Parametric Optimization Based on Bacterial Foraging Optimization

Daria Zaruba(✉), Dmitry Zaporozhets, and Elmar Kuliev

Southern Federal University, Rostov-on-Don, Russia
daria.zaruba@gmail.com, elpilasgsm@gmail.com,
elmar_2005@mail.ru

Abstract. Today parametric optimization problems are widely used in different science and technology domains. These problems are weather forecasting, calculation of electromotor parameters as well as VLSI design problems which belong to NP-problems and do not have deterministic algorithms to solve it. So, it is necessary to develop up-and-coming heuristic methods to obtain quazi-optimal solutions during polynomial time. The paper deals with parametric optimization of technical objects. From the mathematical point of view the parametric optimization problem reduces to the global constrained continuous optimization. The formulation of parametric optimization problem is made. To solve this problem it is developed a stochastic algorithm based on foraging behavior of E.coli bacteria. A bacterial colony is considered as multiagent system in which each agent operates in autonomy according with quite elementary rules. Colony behavior is based on self-organization to reach common goals by low-level interconnection. The colony does not have centralized control. Conjunction of simple agents creates a behavioral strategy without any global control. To analyze the developed algorithm there were carried out a set of experiments, which confirm theoretical estimations and calculate optimal values of algorithm parameters. Experimental results show perspective of this approach.

Keywords: Parametric optimization · Swarm intelligence · Bioinspired search · Bacterial colony

1 Introduction

Nowadays, in terms of parametric optimization problems probabilistic methods based on simulation of creatures' behavior in nature are most often used. An experience has shown that multi-agent methods are efficient tool for optimization in various scientific and technical fields. To design high-performed mechanisms for specific optimization problems it is necessary to developed promising methods inspired by natural systems. There are algorithms based on evolutionary computation, ants, bacterial and particle swarm optimization methods [1–5].

In the paper authors suggest a probabilistic algorithm on the basis of multi-agent intelligence conception that is simulation of E.coli bacteria movement. The key idea is that during the movement in environment the bacterium strives towards nutritious areas and avoids harmful ones. In semisolid useful environment bacteria can create stable

R. Silhavy et al. (eds.), *Artificial Intelligence Trends in Intelligent Systems*,
Advances in Intelligent Systems and Computing 573, DOI 10.1007/978-3-319-57261-1_6

spatiotemporal structures. Under specific conditions E.coli can group and show intelligence cooperative behavior.

To estimate the algorithm's quality and compare obtained results with another swarm intelligence algorithms there were conducted a computational experiment. On the basis of these researches the authors can confirm that the time complexity of the developed algorithm has polynomial complexity.

2 Formulation of Parametric Optimization Problem

During optimal design a mathematical model of technical object is formalized description of quality criterion which execute function, requirements, etc. [6, 7].

The parametric optimization problem is solved by finding such input circuit parameters that output parameters have specified characteristic but circuit elements and way of its connection remain the same.

Let n are control parameters in the vector X = (x1, x2, …, xn). Let F(X) is an objective function (OF), XO is its definition area. The vector X defines coordinates of point within definition area XO. If elements in X have only discrete values, then XO is a discrete set and the optimization problem belongs to discrete programming field.

The aim of parametric optimization algorithms is define such control parameter vector that the specified objective function posses optimal value.

During the mathematical model development it is necessary to calculate object parameters affecting on an optimal criterion. Then, there are defined parametric, discrete and functional restrictions of the technical object [5, 6].

Parametric restrictions are represented as follows:

$$x_i' \leq x_i'' \tag{1}$$

Where x_i is i-th parameter of the object; $x_i^{'}$ and $x_i^{'}$ are minimum and maximum values of the i-th parameter correspondingly.

Discrete restrictions are written as:

$$x_j = \left\{x_{j1}, x_{j2}, \ldots, x_{jm}\right\} \tag{2}$$

Where x_j is j-th parameter of the object; x_{jk} is acceptable value of j-th parameter (k = 1,2,…,m). Such restrictions are imposed on parameters value in connection with its physical entity.

Functional restrictions are constraining conditions of its values and are represented as follows:

$$g_i(x) \leq 0; g_j(x) = 0; g_k(x) < 0. \tag{3}$$

Functional restrictions involve strength, rigidity and stability conditions which provide desirable values of technical characteristics [7–10].

