorithm (EBIGA). The concept of entropy was introduced in the GA to resolve the problem of finding local optima. The experimental results demonstrated that EBIGA could achieve optimal and competitive solutions for both fixed-outline and outline-free floor plans.

In [33], artificial intelligence-based nature-inspired techniques, such as ant-colony optimization (ACO), modified ant-colony optimization (MACO), artificial

bee-colony optimization (ABC), bat, and firefly algorithms were proposed to perform effective test scheduling, thereby reducing the total cost of the chip. The performance of the various algorithms was evaluated, and it was concluded that the bat algorithm performed best in reducing the overall testing time of a system-on-a-chip (SoC). Hence, the SoC cost was also reduced.

In [34], the ACO, MACO, ABC, Modified ABC, Firefly, and Modified Firefly test-scheduling algorithms were tested on two SoC benchmark circuits. When compared with the ACO, Modified ACO, ABC, Firefly, and Modified Firefly algorithms, the Modified ABC algorithm's testing time was faster by 82%, 69%, 25%, 43%, and 48% for the d695 SoC, and 80%, 73%, 20%, 41%, and 47% for the p22810 SoC, respectively.

The advantage of GAs is a linear evaluation of the time and space complexities of the genetic procedures that are performed during each iteration. This enables the solution of high-dimensional problems, and is important for the VLSI design process. Many methods are available for selecting a chromosome-mutation engine, and for selecting solutions from the population for the next iteration, depending on how they influence the algorithm convergence.

It is possible to improve GA efficiency by using its parallelization and subsequent chromosome migration among the populations [35]. Although GA has natural parallelism [36], this ability complicates the algorithm implementation, increasing its time and space complexity.

Genetic algorithms also have disadvantages. During a genetic search, many solutions, which are dispersed across the search space, are examined at the beginning of the search. However, some solutions that are worse in comparison with others, but belong to an area with points of global optimum, can be lost.

Another problem with a genetic search is that the solutions from the evolving population are near-optimal. Genetic search mechanisms, which realize stochastic variations, do not often find the chain of changes that lead to an optimum. Hence, reasonable changes are needed, which will achieve an optimum.

Analyses have presented a great variety of methods and approaches for VLSI placement, indicating the relevance of the current research. However, placement optimization with regard to the interconnection delay has not been fully covered in the reviewed literature. Therefore, this domain should continue to be studied.

The current study proposes a new hybrid VLSI placement algorithm that involves a sequential strategy combining global and local searches.

## 3 | PROPOSED METHOD

The additive approach is a well-known method of forming complex placement criteria. Each partial optimization criterion contributes to the total value of the complex objective function. Then, the significance of its contribution is determined by the user with the help of a weighting factor for each partial criterion [37].

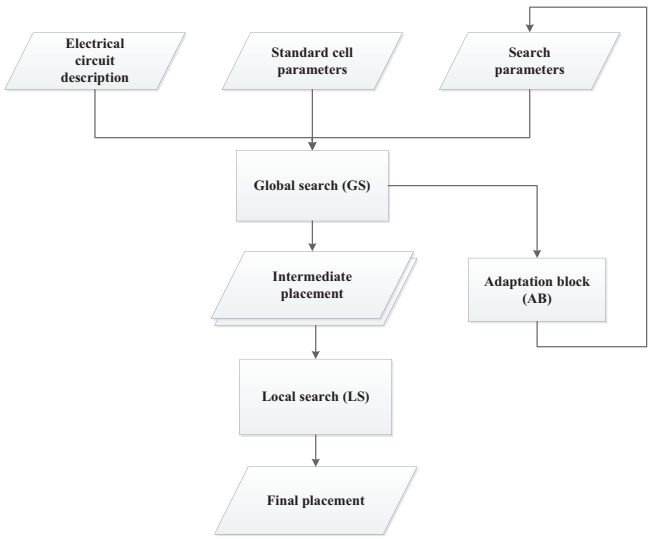



**FIGURE 1** Flowchart of the computational process

For additive criterion, the complex objective function is as follows:

$$F = \alpha f_1^{\mathrm{H}} + \beta f_2^{\mathrm{H}} + \gamma f_3^{\mathrm{H}} \rightarrow \mathrm{opt}, \quad (1)$$

where $\alpha$, $\beta$, and $\gamma$ are significant weighting factors and satisfy the conditions $\alpha + \beta + \gamma = 1$; $f_1^{\mathrm{H}}$ is the normalized indicator of the total circuit length, which is defined as the sum of the semi-perimeters of the describing rectangles; $f_2^{\mathrm{H}}$ is the normalized indicator of the number of critical circuits, where the delay exceeds the maximum allowed; $f_3^{\mathrm{H}}$ is the normalized routability indicator, which corresponds to the total area of all intersecting areas, which are formed by the describing rectangles of the circuits.

The proposed VLSI placement algorithm consists of two stages. In the first stage, a global search is performed using the modified GA. Subsequently, a local search takes place based on the paired-permutation algorithm. A flowchart of the computational process is shown in Figure 1.

The input data for the hybrid algorithm are as follows:

- Electrical schematic, which includes the information about a set of topographical features and a list of circuits that will be implemented in a standard cell;
- Parameters of the designed standard cell, such as the physical dimensions of the work field, step of an approximating grid, and manufacturing technology;
- Search parameters, including GA parameters, weighting factors, and a number of solutions that will be improved with the help of a local search algorithm.

