Co-evolutionary Algorithm for the Placement of VLSI Elements

Oleg B. Lebedev[1], Artemy A. Zhiglaty[2]

[1] *Southern Federal University, 105 Bolshaya Sadovaya st., Rostov-on-Don, 344006, Russia*
[2] *Southern Federal University, 105 Bolshaya Sadovaya st., Rostov-on-Don, 344006, Russia*

Abstract

To improve the efficiency, enhance the convergence of the algorithm and the ability to exit local optima, an approach to co-hybridization of the placement algorithm based on an ant colony is proposed. Simultaneously, several subpopulations evolve in the search space for a solution to the problem. The algorithms differ in search strategies. The authors have developed modifications of the canonical paradigm of the ant algorithm, methods for constructing the interpretation of the solution by agents.

Keywords

VLSI, placement, swarm intelligence, ant algorithm, adaptive behavior, subpopulation, co-evolution, optimization.

1. Introduction

In connection with a decrease in the topological dimensions, an increase in the degree of integration of VLSI and a reduction in design time, there are fundamentally new requirements for the design of VLSI [1-2]. The modern CAD design platform [3] is a complete set of tools for VLSI design. Within the ecosystem of the Synopsys design platform, many algorithms and functions have been merged, in particular the functions of optimization of placement. IC Compiler II is a tool for topology design. It is developed on the basis of a new architecture and data structure, focused on achieving the highest speed of work and high quality results. Particular attention is paid to the support of large complex «systems-on-a-chip» with a multilevel physical hierarchy, containing more than 100 million cells.

Work on optimizing algorithms and reducing design time along the route continues with each new release of the tool. It is worth mentioning that in recent releases, developers have paid special attention to improving algorithms for both global cell placement and their legalization, which provides improved traceability by detailed accounting of trace parameters. Continuous analysis of algorithms and their optimization allows you to improve the quality of the results. Additionally, it is worth noting the development of optimization using parallelization of computations on several cores.

The placement problem [2] can be formulated as follows. A set of rectangular elements (modules) $R=\{r_i|i=1,2,\ldots,m\}$ with terminals (outputs) located on them is given, where m is the number of modules; $W=\{w_i|i=1,2,\ldots,m\}$ is the vector of the widths of the rectangles; $L=\{l_i|i=1,2,\ldots,m\}$ is the vector of the lengths of the rectangles. A set of circuits connecting the terminals of the modules is given. The commutation field (CF) is set, on which the elements can be placed. A rectangular coordinate system XOY is introduced, in which the OX and OY axes coincide, respectively, with the lower and lateral sides of the control panel. It is necessary to place the elements on the checkpoint with the optimization of some quality criteria. Input information includes: a description of the modules, which indicates the shape, dimensions, location of terminals on the modules, a list of circuits indicating the interconnections of the modules and a description of the switching environment (SE) − a list of positions and their coordinates [4]. The output information is a list of X, Y coordinates

Proceedings 19th Russian Conference on Artificial Intelligence (RCAI-2021), October 11-16, 2021, Russia, Taganrog,
EMAIL: lebedev.ob@mail.ru(A.1); artemiy.zhiglaty@gmail.com(A.2)
ORCID: 0000-0001-6678-2947(A.1); 0000-0002-4924-487X(A.2)

of positions on the commutation field in which all modules are placed. The solution to the problem is represented as a set of elements $<X,Y>$, where $X=\{x_i|i=1,2,\ldots,m\}$, $Y=\{y_i|i=1,2,\ldots,m\}$ are the vectors of coordinates of rectangles, (x_i,y_i) – coordinates of the lower left corner of the rectangle along the *OX* and *OY* axes, respectively.

The hypergraph $H=(X, E)$ is used as a model of the circuit, where $X=\{x_i|i=1,2\ldots,n\}$ is the set of vertices modeling elements, and $E=\{e_j|e_j\subset X, j=1,2,\ldots,m\}$ is a set of hypersurfaces modeling chains connecting elements.