3 Biological Foundations of Bacterial Colony Behavior

The E.coli bacterium (Fig. 1) is the most studied. It contains a cell plasma membrane, a cell wall and a capsule with cytoplasm or nucleoid. Bacterium size is about 1 micron in diameter and 2 microns in length, weight is about 1 pg. Under certain conditions the E.coli is increased in size and divided into two descendant bacteria. Obtaining necessary nutrition and maintaining appropriate temperature (about 37 °C), reproduction takes about 20 min, hence, a bacteria population growths exponentially [7–9].

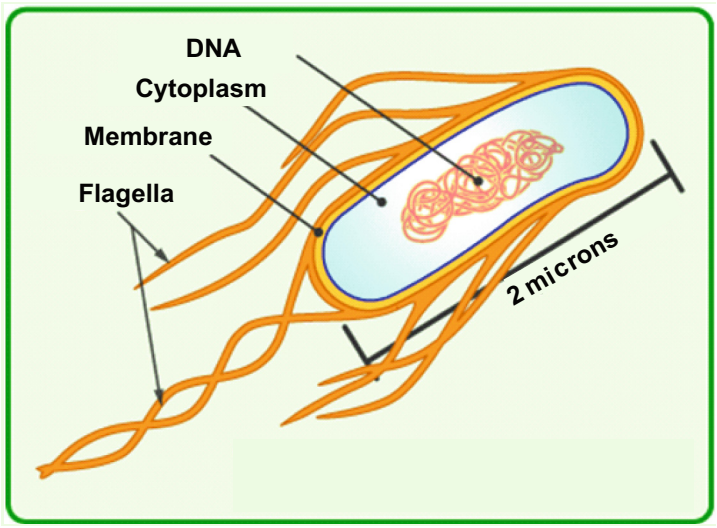

Fig. 1. The E.coli bacterium

Locomotion is achieved via a set of relatively rigid flagella that enable it to "swim" via each of them rotating in the same direction at about 100–200 revolutions per second. Each flagellum is a left-handed helix configured so that as the base of the flagellum (i.e., where it is connected to the cell) rotates counterclockwise, as viewed from the free end of the flagellum looking towards the cell, it produces a force against the bacterium so it pushes the cell. If a flagellum rotates clockwise, then it will pull at the cell. From an engineering perspective, the rotating shaft at the base of the flagellum is quite an interesting contraption that seems to use what biologists call a "universal joint" (so the rigid flagellum can "point" in different directions, relative to the cell). In addition, the mechanism that creates the rotational forces to spin the flagellum in either direction is described by biologists as being a biological "motor" (a relatively rare contraption in biology even though several types of bacteria use it). The motor is quite efficient in that it rotates a complete revolution using only about 1000 protons and thereby E.coli spends less than 1% of its energy budget for motility [10, 11].

An E.coli bacterium can move in two different ways: it can "run" (swim for a period of time) or it can "tumble," and it alternates between these two modes of operation its entire lifetime (i.e., it is rare that the flagella will stop rotating).

The motion patterns that the bacteria will generate in the presence of chemical attractants and repellents are called "chemotaxis." For E.coli, encounters with serine or aspartate result in attractant responses, while repellent responses result from the metal ions Ni and Co, changes in pH, amino acids like leucine, and organic acids like acetate. What is the resulting emergent pattern of behavior for a whole group of E.coli bacteria? Generally, as a group they will try to find food and avoid harmful phenomena, and when viewed under a microscope, you will get a sense that a type of intelligent behavior has emerged, since they will seem to intentionally move as a group.

4 Model of Bacterial Foraging Optimization Algorithm

In terms of search optimization problems a chemotaxis can be viewed as an heuristic optimization mechanism using all resources to find new areas with a great number of useful resources [12, 13].

A classical bacterial foraging optimization (BFO) algorithm is based on three mechanisms: chemotaxis, reproduction and dispersal.