A global search is performed with a modified GA, which provides a set of midline placements. A local search involves

launching the paired permutation (PP) algorithm for each of the received placements. Then, the best solution is selected from among the local search results.

The adaptation block accomplishes the following functions:

- Changes the probability of crossover and mutation operators during the GA execution to increase the population diversity at the beginning of the algorithm (by increasing the mutation probability). It also increases the algorithm convergence rate for the last of the iterations (by increasing the probability of crossovers);
- Defines the number of interim placements, which are improved by the algorithm, depending on the population diversity and size.

The placement of a standard cell is described with a chromosome, called an alternative solution (AS), each gene of which corresponds to a fixed position in the work field.

Each gene contains information about which topographical feature is assigned to its corresponding position. Hence, the gene value can change within the limits of the element numbers. The gene values cannot be repeated in the chromosome because an element cannot be assigned to multiple positions simultaneously. All gene values in a chromosome must be unique; hence, these chromosomes are nonhomologous and require using modified genetic operators, which do not cause illegal solutions [38].

An assessment of the decision quality is conducted in accordance with the complex objective function. The algorithm for calculating a solution to the objective function is written as follows:

For each placement $i$ in the population:

1. Evaluate $f_{1i}$, $f_{2i}$, and $f_{3i}$ partial objective functions.
   1.1. If $f_{1i} \geq f_{1\max}$, then $f_{1\max} = f_{1i}$.
   1.2. If $f_{2i} \geq f_{2\max}$, then $f_{2\max} = f_{2i}$.
   1.3. If $f_{3i} \geq f_{3\max}$, then $f_{3\max} = f_{3i}$.
2. Evaluate normalized partial objective functions for placement $i$:

$$f_{1i}^{\mathrm{H}} = \frac{f_{1i}}{f_{1\max}}, f_{2i}^{\mathrm{H}} = \frac{f_{2i}}{f_{2\max}}, f_{3i}^{\mathrm{H}} = \frac{f_{3i}}{f_{3\max}}. \quad (2)$$

3. Evaluate the complex objective function for placement $i$:

$$F_i = \alpha f_{1i}^{\mathrm{H}} + \beta f_{2i}^{\mathrm{H}} + \gamma f_{3i}^{\mathrm{H}}. \quad (3)$$

The procedures for evaluating partial objective functions are performed in linear time with respect to the number of elements, $N$, which should be allocated. Therefore, evaluating one solution to an objective function has the time complexity $O(N)$.

The proposed GA contains crossover, mutation, selection, and aging operators. Because chromosomes are nonhomologous and illegal solutions are not permitted during the algorithm performance, the following modifications of the main operators were chosen: a single-point ordered crossover operator and a two-point mutation operator.

The aging operator (AO) selects only "young" individuals that have been living for less than a given number of iterations. One iteration is equal to 1 year in the life of an individual. The AO enhances the variability of the solutions and acts as a counter. If an individual reaches the maximum age, it dies and is replaced by a new one, with age zero. The individual age neither mutates nor crosses.

The AO helps to decrease the probability of local optima. It controls the individual lifetimes and eliminates solutions that have been living for several iterations.

It is necessary to assign each individual a value equal to zero. If the CO or MO has not been applied to this individual during the GA operation, then, in the new population, its age will increase by one. If its age exceeds a given value, this individual is eliminated.

The crossover point is selected randomly during the ordered crossover operator's (CO) performance. Then, the left segment of the first parent $P_1$ is copied to the first descendant $P'_1$. The remaining gene values of the first descendant $P'_1$ are copied from left to right from the second parent $P_2$, beginning from the cut point, except for the elements that have already been included in $P'_1$. The second descendant $P'_2$ is treated in the same manner, using the second parent. As a result of the CO execution, the descendant with the better objective-function value is chosen. The ordered CO is performed in linear time, with respect to the number of elements, $N$. Hence, its time complexity is $O(N)$. An example of an ordered crossover operation is shown in Figure 2.

Two cut points are selected randomly during the mutation operator (MO) execution. Then, the values are exchanged between genes that were chosen within the chromosome. A two-point mutation is performed in constant time. Therefore, the time complexity of MO is $O(1)$.

A selection operator chooses one solution from the population for the subsequent crossover and mutation operations. The selection, which is based on a roulette wheel, is used in

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $P_1$: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $P_2$: | 7 | 1 | 2 | 5 | 3 | 4 | 6 | 8 |
| $P'_1$: | 1 | 2 | 3 | 4 | 6 | 8 | 7 | 5 |
| $P'_2$: | 7 | 1 | 2 | 5 | 6 | 8 | 3 | 4 |

**FIGURE 2** Example of an ordered crossover operation

the proposed algorithm. In this case, the probability $P(x_i)$ of selected solution $x_i$ is defined by (4):

$$P(x_i) = \frac{F(x_i)}{\sum_{j=1}^{M} F(x_j)}, \quad (4)$$

where $F(x_i)$ is the objective-function value of solution $x_i$; and $M$ is the population size.

The solution selection, which is based on a roulette wheel, is performed in linear time, relative to the population size, $M$. Its time complexity is $O(M)$ [39].

