Wolf pack algorithm for solving VLSI design tasks

E V Kuliev, D Yu Zaporozhets, V V Kureichik, I O Kursitys

Southern federal university, 105/42 Bolshaya Sadovaya Str., Rostov-on-Don, 344006, Russia.

E-mail: elmar_2005@mail.ru

Abstract. The paper deals with the development of new and modified heuristic mechanisms of searching optimal solutions, which is considered as one of the main problems of artificial intelligence. One of the promising areas of artificial intelligence development is application of methods and models of biological systems behavior for solving NP-complete and NP-difficult optimization tasks. The paper presents the statement of the placement task in designing very large-scale integration circuits (VLSI). The authors propose the algorithm for solving this task on the basis of biological system behavior in nature, e.g. wolf pack. Wolves are considered as typical social animals having clear separation of social work. The paper describes the actions and rules of wolf pack behaviour in nature. Based on the wolves’ behaviour rules and actions, the authors present the modified wolf pack algorithm. The benefits of the developed modified algorithm include the ability to improve each following iteration of the placement task. The wolf pack algorithm is implemented as a computer software on Java. To estimate the effectiveness of the proposed approach, the authors use the well-known IBM benchmarks to compare with the developed algorithm. The comparison is implemented with the results of the following algorithms: Capo 8.6, Feng Shui 2.0, Dragon 2.23. The results show that the wolf pack algorithm is more effective than the analogues.

1. Introduction.

To date, the scientific and technical progress highly depends on computer-aided design (CAD) systems. Building the intelligent CAD systems is related to the development of mathematical support, which includes mathematical models, methods and algorithms of intelligent design. The main part of the CAD systems are VLSI design systems, where the input data are presented as electrical diagrams, and the output data are presented as circuit topology. A VLSI design stage defines the effectiveness of the created circuit and have a lot of issues due to the very large scope of the processed data and information [1]. In terms of the modern information technologies there is a problem of following newest design rules. The available methods and algorithms of design are not effective for solving the mentioned task. In that connection, the complexity of solving the design problems has significantly increased these days. Thus, the VLSI developers need to use the effective program systems to implement the complex advanced effective algorithms [2].

Recently, scientific field devoted to modeling biological systems behavior in nature has been actively developed. The models of biological systems behavior describe mathematical methods supporting the main principles of the natural mechanisms of decision making [3,4]. In that regard, it is necessary to develop new VLSI design algorithms based on the modeling of biological systems behavior.
2. Problem statement
Input data for solving the VLSI elements placement task include the following [1, 5]:
- switch; 
- switching diagram, containing the desired amount of blocks 
- netlist.

The model of the considered switching diagram is composed of single-type cells united in a line comprising a grid. A typical cell is thought of as a geometrical unit of a rectangle form with the internal topology and pre-defined places of external connections. In this case, pins can be placed from any side.

Let us present the placement task as placing a set of elements on the switching board in such way that the system performance is optimal. This is provided by minimizing the total length of connections [6].

Thus, formal description of the VLSI elements placement task can be presented as follows. Let \( B_1, \ldots, B_n \) be the VLSI elements placed on the switching board. Each element \( B_i \) \( 1 \leq i \leq N \) has its own geometrical dimensions: \( h_i \) denotes height and \( w_i \) denotes width. Let \( N = \{|R_i| \mid i = 1, m\} \) denote the netlist and \( L_i \) be the net length, \( N \mid i = 1, m \). The task is to find rectangle areas for each element on the switching board. The rectangle places are represented by \( R = \{R_i \mid i = 1, n\} \) in such way that [4-6]:
1. Each block is placed in the corresponding rectangle area so that \( R_i \) has the height \( h_i \) and the width \( w_i \).
2. Areas from the set \( R \) cannot intersect.
3. Common are of the rectangle defined by the set \( R \) tend to be minimal.
4. Total length of connections is minimized so that:

\[
F = \sum_{i=1}^{m} L_i \quad F \to \min. \tag{1}
\]

The paper proposes using the modified algorithm for finding the length \( L_i \) of the net \( i \).

To solve the VLSI placement task effectively, we propose using the modified wolf pack algorithm. Let us denote the artificial wolf’s space as \( N \times D \), where \( N \) denotes the amount of wolves and \( D \) denotes the amount of variables. Vector \( X_i = (x_i^1, x_i^2, \ldots, x_i^D) \) represents the position of the wolf \( i \), where \( x_i^d \) is a variable \( d \) belongs to the unit \( X_i \). Vector \( Y = f(X) \) is an objective value of the function \( X \), which can be considered as concentration of the prey scent smelled by the artificial wolves.