The distance between two vertices with coordinates (x_i,y_i) and (x_j,y_j) is determined by the formula: $d_{ij}=|x_i–y_i|+|x_j–y_j|$.

The main task of placement is to create the best possible conditions for subsequent routing. There is no formal definition of this concept. Therefore, they introduce criteria and estimates, the optimization of which leads to the best conditions. By now, the most widespread are estimates of the minimum total length of joints. As an estimate lj of the length of the chain tj modeled by the hyperedge e_j, the following are most often used: the length of the semiperimeter of the rectangle describing the set of vertices e_j.

With this in mind, the optimization criterion is as follows:

$$F=\sum_{j=1}^{m} l_j$$

Existing allocation algorithms are divided into two main classes: constructive and iterative [4]. The first class is characterized by a relative speed, but at the same time a low quality of the solution. The second class is more laborious, but it allows you to get better solutions [4].

Among the iterative algorithms are deterministic and probabilistic. In deterministic algorithms, the change in the partition (solution) is implemented on the basis of a clear, deterministic dependence on the solution being changed. The disadvantage is the frequent hit in the local optimum («local pit»).

In hybrid algorithms, the advantages of one algorithm can compensate for the disadvantages of another. Integration of metaheuristics of population algorithms provides a broader view of the search space and a higher probability of localizing the global extremum of the problem. This approach allows to partially solve the problem of premature convergence, provides a way out of local optima and increases the speed of obtaining the result [5].

The metaheuristic of the ant algorithm (AA) is based on a combination of two techniques: the general scheme is built on the base method, which includes an inline procedure. The basic method is to implement an iterative procedure for finding the best solution based on the mechanisms of the adaptive behavior of the ant colony. An embedded procedure is a constructive algorithm for the agent to construct some specific interpretation of the solution [6]. The AA is used to solve the problem of finding the shortest route in a complete graph. Each ant in the colony forms its own route, which is an interpretation of the solution of a certain problem on the complete graph of the search for solutions (GSS). As the ant moves, it marks a path with a pheromone, and this information is used by other ants to choose a path.

The authors have developed modifications of the canonical paradigm of the ant algorithm: the structure of the GSS and the construction of the representation (interpretation) of the solution on it; methods for constructing an interpretation of a solution by an agent on GSS (development of a constructive algorithm).

To increase the efficiency, enhance the convergence of the algorithm and the ability to exit local optima, an approach to co-hybridization [6] of the placement algorithm based on the model of the adaptive behavior of the ant colony is proposed. Simultaneously, several subpopulations evolve in the search space for a solution to the problem, each of which solves the original optimization problem and has its own optimization strategy. The partitioning problem is solved by two groups A_1, A_2 of agents. Agents of the A_1 group build routes in accordance with the C_1 strategy. Accordingly: the agents of the A_2 group – with the C_2 strategy. Strategies C_1 and C_2 differ in constructive algorithms for splitting the scheme by agents.

To solve the placement problem, a co-evolutionary algorithm based on the ant colony method has been developed. The essence of the co-evolutionary approach is that the evolving population is divided into subpopulations that evolve in parallel [5].

Periodically, agents move from one subpopulation to another, passing on their experience. The co-evolutionary approach provides a broader view of the solution space and a higher probability of localization of the global extremum of the problem [5].

2. Development of a co-evolutionary placement algorithm based on models of adaptive behavior of an ant colony

The complete bipartite graph $D=(X\cup P,E)$, is used as the solution search graph, where $X=\{x_i|i=1,2,\ldots,n\}$ is the set of vertices (first share) corresponding to the set of vertices of the graph $H(X,U)$, and $P=\{p_v|v=1,2,...,n\}$ is the set of vertices (second part) corresponding to the set of positions. E is the set of edges (x_i, w_v) connecting the vertices $x_i\in X$ with the vertices of the set P. $|E|=n^2$ of the solution search graph D, Fig. 1.