Let $X_{i,r,l} - (|X| \times l)$ is a current position of a bacterium $s_i \in S$ at iteration t, step of reproduction r, step of elimination l. There are $i \in [1 : |S|]$, $t \in [1 : \hat{t}]$, $l \in \left[1 : \widehat{t^l}\right]$, where $|S|$ is an even number of agents in colony S, $\hat{t}$, $\widehat{t^r}$, $\widehat{t^l}$, are total number of iteration (steps of chemotaxis), steps of reproduction and elimination. Besides that, corresponding value of objective function is denoted as $\varphi_{i,r,l}$.

Chemotaxis. In terms of bacterial foraging optimization algorithm a local optimization is realized as chemotaxis procedure. The next position $X'_{i,r,l}$ of the bacterium s_i is calculated as

$$X'_{i,r,l} = X_{i,r,l} + \lambda_i \frac{V_i}{\|V_i\|_E} \tag{4}$$

Where V_i is a vector of chemotaxis step for the bacterium s_i; λ_i is a current value of step. During the swimming vector V_i remains constant at the next iteration, i.e. $V'_i = V_i$. When bacterium is tumble, vector V'_i is a random vector in the interval $[-1;1]$.

Reproduction. The main aim of the reproduction mechanism is increasing of convergence rate of the algorithm by means of search space narrowing. Let h_i is a "health" of bacterium s_i. Then, for all trajectory points the total value of objective function is calculated as

$$h_i = \sum\nolimits_{\tau=1}^{t} \varphi_{i,r,l}(\tau). \tag{5}$$

Further, it is denoted h_i, $i \in [1 : |S|]$ and bacteria are sorted in the decreasing order of its health. As a result we obtain a linear list.

Elimination. To find a global optimum of the objective function there are not enough chemotaxis and reproduction mechanisms since they are not solve a preliminary

convergence problem. To overcome the weakness of the method it is used an elimination procedure.

This procedure initializes after reproduction and consists in the following. According with predetermined probability ξ_e it is selected n bacteria s_{i_1}, s_{i_2}, …, s_{i_n} in a random way and delete it from population. Instead of these in random point of search space there are generated new bacteria (agents) with the same number. Should be noted, after elimination procedure a number of agents in the population remains constant [14].

5 Description of the Algorithm

A block diagram of the developed algorithm for the parametric optimization problem is shown on Fig. 2. To illustrate key features of the BFO algorithm let us consider it in the context of a VLSI placement problem.

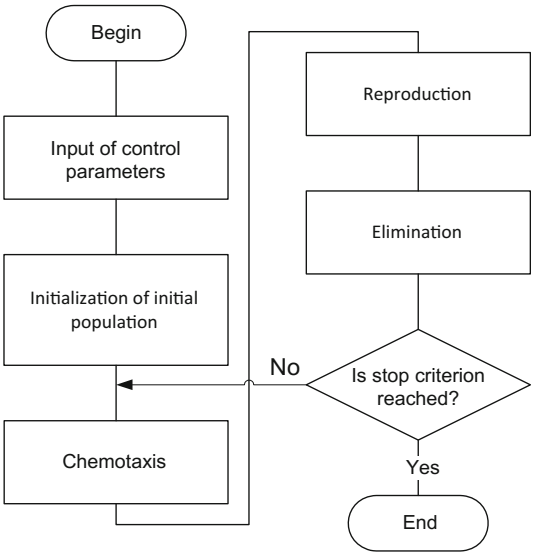

Fig. 2. The block diagram of the BFO algorithm

On the basis of the heuristic described above, the authors distinguish the following steps: input of control parameters, initialization of initial population, chemotaxis, reproduction, elimination.

The VLSI algorithm based on the BFO works with the following control parameters: a set of alternative solutions in the population N_S, a number of generations (iteration) T, initial value of "health" for each alternative solution H_0, a number of chemotaxis step N_x.

During initialization stage it is generated a set of alternative solutions for each of which OF values are calculated. A set of solutions can be obtained in previous stages,

for example, as a result of a hierarchical algorithm or generates by the random way. Also, at this stage for each alternative solution it is necessary to define an initial index of gene (element of alternative solution) that began the agent's movement. The authors suggest to distinguish two direction: increasing and decreasing of the gene's index. In initialization stage an agent's direction is random.