The proposed GA consists of the following procedures:

1. Create the initial solution population.
2. Apply the operations and select the solutions, until the specified number of $G$ iterations is performed:
2.1. Apply an ordered CO to the solutions that were selected from the current population by the selection operator, in accordance with the crossover probability $P$, and obtain new solutions.
2.2. Apply a two-point MO to the solutions that were selected from the current population by the selection operator, in accordance with the probability $P$ of the mutation operator, and obtain new solutions.
2.3. Select new and old solutions based on an elitist strategy for the next generation.
3. Output the set of the best solutions ($D$).

The initial solution population is created randomly and based on the "shotgun" strategy to maximally cover an acceptance placement region and prevent local optima. This procedure is performed in linear time, in relation to the number of elements $N$. It requires the evaluation of the objective function for each new solution; thus, the time complexity of this procedure is $O(M \times N)$.

Traditionally, the probability of CO and MO is determined by the number of new solutions that they create. Thus, new solutions $P(OK) \times M$ are formed by CO and $P(OM) \times M$ are formed by MO. The proportion of $P(OK)$ and $P(OM)$ affects the solution diversity in the population. Creating a new solution requires a selection operator and the recalculation of the objective function for each CO and MO call.

Selecting the solutions for the next generation uses the elitist strategy and is performed as follows. Solutions from the current population are combined with new solutions, which were obtained using the genetic operator application, and are ranked according to the objective-function value. The $M$ best solutions are selected for the new generation. Therefore, the population size is permanent in each generation. The selection for the new generation requires an objective-function recalculation for $M \times 2$ solutions. Hence, the time complexity of the selection procedure is $O(N \times M)$.

After evaluating the execution time of the GA blocks, a theoretical evaluation of the complete algorithm can be conducted. Thus, the GA operating time is proportional to the product of the number of allocated elements ($N$) times the population size ($M$) times the number of genetic search generations ($G$): $N \times M \times G$. $P(CO)$ and $P(MO)$ are the probabilities of CO and MO, respectively. $D$ is the number of interim placements in a hybrid algorithm, which are the search or hybrid algorithm parameters.

The paired-permutation algorithm (PPA), which performs a local search in the solution neighborhood that was obtained by the global search, uses the same solution model and decision-quality assessment method as the proposed GA.

Following is the PPA execution scheme:

1. Sort the allocated elements of the number of links in descending order.
2. Perform the test permutations with the following $(N - i)$ elements for each $i$-th element:
2.1. Evaluate the objective function for each permutation.
2.2. If certain permutations improve the objective function, select the one among them that improves it in the greatest degree.
3. Output the resultant placement.

The described PPA performs $N$ iterations, for each of which, $(N - i)$ permutations are performed, where $i$ is the iteration number. Evaluating the objective function does not require its complete recalculation, which was described above; hence, it is performed in constant time: $O(1)$. Thus, the theoretical evaluation of the PPA time complexity is $O(N^2)$.

Figure 3 shows a flowchart of the developed hybrid placement algorithm. The theoretical evaluation of the hybrid algorithm time complexity is $O(N^2)$.

The program that implements the proposed hybrid placement algorithm for VLSI elements in standard cells was developed using Microsoft Visual Studio 2010 Express and was written in C# for execution in a Windows environment.

The developed program allows us to allocate up to 500 device features, which connect up to 800 circuits. Files of a certain format are used for data transfers between the placement program and the software for other stages of standard cell-topology synthesis. The user interface of the program is shown in Figure 4.

A program window contains a main menu, the input tools for the placement source data, and the tools for presenting the results. The main menu controls the input and output data flow, searches the results for the optimal placement, selects the tracing method (dynamic or static tracing), and provides online help.

The «Search Settings» console is presented in Figure 5. It displays the main algorithm parameters, such as the number of generations in a genetic search, the solution population size, the probability of applying crossover and mutation operators, and the number of interim placements, which were obtained by the GA.

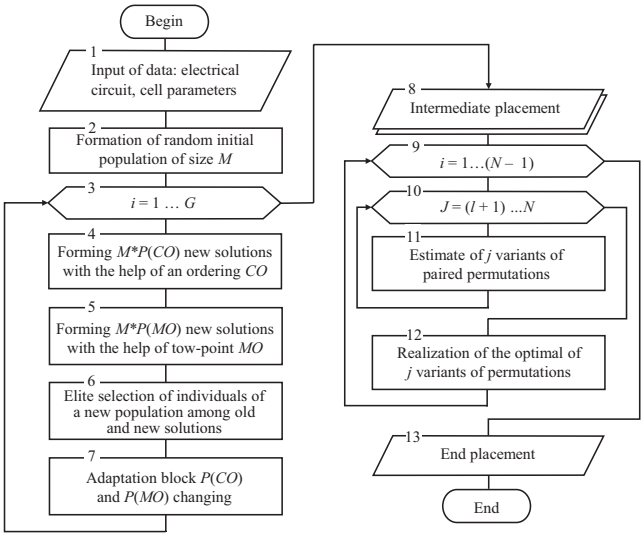



**FIGURE 3** Flowchart of the hybrid-placement algorithm

When the "Automatic configuration" is selected, the genetic-search adaptation unit automatically selects crossover and mutation probability settings, and the number of interim solutions.

The «Placement Criteria» console allows priorities to be set for partial optimization criteria, such as the total length, number of circuits with a critical signal delay, and the traceability while the search takes place. Weight values can be inserted in the editable fields, and determine the contribution of each criterion.