The problem of a constructive algorithm is reduced to searching on a complete bipartite graph $D=(X\cup P,E)$, a bipartite subgraph $D^k=(X\cup P,E^k)$, which defines a one-to-one relation between the sets of vertices X and P with the minimum value estimates F. The bipartite subgraph D^k is used as a representation of some k-th solution to the location problem specifies the distribution of the set of vertices X over the set of nodes P.

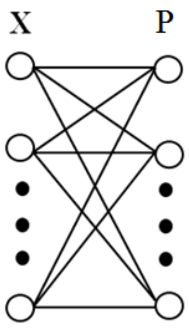

Figure 1: Solution search graph example

The main restrictions of the generated bipartite subgraph $D^k=(X\cup P,E^k)$ are as follows: $|E^k|=|P|=|X|$.

- each edge $(x_i,p_v)=e^k_{iv}\in E^k$, on the one hand, is incident to only one vertex $p_v\in P$, on the other hand, it is incident to one and only one vertex $x_i\in X$;

- the local degree of the vertex $x_i\in X$ is equal to one, $\rho(x_i)=1$;

- the local degree of the vertex $p_v\in P$ is equal to one, $\rho(p_v)=1$.

The paper considers two equivalent constructive placement algorithms that are part of the search placement algorithm. Agents of subpopulation Z_1 build a solution using a constructive algorithm on the set of vertices X. Agents of subpopulation Z_2 use a constructive algorithm on the set of vertices P. Since the constructed graph D^k is completely determined by the set of edges E^k, we will consider E^k as an interpretation of the solution. The process of finding solutions is iterative. Each iteration l includes three stages. At the first stage, each agent z_k forms its own solution E^k. At the second stage, agents deposit (add) a pheromone on the edges of the solution search graph D. To store the pheromone accumulated during the operation of the algorithm, the memory matrix $Q=||q_{iv}||_{n\times n}$ is used, where q_{iv} is the amount of pheromone deposited by the agents on the edge $e_{iv}\in E$. The work uses the cyclical (ant-cycle) method of ant systems. In this case, the pheromone is deposited on the edges of the graph D after the complete formation of solutions at each cycle by all agents of both subpopulations. For these purposes, an auxiliary matrix $\Delta=||\delta_{iv}||_{n\times n}$ is used, where δ_{iv} is the total amount of pheromone deposited by agents during one iteration on the edge e_{iv}.

The subgraph D^k (set E^k) is formed by each agent step by step on the basis of the set E of edges of the complete bipartite graph $D=(X\cup P,E)$ (step by step).

At the initial stage, the same (small) amount of pheromone $\Theta=\xi/\varepsilon$, where $\varepsilon=|E|$, is deposited on all edges of the graph $D=(X\cup P,E)$. In other words, all elements q_{iv} of the memory matrix Q are assigned the value Θ.

Agents have memory. At each step t in the memory of agent z_k there is:

- the list of vertices $X_{1k}(t)\in X$ already located in the positions $P_{1k}(t)\in P$;

– the list of vertices $X_{2k}(t)\in X$ X that remain unplaced, $X_{1k}(t)\cup X_{2k}(t)=X$;

- the list of free positions $P_{2k}(t)\in P$, $P_{1k}(t)\cup P_{2k}(t)=P$;

- $q_{iv}(t)$ − the amount of pheromone, on each edge $e_{iv}=(x_i,p_v)$ of the graph D;

- $\rho_v(t)$ is the local degree of the vertex $\rho_v(t)$ of the graph $D^k(t)$ at step t;

- $\rho_i(t)$ is the local degree of the vertex $x_i(t)$ of the graph $D^k(t)$at step t.

At the first stage of each iteration, first, each of the agents $z_k\in Z_1$, using the first allocation algorithm, sequentially generates a set of edges E^k, where k is the agent's number. Then, by the second algorithm, the sequential formation of the set of edges E^k is performed by each of the agents $z_k\in Z_2$.

Consider the first constructive placement algorithm.