Also, the authors introduce the step of agent that means a number of genes the same type between initial and finite indexes. In terms of the developed algorithm there are used two types of genes: operands (encoded by positive integer) and operates (encoded by negative integer). So, a gene, situated in gene-operand at initial step, can move only in gene-operands, and a gene, situated in gene-operator at initial step, can move only in gene-operators. When the first of the last gene is reached, the calculation of finite gene's index is conducted repeatedly. The example of agent movement with the initial index 3 in gene-operators with step 4 is shown on Fig. 3.

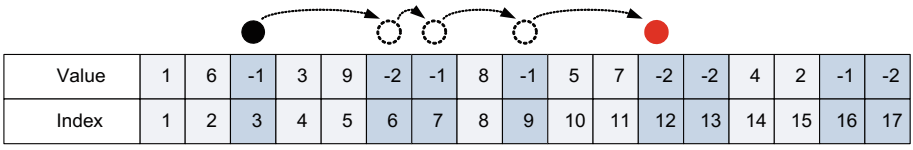

Fig. 3. Agent movement in gene-operators

Example of agent movement with the initial index 5 in gene-operands with step 6 is shown on Fig. 4.

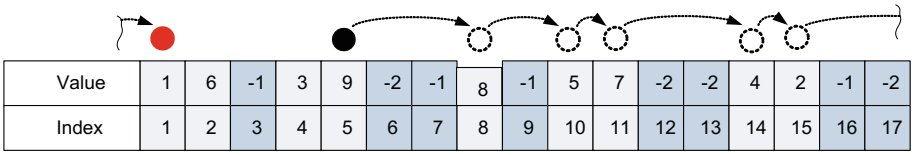

Fig. 4. Agent movement in gene-operands

A result of agent movement is a pair permutation of genes with initial and finite index. After that, an increment of the OF value before and after permutation are calculated. The value of increment is summed with a current value of agent's "health". Here, the "health" is total increment of the OF value during all steps of chemotaxis. Obviously, that increment can be positive and negative. If after regular step of chemotaxis the value of "health" is less or equal to 0, then movement is stopped for current alternative solution. Note, an agent changes its direction at the next step if increment will be negative.

When chemotaxis is stopped for each alternative solution, there is a reproduction is initialized. Reproduction contains the following steps:

- All sets of alternative solutions (populations) are sorted in decreasing of OF values.
- The first half of solutions is doubled. Alternative solutions from the first half of the sorted list are copied and migrate to the current population.

- For copies the initial index of agent is changed by a random way and the "health" value is equal to H_0.

Next, there is elimination for saving constant quantity of alternative solutions in the population N_S. Elimination contains the following steps:

- Population is sorted in increment of the value of "health".
- All alternative solutions with "health" less or equal to 0 are removed from the population.
- If current size of population $N_S^{cur} < N_S$, then there are generated additional quantity of solutions $N_S - N_S^{cur}$.
- If current size of population $N_S^{cur} > N_S$, then alternative solutions with the lowest value of "health" are deleted until N_S^{cur} will be equal to N_S^{cur}.

The elimination stage is last step of iteration in terms of VLSI fragments placement on the basis of BFO. After that optimization continues iteratively until a stop criterion will be reached. Here, a stop criterion is a number of iteration in the algorithm.

6 Experiments

The authors developed software in the Borland C++ Builder™ 6.0. Testing of the developed algorithm was carried out on AMD FX(tm)-8121 Eight-Core Processor 3.10 GHz, RAM 4,00 Gb.

To conduct computational experiments there was developed software for VLSI fragments placement.

The obtained results allow to define a dependence of algorithm execution time on input parameters (Fig. 5).

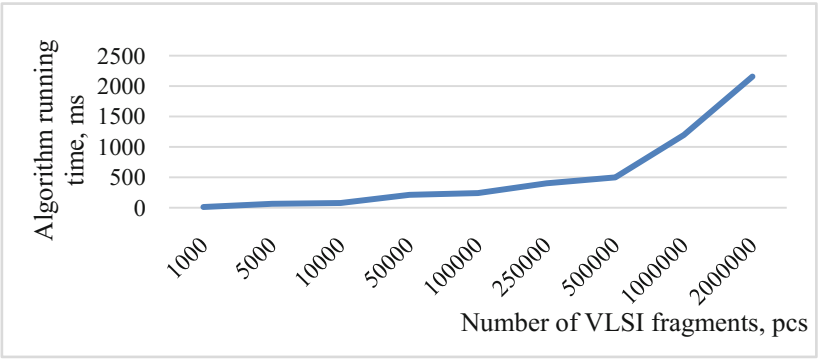

Fig. 5. Dependence of algorithm execution time on input parameters

The algorithm time complexity is represented as $O(n^2)$, where n is a number of input data.