3. Bioinspired algorithm based on the wolf pack behavior
In terms of the optimization problems the wolves can be considered as typical social animals, and its social work is strictly distinguished. The wolf pack contains three types of the wolves: wild wolves, scout wolves and a leader wolf. Wolves’ behavior is characterized by three intelligent actions: encircling, hunting and scouting. Additionally, in a wolf pack there are two rules, that can be represented as: “the wines takes all” for the leader and “the strongest survives” for the rest of the pack. The more detailed description of the wolves’ behavior is presented in the following way.

1) “The winner takes all” for the leader [7-12]
   The leader is the wolf with the best fitness function value. Thus, the leader position can be considered as the prey position. During each iteration the value of the leader’s function is compared with the values of other wolves’ functions. If the \( Y_{lead} \) value is not the best, then the leader is replaced with the best wolf. The leader does not perform the intelligent actions of scouting, hunting and encircling. It will move to the next iteration before another best wolf replaces him.

2) Scouting [7-12]
   Besides the leader wolf, there are several elite wolves considered as scouts. Scout wolves has better fitness function value in comparison with wild wolves. \( Y_i \) and \( Y_{lead} \) are fitness function values of scout and the leader wolves correspondently. If \( Y_i > Y_{lead} \), then the leader wolf is not the best at this iteration, therefore, it is replaced by the scout wolf \( (Y_i = Y_{lead}) \). If \( Y_i < Y_{lead} \), scout wolves behave as follows:
Firstly, the scout wolf \( i \) moves to \( h \) by several ways and remember the values of the objective function of each direction. After the move to the direction \( i \), the position of the scout \( i \) is calculated as follows [7]:

\[
x_{i,d}^p = x_{i,d} + \sin \left( 2\pi \frac{p}{n} \right) \cdot \text{step}_a.
\]

The next step is to choose the direction providing the best value of the objective function and to update the value of \( X_i \). After that, the scout’s behavior is repeated until we meet \( Y_i > Y_{lead} \) or the maximum number of the iterations \( T_{\text{max}} \). It should be noted, that the valued of \( h \) is different for each wolf and is represented by integer from \([h_{\text{min}}, h_{\text{max}}]\). It is focused on enhancing the different hunting strategies for different wolves. The step length is denoted by the parameter \( \text{step}_a \).

3) Hunting (chasing) [7-12]:
According to this rule, the leader howls and calls the wild wolves to encircle the prey. The leader position is considered as the possible position of the prey at this step, and the wild wolves gather around the leader. The parameter \( \text{step}_b \) represents the step lengths; \( g_d^k \) denotes the leader’s position at the iteration \( k \), \( k \) denotes the number of iterations. The wild wolf’s position is updated according to the following equation:

\[
x_{i,d}^{k+1} = x_{i,d}^k + \text{step}_b \cdot \frac{g_{d}^k - x_{i,d}^k}{|g_{d}^k - x_{i,d}^k|}.
\]

If \( Y_i > Y_{lead} \), then the wild wolf becomes the leader and \( Y_i = Y_{lead} \). In fact, the leader performs the hunting work. If \( Y_i < Y_{lead} \), then the wild wolves continue gathering around the leader at a high speed before we meet \( L(i, lead) < L_{\text{near}} \). After that, the wolf performs the precipitating action. \( L(i, lead) \) presents the distance between the leader and the wolf \( i \); \( L_{\text{near}} \) is the encircling distance, which can be considered as the criterion to define if the wolf changes its behavior from chasing to encircling. \( L_{\text{near}} \) is calculated as follows [9]:

\[
L_{\text{near}} = \frac{1}{D \cdot w} \cdot \sum_{d=1}^{D} |\max_d - \min_d|
\]

where \( w \) denotes the distance definition coefficient, \([\min_d, \max_d]\) denotes the interval of the values of the variable \( d \).

4) Encircling the prey
After the big-step moving towards the leader, the wild wolves are close to the prey and change their behavior from chasing to encircling the prey. The position of the wolf \( i \) is updated according the following equation:

\[
x_{i,d}^{k+1} = x_{i,d}^k + \lambda \cdot \text{step}_c \cdot |g_{d}^k - x_{i,d}^k|
\]

where \( \lambda \) denotes the random valued from the interval \([-1,1]\); \( \text{step}_c \) denotes the step length in terms of encircling. If \( Y_i^{\text{new}} > Y_i^{\text{old}} \) after the encircling started, then the position \( X_i \) is updated. Otherwise, the position \( X_i \) is not changed. Relationship between \( \text{step}_a \), \( \text{step}_b \), and \( \text{step}_c \) is implemented in the following way:

\[
\text{step}_a = \frac{\text{step}_b}{2} = 2 \cdot \text{step}_c = S
\]

where \( S \) is a step coefficient defining the degree of the artificial wolves hunting in the space.
5) Updating rule "the strongest survives"

The rule implements the natural selection by Charles Darwin. As the prey is divided between the strong and the weak wolves, some of the weak wolves can die. The developed algorithm regenerates the wolves by a random way while reducing the $R$ weak wolves, which are not fitted enough. $R$ represents the integer randomly chosen from the interval $\left[\frac{N}{2\beta}, \frac{N}{\beta}\right]$ where $\beta$ denotes the proportional coefficient of updating the population.