The results of the placement algorithm are displayed on the «Search Results» and «Placement» tabs. The first tab contains a graph of the objective-function changes, the algorithm run time, and the obtained estimates of the partial optimization criteria. The second tab contains a visual representation of the obtained placement.

The «Search Results» tab contains a graph of the objective-function changes that occurred during the placement algorithm performance (Figure 6).

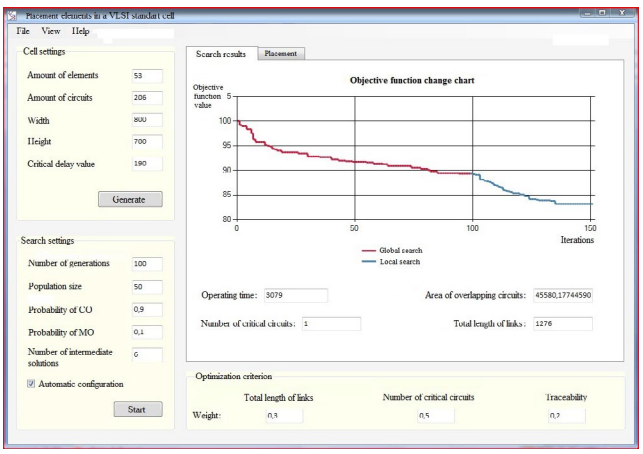



**FIGURE 4** User interface

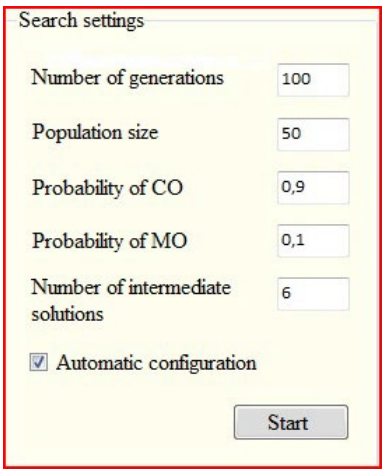



**FIGURE 5** «Search Settings» console

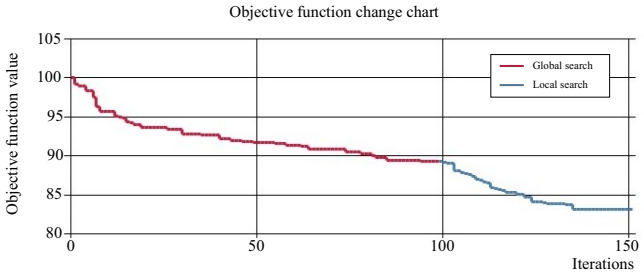



**FIGURE 6** «Search Results» tab

The graph can change dynamically during the algorithm performance (if the corresponding check box is set). In this case, values are scaled dynamically, enabling the optimization process to be visualized more clearly. If the dynamic output is disabled, the graph will change only after the algorithm completes.

The objective-function diagram consists of two parts, corresponding to the different stages of the hybrid algorithm. The solid line corresponds to the global genetic search implementation and the dashed line corresponds to the local search implementation.

The «Placement» tab contains a visual representation of the resultant placement. In the first stage, a regular grid, which is the height and width of a standard cell, is displayed. In the second stage, the transistor output is displayed, in accordance with its number in the list of elements. The circuit display is also provided. The critical circuits, where the signal delay exceeds the specified level, are indicated in red; the others are green. The color intensity of the circuits depends on the density of the links.

The described visualization methods allow us to observe the placement quality from the viewpoint of all of the particular optimization criteria. The number of critical circuits is estimated by the proportion of red and green colors.

The traceability is estimated, based on the color intensity difference between the different areas of the work field.

## 4 | RESULTS

The experimental studies are conducted on the presented hybrid algorithm to determine the parameters that enable the algorithm to achieve globally optimal solutions in the minimum amount of time. They also prove the effectiveness of the proposed algorithm.

Experimental studies were conducted for 50 randomly generated schemes, and for 50 real cell schemes, which were taken from the NanGate library. The randomly generated schemes have the following characteristics:

- Number of elements ≤200;
- Number of circuits ≤800 (it exceeded the amount of elements three or four times);
- Number of united elements in one circuit.

$\alpha$, $\beta$, and $\gamma$ are chosen by the decision maker for $f_1^{\mathrm{H}}$, $f_2^{\mathrm{H}}$, and $f_3^{\mathrm{H}}$, respectively. As a rule, $\alpha = 0.6$, $\beta = 0.2$, and $\gamma = 0.2$; however, they can be changed in compliance with a given technical design specification. In some cases, recommendations and expert systems can be applied.

The 45-nm NanGate Open Cell Library [40] is an open library of 45-nm standard cells, which is designed for testing and researching computer-aided design tools for electronic computing equipment. The library contains all widely used standard cells, from simple buffers to complex triggers, with editable values.

All cells are presented in several variants for standard synthesis routes in the standard cell stages of placement and tracing.

Figure 7A,B shows the placement result, which was obtained using the proposed hybrid algorithm, for a standard cell of the NanGate 45-nm Open Cell Library. Figure 7A demonstrates the initial electrical circuit (19 elements) and Figure 7B shows the obtained placement.