The agents generate solutions E^k on the set of edges E of the graph $D=(X\cup P,E)$.

The memory of the agent z_k is initialized:

$X_{1k}(1)=\varnothing;\ X_{2k}(t)=X;\ P_{1k}(1)=\varnothing;\ P_{2k}(1)=P;\ E^k(1)=\varnothing,\ \forall v(\rho_v(1))=0.\ \forall i(\rho_i(1))=0.$

The auxiliary matrix $\Delta=||\delta_{iv}||_{n\times n}$ is set to zero.

The process of forming E^k includes two stages, which are performed at each step t.

At the first stage of step t, a set of vertices $P_{2k}(t)\subset P$ is formed such that for each vertex $p_v\in P_{2k}(t)$ its current local degree $\rho_v(t)=0$. Similarly, a set of vertices $X_{2k}(t)\subset X$ is formed such that for each vertex $x_i\in X_{2k}(t)$ its current local degree $\rho_i(t)=0$.

For each vertex $x_i\in X_{2k}(t)$, a set of edges $U_i(t)$, $|U_i(t)|=|P_{2k}(t)|$, connecting x_i with the vertices of the set $P_{2k}(t)$ is defined. For each edge $e_{iv}=(x_i,p_v)\in U_i(t)$ incident to the vertex $x_i\in X_{2k}(t)$, the parameter $q_{iv}(t)$ is determined − the total pheromone level on this edge e_{iv}. For each vertex $x_i\in X_{2k}(t)$, the average amount of pheromone per one edge of the set $U_i(t)$ is determined:

$$\mu_{ik}(t)=\sum_v(q_{iv}(t))/\ |U_i(t)|.$$

The value $\mu_{ik}(t)$ is declared the value of the vertex x_i. After that, agent z_k with probability $\Psi_{ik}(t)=\mu_{ik}(t)/\sum_i(\mu_{ik}(t))$, proportional to $\mu_{ik}(t)$, chooses the vertex $x^*_i\in X_{2k}(t)$.

At the second stage of step t, among the edges $U_i(t)$ incident to the selected vertex x^*_i, with probability $\Psi_{vk}(t)=q_{iv}(t)/\sum_v(q_{iv}(t))$, the edge $e^*_{iv}=(x^*_i,p^*_v)$, which is included in the set of edges $E^k(t)$ formed by the agent z_k.

Next, the post-procedures are performed.

$E^k(t+1)=E^k(t)\cup e^*_{iv}$.

$X_{1k}(t+1)=X_{1k}\cup x^*_i;\ X_{2k}(t+1)=X_{2k}\backslash x^*_i$.

$P_{1k}(t+1)=P_{1k}\cup p^*_v;\ P_{2k}(t+1)=P_{2k}\backslash p^*_{vi}$.

The local degree of the vertices x^*_i and p^*_v takes the value 1, $\rho_i=\rho_v=1$.

Go to the next step.

The process of forming the set E^k by the agent z_k ends when $X_{2k}(t)=P_{2k}(t)=\varnothing$.

Calculation of the estimate $F_k(l)$ of the allocation given by the graph $D^k=(X\cup P,E^k)$.

Calculation of the amount of deposited pheromone proportional to the estimate $F_k(l)$.

The second constructive placement algorithm is equivalent to the first. The difference lies in the inverted search space. To do this, symmetric re-indexing of variables and arrays is performed.

$i=v;\ v=i;\ x=p;\ p=x;\ x_i=p_v;\ x_i=p_v;\ e_{iv}=e_{vi};\ (x_i,\ p_v)=(p_v,\ x_i,);$

$X_{1k}(t)=P_{1k}(t);\ P_{1k}(t)=X_{1k}(t);\ X_{2k}(t)=P_{2k}(t);\ P_{2k}(t)=X_{2k}(t);$

$E^k(t)=\{e_{iv}|i=1,2,\dots,n;\ v=1,2,\dots,n\},\ E^k(t)=\{e_{vi}\ v=1,2,\dots,n;\ i=1,2,\dots,n\}.$