Also, it was considered a dependence of algorithm execution time on a number of iteration (Fig. 6).

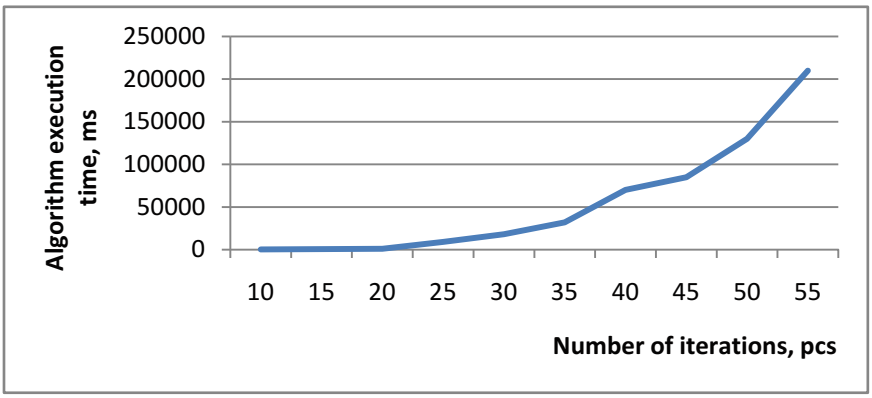

Fig. 6. Dependence of algorithm execution time on a number of iteration

The time complexity of this dependence is represented as $O(n^4)$, where n is a number of iterations.

To estimate a quality of obtained solutions there were compared the BFO algorithm with well known VLSI fragment placement algorithms such as ant colony optimization and genetic algorithms (ACO and GA). The results of comparison are shown on Table 1 and Fig. 7.

Table 1. The results of comparison

Sr. No	A number of elements	Ser. size	ACO	GA	BFO
			Average value of OF, m	Average value of OF, m	Average value of OF, m
1	500	20	0,000125	0,000321	0,00013
2	1000	20	0,00412	0,005121	0,004112
3	2000	20	0,009124	0,014101	0,009141
4	3000	20	0,018121	0,021231	0,016121
5	5000	20	0,04101	0,051211	0,040012
6	8000	20	0,100231	0,105121	0,101643
7	10000	20	0,191231	0,2412	0,195512
8	15000	20	0,248	0,312	0,247
9	50000	20	1,36	1,98	1,34
10	100000	20	3,2	3,9	3,01
11	500000	20	18,6	19,6	18,33
12	750000	20	22,32	24,12	22,1
13	1000000	20	29,98	31,06	29,23

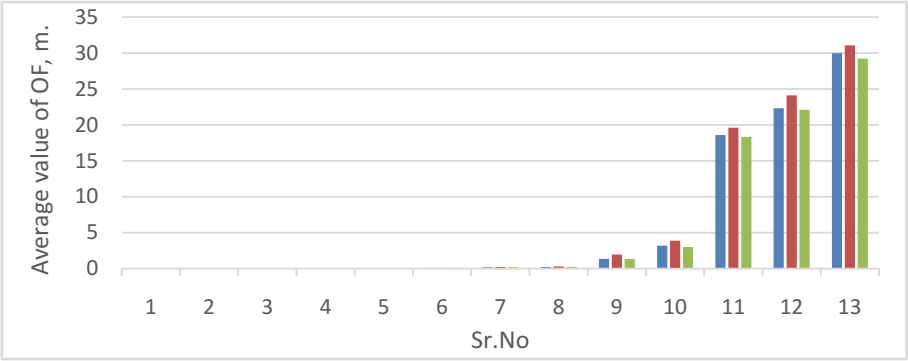

Fig. 7. Comparison of experimental results

7 Conclusion

In the paper the formulation of parametric optimization problem was made. To solve it the authors suggested the optimization method which simulates a behavior of bacterial colony. This algorithm allows to control a search process and eliminate the preliminary convergence problem. Software was developed in C++. To estimate time complexity of the developed algorithm there were carried out computational experiment. So, there were obtain empirical decencies, range of variation of input parameters, as well as give recommendations for its optimal choice. On the basis of obtain results the authors conclude that algorithm time complexity do not exceed polynomial complexity.