Let us present the developed modified wolf pack algorithm as the following steps:
- initialize the population of grey wolves $X_i$ ($i = 1, 2, ..., n$);
- initialize the parameters $a, A, and C$;
- estimate the fitness of each search agent;
- initialize the first best solution as $X_\alpha$, the second best solution as $X_\beta$, the third best solution as $X_\delta$;
- set the maximum number of iterations in the initial data;
- increment the count $t$;
- update the search agents positions;
- update the parameters $a, A, and C$;
- estimate the fitness of all search agents;
- update the best agents $X_\alpha, X_\beta, X_\delta$;
- finish the algorithm and to visualize the solution obtained by the first best agent $X_\alpha$ at the moment.

The benefits of the developed algorithm include the ability to improve each following stage of solution of the VLSI placement task. The stage of solution is considered as the alternatives of the placement task solution previously developed and used in terms of the software system meeting all required criteria.

![Figure 1](image1.png)
**Figure 1.** Diagram of quality of the solutions obtained by different algorithms.

![Figure 2](image2.png)
**Figure 2.** Diagram of the objective function increment depending of the used analogue.

4. Experimental research.

In terms of this paper the developed wolf pack algorithm is implemented as software on Java. To carry out the experiments, the authors used the well-known benchmarks of IBM [13]. The purpose of research is to investigate the quality of the solution obtained by the developed wolf pack algorithm. The algorithm is compared with such popular algorithms as Capo 8.6 [13], Feng Shui 2.0 [13], Dragon 2.23 [14-15].
The experiments are performed on the basis of the Intel i7 4 cores, 2.7 GHz processor. The results are presented in Table 1 and Figures 1,2.

### Table 1. Comparison of the algorithms

| Testing circuit | Name of circuit | Number of elements | Length, m (Capo 8.6) | Length, m (Feng Shui 2.0) | Length, m (Dragon 2.23) | Length increment (Capo 8.6), m | % Length increment (Capo 8.6) | Length increment (Feng Shui 2.0), m | % Length increment (Feng Shui 2.0) | Length increment (Dragon 2.23), m | % Length increment (Dragon 2.23) |
|-----------------|-----------------|--------------------|----------------------|--------------------------|------------------------|--------------------------------|-----------------------------|--------------------------------|-----------------------------|-------------------------------|-------------------------------|
| ibm01           | ibm01           | 12752              | 4,97                 | 4,87                     | 4,42                   | -0.82                          | -19.76%                     | -0.72                          | -17.35%                     | -0.27                          | -6.51%                        |
| ibm02           | ibm02           | 19601              | 15,23                | 14,38                    | 13,57                  | 12,4                          | 2,83                         | -22,82%                      | 1,98                         | -15,97%                      | 1,17                          | -9,44%                        |
| ibm03           | ibm03           | 23136              | 14,06                | 12,84                    | 12,33                  | 11,44                          | 2,62                         | -22,90%                      | 1,4                          | -12,24%                      | -0.89                         | -7,78%                        |
| ibm04           | ibm04           | 27507              | 18,13                | 16,69                    | 15,41                  | 15,5                          | -2,63                        | -16,97%                      | -1,19                        | -7,68%                        | 0,09                          | 0,58%                         |
| ibm05           | ibm05           | 29347              | 44,73                | 37,3                     | 36,38                  | 35,66                          | -9,07                        | -25,43%                      | -1,64                        | -4,60%                        | -0,72                         | -2,02%                        |
| ibm06           | ibm06           | 32498              | 21,96                | 20,27                    | 20,38                  | 18,5                          | -3,46                        | -18,70%                      | -1,77                        | -9,57%                        | -1,88                         | -10,16%                       |
| ibm07           | ibm07           | 45926              | 36,06                | 31,5                     | 29,97                  | 31,5                          | -4,56                        | -14,48%                      | 0                           | 0,00%                         | 1,53                          | 4,86%                         |
| ibm08           | ibm08           | 51309              | 37,89                | 34,14                    | 32,2                   | 29,01                          | -8,88                        | -30,61%                      | -5,13                        | -17,68%                      | -3,19                         | -11,00%                       |
| ibm09           | ibm09           | 53395              | 30,28                | 29,86                    | 28,1                   | 25,41                          | -4,87                        | -19,17%                      | -4,45                        | -17,51%                      | -2,69                         | -10,59%                       |
| ibm10           | ibm10           | 69429              | 61,25                | 57,99                    | 57,2                   | 53,1                          | -8,15                        | -15,35%                      | -4,89                        | -9,21%                        | -4,1                          | -7,72%                        |
| ibm11           | ibm11           | 70558              | 46,45                | 43,28                    | 40,77                  | 38,6                          | -7,85                        | -20,34%                      | -4,68                        | -12,12%                      | -2,17                         | -5,62%                        |
| ibm12           | ibm12           | 71076              | 81,55                | 75,91                    | 71,03                  | 70,41                          | -11,14                       | -15,82%                      | -5,5                         | -7,81%                        | -0,62                         | -0,88%                        |
| ibm13           | ibm13           | 84199              | 56,47                | 54,09                    | 50,57                  | 45,6                          | -10,87                       | -23,84%                      | -8,49                        | -18,62%                      | -4,97                         | -10,90%                       |