The first constructive allocation algorithm solves the problem of assigning the set of vertices X of the original graph $D=(X\cup P,E)$ to the set of vertices P. After the symmetric reindexing of variables and arrays, the same algorithm solves the problem of assigning the set of vertices P of the original graph $D=(X\cup P,E)$ to the set of vertices X. The solution E^k obtained by the second algorithm after reverse reindexation is the solution E^k of the problem of assigning the set of vertices X of the original graph $D=(X\cup P,E)$ to the set of vertices P. After calculating the estimate $F_k(l)$ of the allocation given by the graph $D^k=(X\cup P,E^k)$ the amount of pheromone is calculated proportional to the estimate $F_k(l)$.

3. Co-evolutionary placement algorithm

1. In accordance with the initial data, a complete bipartite solution search graph $D=(X\cup P,E)$ is formed.

2. The number of agents N_a in each of the subpopulations Z_1 and Z_2 is specified.

3. The values of the control parameters are set.

4. The number of iterations is set − N_l.

5. The initial amount of pheromone is deposited on all edges of the initial graph $D=(X\cup P,E)$. The auxiliary matrix $\Delta=||\delta_{iv}||_{n\times n}$ is set to zero.

6. $l=1$. (l is the iteration number).

Start of work of 1 constructive algorithm.

7. $k=1$. (k − agent number).

8. The initial values of the memory parameters for the agent z_k of the population Z_1 are formed. $t=0$. $X_{1k}(1)=\varnothing$. $X_{2k}(1)=X$. $P_{1k}(1)=\varnothing$. $P_{2k}(t)=P$. $E^k(1)=\varnothing$. $\forall v(\rho_v(1))=0$. $\forall i(\rho_i(1))=0$.

9. $t=t+1$. (t is the step number).

10. (<u>*Stage 1*</u>). For each vertex $x_i\in X_{k2}(t)$, a set $U_i(t)$ of incident edges $e_{iv}=(x_i,w_v)\in U_i(t)$ is defined that connect x_i with all vertices p_v of the set $P_{2k}(t)$.

11. For each edge $e_{iv}=(x_i,p_v)\in U_i(t)$, the parameter $q_{iv}(t)$ is determined − the amount of pheromone placed on e_{iv}.

12. The average amount of pheromone per one edge of the set $U_i(t)$ is determined: $\mu_{ik}(t)=\sum_v(q_{iv}(t))/|U_i(t)|$. $\mu_{ik}(t)$ is the cost $x_i\in X_{2k}(t)$.

13. Among the set of vertices $X_{2k}(t)$ with probability $\Psi_{ik}(t)=\mu_{ik}(t)/\sum_i(\mu_{ik}(t))$, a vertex $x^*_i(t)\in X_{2k}(t)$ is chosen.

14. (<u>*Stage 2*</u>). Among the edges $U^*_i(t)$ incident to the selected vertex $x^*_i(t)$, with the probability $\Psi_{vk}(t)=q_{iv}(t)/\sum_v(q_{iv}(t))$, the edge $e^*_{iv}=(x^*_i, p^*_v)$, which is included in the set of edges $E^k(t)$ formed by the agent z_k.

15. Execution of post-procedures after step t.

$E^k(t+1)=E^k(t)\cup e^*_{iv}$.

$X_{1k}(t+1)=X_{1k}\cup x^*_i$. $X_{2k}(t+1)=X_{2k}\backslash x^*_i$. $P_{1k}(t+1)=P_{1k}\cup p^*_v$. $P_{2k}(t+1)=P_{2k}\backslash p^*_{vi}$.

The local degree of the vertices x^*_i and p^*_v takes the value 1, $\rho_i=\rho_v=1$.

16. If $X_{2k}(t)=\varnothing$, then go to 17, otherwise go to 9.

17. Calculation of the estimate $F_k(l)$ of the solution. Calculation of the amount of pheromone proportional to the estimate $F_k(l)$: $\tau_k(l)=Q / F_k(l)$.