Acknowledgements. This research is supported by the Council for Grants (under RF President), the project #MK-92.2017.8.

References

1. Kureichik, V.V., Kureichik, V.M., Malioukov, S.P., Malioukov, A.S.: Algorithms for Applied CAD Problems. Springer, Berlin (2009). 487 p.
2. Alpert, C.J., Dinesh, P.M., Sachin, S.S.: Handbook of Algorithms for Physical design Automation. Auerbach Publications Taylor & Francis Group, Boca Raton (2009)
3. Zaruba, D., Zaporozhets, D., Kureichik, V.: Artificial bee colony algorithm—a novel tool for VLSI placement. In: Abraham, A., Kovalev, S., Tarassov, V., Snášel, V. (eds.) Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry (IITI 2016). AISC, vol. 450, pp. 433–442. Springer, Cham (2016). doi:10.1007/978-3-319-33609-1_39
4. Zaruba, D., Zaporozhets, D., Kureichik, V.: VLSI placement problem based on ant colony optimization algorithm. In: Silhavy, R., Senkerik, R., Oplatkova, Z., Silhavy, P., Prokopova, Z. (eds.) Artificial Intelligence Perspectives in Intelligent Systems. AISC, vol. 464, pp. 127–133. Springer, Cham (2016). doi:10.1007/978-3-319-33625-1_12

5. Kureichik, V., Kureichik, V., Zaruba, D.: Hybrid bioinspired search for schematic design. In: Abraham, A., Kovalev, S., Tarassov, V., Snášel, V. (eds.) Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry (IITI 2016). AISC, vol. 451, pp. 249–255. Springer, Cham (2016)
6. Kureichik, V., Kureichik, V., Bova, V.: Placement of VLSI fragments based on a multilayered approach. In: Silhavy, R., Senkerik, R., Oplatkova, Z., Silhavy, P., Prokopova, Z. (eds.) Artificial Intelligence Perspectives in Intelligent Systems. AISC, vol. 464, pp. 181–190. Springer, Cham (2016). doi:10.1007/978-3-319-33625-1_17
7. Kureichik, V.V., Kravchenko, Y.A.: Bioinspired algorithm applied to solve the travelling salesman problem. World Appl. Sci. J. **22**(12), 1789–1797 (2013)
8. Kar, A.K.: Bio inspired computing - a review of algorithms and scope of applications. Expert Syst. Appl. **59**, 20–32 (2016)
9. Lim, S.K.: Practical Problems in VLSI Physical Design Automation. Springer Science +Business Media B.V, Heidelberg (2008)
10. Yang, C., Ji, J., Liu, J., Yin, B.: Bacterial foraging optimization using novel chemotaxis and conjugation strategies. Inf. Sci. **363**, 72–95 (2016)
11. Zhao, W., Wang, L.: An effective bacterial foraging optimizer for global optimization. Inf. Sci. **329**, 719–735 (2016)
12. Kureichik, V.V., Zaruba, D.V.: The bioinspired algorithm of electronic computing equipment schemes elements placement. In: Silhavy, R., Senkerik, R., Oplatkova, Z., Silhavy, P., Prokopova, Z. (eds.) Artificial Intelligence Perspectives in Intelligent Systems. AISC, vol. 347, pp. 51–58. Springer, Cham (2015). doi:10.1007/978-3-319-18476-0_6
13. Zaporozhets, D., Zaruba, D.V., Kureichik, V.V.: Hierarchical approach for VLSI components placement. In: Silhavy, R., Senkerik, R., Oplatkova, Z., Silhavy, P., Prokopova, Z. (eds.) Artificial Intelligence Perspectives in Intelligent Systems. AISC, vol. 347, pp. 79–87. Springer, Cham (2015). doi:10.1007/978-3-319-18476-0_9
14. Hernández-Ocaña, B., Mezura-Montes, E., Pozos-Parra, Ma.D.P. Evolutionary bacterial foraging algorithm to solve constraint numerical optimization problems. In: CEUR Workshop Proceedings, vol. 1659, pp. 58–65 (2016)