The objective function is improved by 13%, and is shown in Figure 8.

A series of experiments with different numbers of $N$ was performed to estimate the time-expenditure growth. During the experiments, the number of elements was changed from 20 to 1000 in 20-element increments. The number of generations and the size of the population remained constant at 50. The time complexity of the program diagram is shown in Figure 9.

The obtained diagram indicates the quadratic nature of the algorithm's time complexity, which agrees with the theoretical estimation obtained previously.

In general, the amount of memory required for a software algorithm implementation is equal to the amount of memory occupied by the program, output, and intermediate data.

The size of the input and output data is determined by the number of elements, circuits, and other standard cell parameters, which are specified by a user. The size of the intermediate data varies and depends on the search parameters, which are also specified by a user. The minimum and maximum amounts of memory required for the experimental-study implementation are 5 MB and 26 MB, respectively.

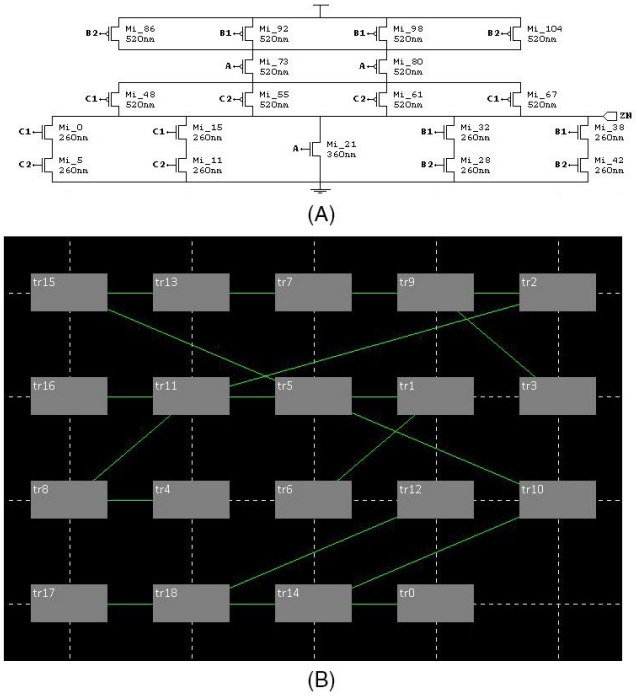



**FIGURE 7** Standard cell placement using NanGate library elements. (A) Initial electric circuit, (B) obtained placement

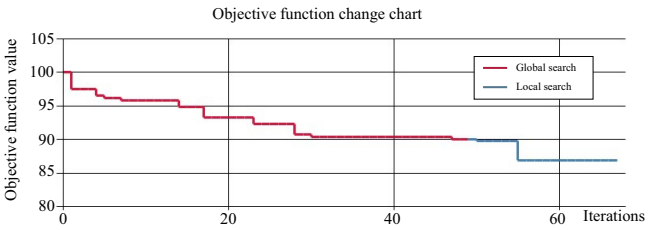



**FIGURE 8** Objective function improvement chart

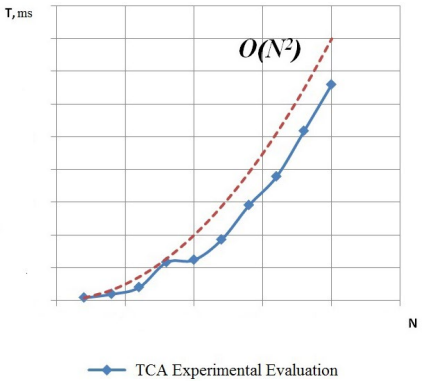



**FIGURE 9** Time complexity of the algorithm

The heuristic algorithm effectiveness considers the degree of the planned action implementation and the achieved required objective-function values [41,42].

In accordance with this definition, the efficiency of the proposed algorithm can be estimated by the number of globally optimal solutions obtained in a single launch, as well as by the number of independent algorithm runs, which guarantee the achievement of the global optimum.

Because a global placement optimum is not known, all solutions that are within 10% of the best solution obtained from the series of launches are considered optimal. The probability of an optimum achievement in at least one independent launch is expressed by the relation:

$$P_1 = 1 - \left(1 - P_2\right)^R, \tag{5}$$

where $P_1$ is the required probability; $P_2$ is the probability of an optimum achievement in a single launch; and $R$ is the number of launches in the series.

When a probability value $P_1$ is specified, for example, 99%, the sufficient number of independent launches can be calculated:

$$R = \left\lceil \frac{\log\left(1 - P_1\right)}{\log\left(1 - P_2\right)} \right\rceil. \tag{6}$$

The probability of achieving an optimum in a single launch, $P_2$, depends on the parameter quality of the genetic search; namely, the number of generations in each launch, $G$, and the population size, $M$. To determine the dependence of the probability of global optimum achievement $P_2$ on the number of search iterations (number of generations), $G$, it is necessary to construct a histogram that reflects the probability distribution density (PDD), and a histogram of the accumulated frequency, which corresponds to the probability distribution integral function (PDIF).

Figure 10 shows the results of the experimental characteristic studies, discussed above. The experiment included 1000 search algorithm runs with a population size $M = 150$. The maximum iteration number $G_{max} = 200$.