18. In the cells of the auxiliary matrix $\Delta=||\delta_{iv}||_{n\times n}$, corresponding to the edges E^k, a pheromone is deposited in an amount proportional to $F_k(l)$: for each edge $e_{iv}\in E^k$, $\delta_{iv}=\delta_{iv}+\tau_k(l)$.

20. If $k<N_a$, then $k=k+1$ and go to 8, otherwise go to 21.

21. Symmetric re-indexing of variables and arrays described above is performed.

<u>*Start of work 2 constructive algorithm*</u>.

22. $k=1$. (k − agent number).

23. The set E^k is formed.

24. The reverse reindexing of the interpretation of the solution E^k obtained by algorithm 2 is performed. In other words: $(\forall e_{iv}\in E^k)[i=v;v=i]$.

25. In the cells of the auxiliary matrix $\Delta=||\delta_{iv}||_{n\times n}$, corresponding to the found set of edges E^k, a pheromone is deposited (added) in an amount proportional to the estimate $F_k(l)$: for $(\forall e_{iv}\in E^k)$ $[\delta_{iv}=\delta_{iv}+\tau_k(l)]$.

26. If $k<N_a$, then $k=k+1$ and go to 22, otherwise go to 27.

27. The deposition of the pheromone accumulated by all agents at the iteration is carried out into the memory matrix $Q=||q_{iv}||_{n\times n}$: $(\forall q_{iv}\in Q)[q_{iv}=q_{iv}+e_{iv}]$.

28. The pheromone evaporation procedure is performed on the edges of the set E of the graph D.

29. If $l<N_l$, then то $l=l+1$ and go to 7, otherwise go to 30.

30. End of the algorithm.

4. Experimental research

Based on the developed placement algorithm, a coevolution placement (CP) program was created. The experiments were carried out by analogy with the well-known method BEKU (Partitioning Examples with Tight Upper Bound of Optimal Solution) [8]. Examples with a known optimum F_{opt} containing up to 1000 vertices were studied. The average dependence of the quality of solutions on the number of iterations (Fig. 2), and the size of the population (Fig. 3) was constructed. The quality is assessed by the value of F_{opt} / F.

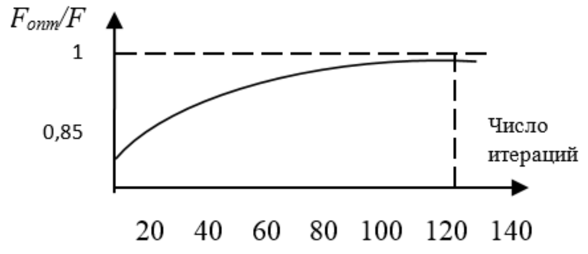

Figure 2: Dependence of the quality of the algorithm solutions on the number of iterations

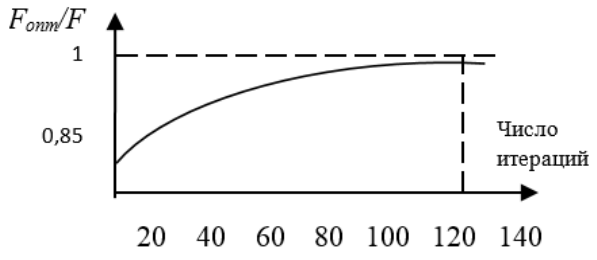

Figure 3: Dependence of the quality of the algorithm solutions on the population size

It was found that for a population size of M=100, the algorithm converges on average at 120 iterations.

To compare the experimental data of placement algorithms, the most famous algorithms *GASP* [7], *ESP* [8] *TimberWolf 4.3* [9-10].

Based on the study of the data, the following dependencies were built (Fig. 4): 1st dependence corresponds to the developed co-evolutionary placement algorithm (CP); 2nd – GASP; 3rd – ESP; 4th – TimberWolf.