5. Conclusion.

The paper presents the common statement of the VLSI placement task, the main actions and rules of the wolf pack behavior in the nature. To provide the effective solutions of the VLSI placement task, the authors developed the modified wolf pack algorithm allowing us to reduce the problem dimension and obtain quazi-optimal solutions in polynomial time. The wolf pack algorithm is implemented as software on Java. The results are presented in the paper. The well-known benchmarks of IBM are used as testing circuits. The paper demonstrates the results of comparison of the developed algorithm and the popular algorithms such as Capo 8.6, Feng Shui 2.0, Dragon 2.23. According to the research, the developed wolf pack algorithm is more effective than its analogues.

Acknowledgments

This work was funded by grant No. MK-1480.2018.9 from the President of the Russian Federation for support of young Russian scientists - candidates of sciences.
References

[1] Alpert C, Mehta D, Sapatnekar S 2009 Handbook of Algorithms for Physical Design Automation (New York, USA: Auerbach Publications) p 1024
[2] Sherwani N A 2013 Algorithms for VLSI Physical Design Automation (Springer; 3rd edition) p 572
[3] Kureichik V Jr, Kureichik V, Bova V 2016 Placement of VLSI fragments based on a multilayered approach Advances in Intelligent Systems and Computing 464 181-190
[4] Kacprzyk J, Kureichik V M, Malioukov S P, Kureichik V V, Malioukov A S 2009 Experimental investigation of algorithms developed Studies in Computational Intelligence 212 211-223, 227-236
[5] Bunglowala A, Singhi B M, Verma A 2009 Optimization of Hybrid and Local Search Algorithms for Standard Cell Placement in VLSI Design International Conference on Advances in Recent Technologies in Communication and Computing (Artcom) 2009 826-828
[6] Zaruba D, Zaporozhets D, Kureichik V 2016 VLSI placement problem based on ant colony optimization algorithm Advances in Intelligent Systems and Computing 464 127-133
[7] Madadi A, Motlagh M 2014 Optimal Control of DC motor using Grey Wolf Optimizer Algorithm Technical Journal of Engineering and Applied Science 2014-4-04 373-379
[8] Rezaei H, Bozorg-Haddad O, Chu X 2018 Grey wolf optimization (GWO) algorithm Studies in Computational Intelligence 720 81-91
[9] Long W, Jiao J, Liang X, Tang M 2018 Inspired grey wolf optimizer for solving large-scale function optimization problems Applied Mathematical Modelling 60 112-126.
[10] El Gayyar M, Emary E, Sweilam N H, Abdelazeem M 2018 A Hybrid Grey Wolf-Bat Algorithm for Global Optimization Advances in Intelligent Systems and Computing 723 3-12
[11] Gupta S, Deep K 2019 A novel Random Walk Grey Wolf Optimizer Swarm and Evolutionary Computation Swarm and Evolutionary Computation 44 101-112
[12] Kureichik V V, Zaruba D V 2015 The bioinspired algorithm of electronic computing equipment schemes elements placement Advances in Intelligent Systems and Computing 347 51-58
[13] Kuliev E V, Kureichik V V, Kursitys I O 2018 Decision Making in VLSI components Placement Problem Based on Grey Wolf Optimization Proc. of 2018 XXI Int. Conf. on Soft Computing and Measurements (SCM) 672-675
[14] IBM-PLACE 2.0 benchmark suits http://er.cs.ucla.edu/benchmarks/ibm-place2/bookshelf/ibm-place2-all-bookshelf-nopad.tar.gz
[15] Wang M, Yang X, Sarrafzadeh M 2000 Standard-cell Placement Tool for Large Industry Circuits ICCAD2000 260-263
[16] Kacprzyk J, Kureichik V M, Malioukov S P, Kureichik V V, Malioukov A S 2009 General questions of automated design and engineering Studies in Computational Intelligence 212 1-22