With the help of the integral probability distribution function diagram and the time complexity diagram, combination of the optimal population number and number of independent runs in the context of time minimization was found. The population size is $M = 150$, the number of required hybrid algorithm runs is $R = 5$, and the number of generations is $G = 30$.

To determine the optimal population size, $M$, it was necessary to build several combined diagrams and select the optimal point that provided the minimum time, from the optimal points of all graphs.

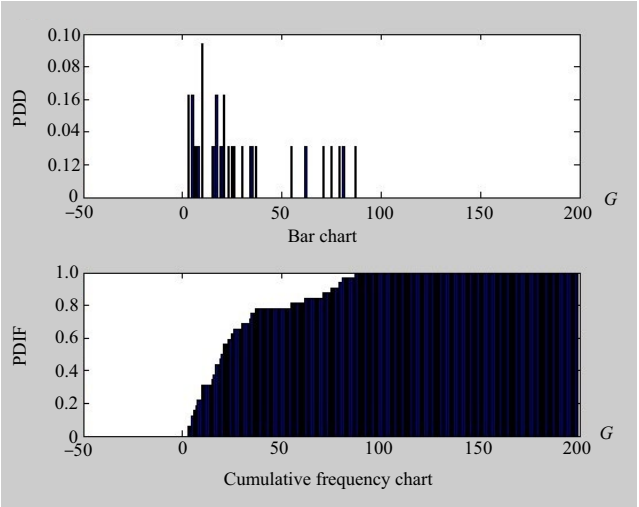



**FIGURE 10** Experimental estimates of the probability distribution functions

## 5 | DISCUSSION

The distinctive feature of the proposed hybrid algorithm is the sequential application of a modified GA and PPA for multi-objective VLSI placement.

For example, compared with [18,19], where an optimal VLSI placement is determined using algorithms based on PSO, the proposed algorithm optimizes the VLSI placement by considering several criteria. Thus, a global placement method is proposed.

Metaheuristics [20,21,29] consider several optimization criteria and provide very good placement results. However, compared with our hybrid algorithm, the metaheuristic approach uses a complex mathematical apparatus, which leads to increased time and computational complexity.

Other studies [23,24,28,33,34] present hybrid metaheuristic algorithms for VLSI optimization. However, they require significant computational resources, compared with the proposed algorithm.

Compared with [32], the proposed algorithm uses nonstandard genetic operators, which are based on a knowledge of the specific problem to be solved. This helps avoid illegal solutions in the interim placement stage.

Using memetic algorithms for optimization [27] is also associated with long-term local searches and requires further study.

## 6 | CONCLUSION

VLSI placement is one of the most important design stages and is quite relevant because of the appearance of new design trends, which require the development of new methods and algorithms.

This paper presented a complex VLSI placement algorithm that accomplished the following:

- Complex placement criteria were synthesized;
- New and modified genetic operators were described by considering the specific placement features;
- A hybrid VLSI placement algorithm, based on sequentially applying parallel-GA and PPA usage, was presented;
- Experimental studies that were carried out showed a 13% objective-function improvement;
- The theoretical and experimental time complexities of the hybrid algorithm were presented ($O(N^2)$);
- An experimental evaluation of the algorithm time and space complexity was described, as well as the test data.

The novelty of the research lies in the choice of operators, which is based on a knowledge of the current problems, alternative solution migrations within several GA iterations, obtaining a set of alternative solutions, and selecting a quasi-optimal one.

The experimental results proved the efficiency of the proposed algorithm for VLSI placement. It demonstrated the usage of the proposed algorithm in the field of VLSI CAD multi-objective decision-making designs.

The list of swarm algorithms (ACO, BCA, cuckoo-search algorithm, wolf-pack search algorithm, and so forth) was assembled by the authors. Further, with the help of hybrid approaches and expert and recommendation systems, we will experimentally evaluate the placement algorithm efficiency, which depends on current problem knowledge.