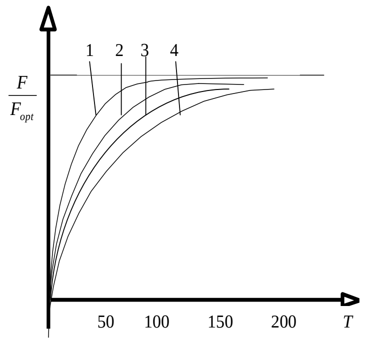

Figure 4: Comparison of algorithms

The developed (1) and compared (2,3,4) algorithms allow obtaining solutions close to optimal. However, the developed algorithm has a high performance – fewer generations (iterations) are required.

Testing was carried out on benchmarks 19s, PrimGA1, PrimGA2. The results compared to existing algorithms are improved by 6-7%. The probability of obtaining a global optimum was 0,96. On average, the solutions differ from the optimal one by less than 2%. The time complexity of the algorithm for fixed values of M and T lies within $O(n)$.

5. Conclusion

To improve efficiency, enhance the convergence of the algorithm and the ability to exit from local optima, a co-evolutionary approach to the construction of a placement algorithm is proposed. To solve the placement problem, the authors have developed a modified metaheuristic by analogy with the models of the adaptive behavior of the ant algorithm. Simultaneously, in the search space of the optimization problem, two subpopulations evolve, each of which solves the same initial optimization problem, but having different search areas and optimization strategies. The co-evolutionary approach provides a broader view of the solution space and a higher probability of localizing the global extremum of the problem. Distinctive features of the interaction between subpopulations are that they are based on the use of a common graph for finding solutions, common evolutionary memory, and the formation of a single interpretation of a solution in the form of a bipartite graph.

6. Acknowledgements

This research is supported by grants of the Russian Foundation for Basic Research of the Russian Federation, the project №20-07-00260.

7. References

[1] A.V. Tuchin, E.N. Bormontov, K.G. Ponomarev, Introduction to computer-aided design systems for integrated circuits, Voronezh State University Publishing House (2017) 110 p.

[2] G.G. Kazennov, Basics of designing integrated circuits and systems. Binom, Knowledge Laboratory, Moscow, (2010) 295 p.

[3] K. Rose, D. Radchenko, Synopsys platform for designing digital systems − a new level of SoC design technologies, ELECTRONICS, (2018).

[4] B.K. Lebedev, O.B. Lebedev Hybrid bioinspired algorithm for the formation of lines of standard cells in the design of VLSI topology, Problems of the development of promising micro- and nanoelectronic systems, IPPM RAS, Moscow, (2018) pp. 180-187.

[5] A.P. Karpenko, Modern search engine optimization algorithms. Algorithms Inspired by Nature: A Study Guide. 2nd edition, Moscow, (2016) 448 p.

[6] B.K. Lebedev, O.B. Lebedev, A.A. Zhiglatiy, Placement of VLSI elements based on swarm intelligence models, Problems of the development of promising micro- and nanoelectronic systems, IPPM RAS, Moscow, (2020) pp. 118-126.

[7] S. Gavrilov, D. Zheleznikov, R. Chochaev, Simulated Annealing Based Placement Optimization for Reconfigurable Systems-on-Chip, 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, Moscow, (2019) pp. 1597-1600.

[8] V.V. Kureichik, E.V. Kuliev, Integrated algorithm for elements placement on the printed circuit board, IOP Conference materials science and engineering, Institute of Physics and IOP Publishing Limited, (2020) pp. 121-146.

[9] V.V. Kureichik, S.I. Rodzin, State, problems and prospects for the development of bioheuristics, Software systems and computational methods, (2016) pp. 158-172.

[10] B.K. Lebedev, O.B. Lebedev, E.M. Lebedeva, A.A. Zhiglaty Hybrid Optimization Method Based on the Integration of Evolution Models and Swarm Intelligence in Affine Search Spaces, Artificial Intelligence Methods in Intelligent Algorithms, Springer, Czech Republic, (2019) pp. 32-39.