## ORCID


*Vladimir V. Ignatyev* https://orcid.org/0000-0002-7988-3111

## REFERENCES

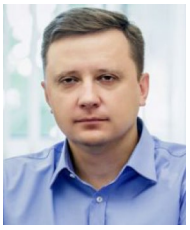
1. L. Jain and G. A. Singh, *Review: Meta-heuristic approaches for solving rectangle packing problem*, Int. J. Comput. Eng. Inf. Technol. **4** (2013), no. 2, 410–424.
2. K. S. P. Kumari, S. Kumar, and S. Sinha, *VLSI systems energy management from a software perspective: A literature survey*, Perspectives Sci. **8** (2016), 611–613.
3. L. Jain and A. Singh, *Non slicing floorplan representations in VLSI floorplanning: A summary*, Int. J. Comput. Appl. **71** (2013), no. 15, 12–19.
4. T. Singha, H. S. Dutta, and M. De, *Optimization of floor-planning using genetic algorithm*, Procedia Technol. **4** (2012), 825–829.
5. A. P. Karpenko, *Sovremennye algoritmy poiskovoi optimizatsii. Algoritmy, vdokhnovlennye prirodoi [Modern algorithms of search engine optimization: Nature-inspired optimization algorithms]*, Moscow, Bauman MSTU Publ. **446** (2014), (in Russian).
6. R. Martins, N. Lourenço, and N. Horta, *Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates*, Expert Syst. Appl. **42** (2015), no. 23, 9137–9151.
7. A. Darwish, *Bio-inspired computing: algorithms review, deep analysis, and the scope of applications*, Future Comput. Inform. J. **3** (2018), no. 2, 1–16.
8. V. Ignatyev et al., *The fuzzy rule base automatic optimization method of intelligent controllers for technical objects using fuzzy clustering*, in Proc. Conf. Creativity Intell. Technol. Data Sci. (Bolgograd, Russia), Sept. 2019, pp. 135–152, https://doi.org/10.1007/978-3-030-29750-3_11
9. V. Ignatyev et al., *System for automatic adjustment of intelligent controller parameters*, in Proc. Conf. Creativity Intell. Technol. Data Sci. (Bolgograd, Russia), 2019, pp. 226–242, https://doi.org/10.1007/978-3-030-29750-3_18.
10. E. Sopov, *Genetic programming hyper-heuristic for the automated synthesis of selection operators in genetic algorithms*, in Proc. Int. Joint Conf. Comput. Intell. (Madeira, Portugal), Nov. 2017, https://doi.org/10.5220/0006497002310238.
11. P. Orzechowski, W. La Cava, and J. H. Moore, *Where are we now? a large benchmark study of recent symbolic regression methods*, in Proc. Genetic Evolutionary Comput. Conf. (Kyoto, Japan), July 2018, pp. 1183–1190.
12. H. Jahanirad and K. Mohammadi, *Reliable implementation on SRAM-based FPGA using evolutionary methods*, IETE J. Research **59** (2013), no. 5, 597–603.
13. H. Jahanirad, *Co-evolutionary approach to reduce soft error rate of implemented circuits on SRAM-based FPGA*, Int. J. Comput. Applicat. **180** (2018), no. 43, 42–49.
14. J. Funke, S. Hougardy, and J. Schneider, *An exact algorithm for wirelength optimal placements in VLSI design*, Integr. VLSI J. **52** (2016), 355–366.
15. V. Kureychik and A. Kulakov, *Algorithm of thermal optimization of placement of basic elements of VLSI*, in Proc. IV Int. Research Conf.: Inf. Technol. Sci., Manag., Social Sphere, Medicine (Tomsk, Russia), Dec. 2017 pp. 63–67.
16. P. Bateson, *Evolution, epigenetics and cooperation*, J. Biosci. **38** (2013), 1–10.
17. M. Rahul, S. Narinder, and S. Yaduvir, *Genetic algorithms: Concepts, design for optimization of process controllers*, Comput. Inf. Sci. **4** (2011), no. 2, 39–54.
18. S. Venkatraman and M. Sundhararajan, *Particle swarm optimization algorithm for VLSI floorplanning problem*, J. Chem. Pharm. Sci. **10** (2017), no. 1, 311–316.
19. B. Xue, M. Zhang, and W. N. Browne, *Particle swarm optimization for feature selection in classification: A multi-objective approach*, IEEE Trans. Cybern. **43** (2013), no. 6, 1656–1671.
20. K. B. Maji et al., *An evolutionary algorithm based approach for VLSI floor-planning*, in Proc. Int. Conf. Sci. Technol. (Pathum Thani, Thailand), Nov. 2015, pp. 248–253.
21. L. L. Laudist et al., *MOBA: Multi objective bat algorithm for combinatorial optimization in VLSI*, Procedia Comput. Sci. **125** (2018), 840–846.
22. S. Basir-Kazeruni et al., *SPECO: Stochastic perturbation based clock tree optimization considering temperature uncertainty*, Integr. VLSI J. **46** (2013), no. 1, 22–32.
23. P. Yang et al., *Optimal approach on net routing for VLSI physical design based on Tabu-ant colonies modeling*, Appl. Soft Comput. **21** (2014), 376–381.
24. S. Ghosh and S. Susovon, *Fixed structure compensator design using a constrained hybrid evolutionary optimization approach*, ISA Trans. **53** (2014), no. 4, 1119–1130.

25. R. Martins, N. Lourenco, and N. Horta, *Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates*, Expert Syst. Appl. **42** (2015), no. 23, 9137–9151.
26. T. Kourany et al., *PASSIOT: A Pareto-optimal multi-objective optimization approach for synthesis of analog circuits using Sobol' Indices-based Engine*, in Proc. IEEE Int. Midwest Symp. Circuits Syst. (Abu Dhabi, United Arab Emirates), Oct. 2016, pp. 16–19.
27. J. Chen et al., *An adaptive hybrid memetic algorithm for thermal-aware non-slicing VLSI floorplanning*, Integr. **58** (2017), 245–252.
28. P. Sivaranjani and A. Senthil Kumar, *Thermal-aware non-slicing VLSI floorplanning using a smart decision-making PSO-GA based hybrid algorithm*, Circuits Syst. Signal Process. **34** (2015), no. 11, 3521–3542.
29. J. Chen et al., *Combining the ant system algorithm and simulated annealing for 3D/2D fixed-outline floorplanning*, Appl. Soft Comput. **40** (2016), 150–160.
30. B. Xue et al., *A survey on evolutionary computation approaches to feature selection*, IEEE Trans. Evolutionary Comput. **20** (2016), no. 4, 606–626.
31. W. A. Albukhanajer, J. A. Briffa, and Y. Jin, *Evolutionary multi-objective image feature extraction in the presence of noise*, IEEE Trans. Cybern. **45** (2015), no. 9, 1757–1768.
32. A. Singh and L. Jain, *VLSI floorplanning using entropy based intelligent genetic algorithm*, Comput., Anal. Netw. (Chandigarh, India), July 2018, pp. 53–71.
33. G. Chandrasekaran, S. Periyasamy, and P. R. Karthikeyan, *Minimization of test time in system on chip using artificial intelligence-based test scheduling techniques*, Neural Comput. Applicat. **32** (2019), 5303–5312. https://doi.org/10.1007/s00521-019-04039-6
34. G. Chandrasekaran, S. Periyasamy, and P. R. Karthikeyan, *Test scheduling for system on chip using modified firefly and modified ABC algorithms*, SN Appl. Sci. **1** (2019), no. 1, https://doi.org/10.1007/s42452-019-1116-x.
35. E. Hancer et al. *A multiobjective artificial bee colony approach to feature selection using fuzzy mutual information*, in Proc. IEEE Congr. Evol. Comput. (Sendai, Japan), 25–28, May 2015, pp. 2420–2427.
36. M. Gong et al., *Evolutionary computation in China: A literature survey*, CAAI Trans. Intell. Technol. **1** (2016), no. 4, 334–354.
37. C. Wu, J. Fang, and Q. Li, *Multi-material topology optimization for thermal buckling criteria*, Comput. Methods Appl. Mech. Eng. **346** (2019), 1136–1155.
38. L. A. Gladkov, V. V. Kureichik, and V. M. Kureichik, *Geneticheskie algoritmy [Genetic algorithms]*, 2nd ed, Moscow: Fizmatlit, 2010, 368.
39. N. A. Sherwani, *Algorithms for VLSI physical design automation*, 3rd ed, Kluwer Academic Publisher, USA, (2013).
40. NanGate FreePDK45 Open Cell Library, http://www.nangate.com/?page_id=2325
41. V. V. Kureichik and V. Kureichik, *Integrated VLSI fragment placement algorithm, Izvestiya SFedU*, Eng. Sci. (2015), 196–205.
42. V. V. Kureichik, V. M. Kureichik, and S. I. Rodzin, *Teoriya evolyucionnyh vychislenij [The theory of evolutionary computation]*, OOO Izdatel'skaya firma "Fiziko-matematicheskaya literatura", Moscow, Russia (2012).


## AUTHOR BIOGRAPHIES

**Vladimir V. Ignatyev** graduated from Southern Federal University, Rostov Oblast, Russia, as an engineer of informatics and control in technical systems in 2008. He defended his candidate's thesis in technical sciences, with specialties of "System analysis, control, and Information Processing" and "Automation and control of technological processes and production" in 2011. He is currently a doctoral candidate at Southern Federal University and head of the Department of the Design Bureau of Modeling and Controlling Systems of Southern Federal University. He has authored more than 50 scientific publications. His scientific interests are automated control systems for technological processes, design and development of hybrid control systems, intelligent control systems, and decision support.

**Andrey V. Kovalev** graduated from Taganrog State University of Radio Engineering, Taganrog, Russia, as an engineer of Electronic Computing Design and Technology in 1998. In 2001, he defended his candidate's thesis in the field of microelectronics and CAD, and in 2009, he received the degree of doctor of engineering. He has authored more than 60 scientific publications. He is currently an assistant professor and head of the Engineering Center of Radio and Microelectronic Instrument Making of Southern Federal University. His scientific interests are microelectronics, methods and means of automated VLSI design, and IP core design.



**Oleg B. Spiridonov** graduated from Taganrog State University of Radio Engineering as an engineer of computing machines, complexes, systems, and networks in 1998. In 2001, he defended his candidate's thesis on the "Use of computer technology, mathematical modeling, and mathematical methods in scientific research." He is currently Director of the Design Bureau of Modeling and Controlling Systems of Southern Federal University and has authored over 130 scientific publications. His scientific interests are information-measuring and control systems for the creation of ground-based and onboard-based devices.

**Viktor M. Kureychik** graduated from the Taganrog Radio Engineering Institute, Taganrog, Russia, in 1967 as an engineer of counting and solving instruments and devices. In 1971, he defended his thesis for the degree of candidate of technical sciences. In 1978, he defended his doctoral dissertation, and he has authored more than 700 scientific publications. He is currently a professor at the Department of Computer-Aided Design Systems of Southern Federal University and head of the scientific school, "Theory and Practice of Designing Intelligent Problem-Oriented Information Systems based on Evolutionary Calculations." His scientific interests are the theory of computer-aided design of computer systems based on large and super-large integrated circuits, the construction of intelligent information processing systems, knowledge bases, and expert CAD systems, evolutionary modeling, and genetic algorithms and their applications in CAD.

**Alexandra S. Ignatyeva** graduated from Southern Federal University as an engineer of information security organization and Technology in 2009. She is currently a PhD student at the Department of Computer-Aided Design Systems of Southern Federal University. Her scientific interests are big data processing and intelligent control systems.

**Irina B. Safronenkova** graduated from Southern Federal University as a research teaching fellow in the field of information and computer science in 2019. She is currently a junior research scientist in the Federal Research Center at the Southern Scientific Center of the Russian Academy of Sciences, Rostov-on-Don, Russian Federation. Her scientific interests are intelligent CAD systems, distributed systems, and ontologies.