Hybrid binary GA-EDA algorithms for complex "black-box" optimization problems

E Sopov

Reshetnev Siberian State Aerospace University, 31 Krasnoyarskiy Rabochiy Ave., 660037, Krasnoyarsk, Russia

E-mail: evgenysopov@gmail.com

Abstract. Genetic Algorithms (GAs) have proved their efficiency solving many complex optimization problems. GAs can be also applied for "black-box" problems, because they realize the "blind" search and do not require any specific information about features of search space and objectives. It is clear that a GA uses the "Trial-and-Error" strategy to explorer search space, and collects some statistical information that is stored in the form of genes in the population. Estimation of Distribution Algorithms (EDA) have very similar realization as GAs, but use an explicit representation of search experience in the form of the statistical probabilities distribution. In this study we discus some approaches for improving the standard GA performance by combining the binary GA with EDA. Finally, a novel approach for the large-scale global optimization is proposed. The experimental results and comparison with some well-studied techniques are presented and discussed.

1. Introduction

Many real-world problems can be stated as a problem of finding an optimal alternative from a list of options. The problems are decision making, design, structure and parameters optimization, control and many other. Many real-world design and decision making support problems are complex and bad-formalized, thus the quality criterion is usually considered as the "black-box" model [1]. Moreover, alternatives cam be represented by complex structures that contain variable parameters of many different types, including categorical, integer, rank, binary and others. There exist many efficient and well-studied techniques that can deal with the global black-box (BB) optimization problems.

One of the main difficulties for designing better BB optimization techniques is the limitation of the number of objective evaluations. There exist the so-called "cheap" BB problems, where the objective can be evaluated sufficiently many times (many thousands), and the "expensive" or "costly" BB problems, where the objective evaluations are strongly limited (a few hundreds). In the case of costly problems, we cannot provide adequate analysis of the search space and the objective landscape. Thus the "blind" search that implying more advanced optimization techniques like genetic algorithms (GAs) is more preferable [2]. At the same time GAs are also computational costly methods even they apply parallel stochastic search. Moreover, the GA's performance usually decreases when the dimensionality of the search space increases. There exists the challenge of designing and investigating approaches that are able to better exploit an experience of the past search trials and to improve the convergence to the optimal solution.

In this study we will combine the Estimation of Distribution Algorithm (EDA) and the GA for extracting statistical data about the GA's past search experience (using the EDA conception), and for using such statistical distribution in the GA.

The rest of the paper is organized as follows. Section 2 describes related work. Section 3 describes the proposed approaches. In Section 4 the results of numerical experiments are discussed. In the Conclusion the results and further research are discussed.

2. Related work

The BB optimization refers to a problem setup in which an optimization algorithm is supposed to optimize (minimize or maximize) an objective function and the algorithm may only query the value of the objective for a candidate-solution, but it does not obtain gradient information, and in particular it cannot make any assumptions on the analytic form of the objective. The goal of optimization is to find an as good as possible value the objective within a predefined time, often defined by the number of available queries to the black box (the number of the objective evaluations). Problems of this type regularly appear in practice, when optimizing parameters of a model is in fact hidden in a black box (there is no possibility to analyze the objective), or the problem is too complex to be modeled explicitly.

A large variety of optimization algorithms for the BB optimization has been proposed. The majority of techniques that demonstrates high performance with arbitrary BB problems are based on evolutionary (including genetic) and bio-inspired algorithms. Nevertheless, the BB optimization is still a challenge for the algorithms. In recent years many BB optimization competitions have been held to compare performances of different traditional and new approaches (the most known are the BBOB/COCO as a part of the GECCO conference, and the BBComp as a part of the IEEE CEC conference). The given results show that BB problems are still hard for existing optimizers.

Another essential difficulty for optimization algorithms is related with an effect that is called the "curse of dimensionality". Performance of algorithms usually decreases when the dimensionality of the search space increases. Nowadays, optimization problems with many hundreds or thousands of objective variables are called large-scale global optimization (LSGO) problems. BB LSGO problems have become a great challenge even for optimizers especially with non-separable problems that excludes a straightforward variable-based decomposition. Nevertheless, some assumption can be done, and there exist many efficient LSGO techniques for the continuous search space [3].

Many real-world optimization problems encode different complex structures and contain variables of many different types, which cannot be represented only by real values. In this case binary representation of candidate solutions can be used. In this study we will use the binary GAs. As we can see from papers, the majority of approaches for BB and BB LSGO are designed for the continuous search space, and there is a lack of approaches using the binary GAs as the core technique.

As previously mentioned, population-based stochastic algorithms collect and exploit some statistical information about search space that is not presented in explicit way. At the same time there exist a family of algorithms that are also operate with a set of candidate-solutions at each trial search step, and estimate and use an explicit probability distribution to generate new solutions. Such approaches are called Estimation of Distribution Algorithms [4]. There also exist binary EDAs.

There are some straight-forward attempts to combine a GA with the binary EDA. The main idea to estimate a distribution of genes with one-values in chromosomes of a current population. The distribution then is applied to generated next generation instead of applying the standard crossover. There exist some well-studied approaches like Probabilistic Incremental Learning Algorithm (PBIL) [5], hybrid EDA-EA [6,7] and other.

In [8] similar approach has been proposed, in [9] the approach has been applied for multi-objective optimization problems.

3. Proposed approaches

A binary GA is conventional evolutionary algorithm (EA) with a population of solutions, which are represented by fix-length vector containing zero- or one-value in each position. The binary GA and the EDA have very similar schemes. As we can from Table 1, the only differences in algorithms the way they collect and exploit search experience. GAs collect and update statistics inside population, EDAs do that using external explicit distribution.

Table 1. Comparison of general steps of GA and EDA algorithms.

GA	EDA
1. Initialize population with random. 2. Evaluate current population. 3. Form a new population using genetic operators. 4. Update current population using the given strategy. 5. Until a stop criterion is satisfied, repeat steps 2-4.	1. Initialize population with random. 2. Evaluate current population. 3. Form a new population according to the probability distribution. 4. Update the distribution using the given strategy. 5. Until a stop criterion is satisfied, repeat steps 2-4.

We can estimate a distribution of one-values in population of the GA by evaluating a probabilities vector for the population using the binary EDA way (see example on Figure 1):

$$P^k = \left(p_1^k, p_2^k, \dots, p_n^k\right), p_i^k = P\left(\left(x^i\right)^k = 1\right) = \frac{1}{N}\sum_{j=1}^{N}\left(x_j^i\right)^k \ , \forall i, k \qquad (1)$$

where p_i^k is the probability of the one-value in the *i*-th gene in chromosome, *n* is a chromosome length, x_j^i is the value of the *j*-th gene of *i*-th chromosome, *k* is the current generation number.

Gene position	1	2	3	4	5	6	7	8
Individual 1	1	1	1	1	0	0	0	0
Individual 2	1	1	0	0	0	0	0	1
Individual 3	0	1	0	1	1	0	0	0
Individual 4	0	1	1	0	0	0	1	1
Individual 5	0	1	0	0	0	0	0	0
Individual 6	0	1	1	0	1	0	0	1
Individual 7	1	1	1	0	0	0	0	0
Individual 8	1	1	1	1	0	0	0	1
Individual 9	0	1	0	1	0	0	0	0
Individual 10	1	1	1	0	1	0	1	1
Distribution	0,5	1	0,6	0,4	0,3	0	0,2	0,5

Figure 1. Example of distribution estimation in a GA.

3.1. Optimal solution prediction for the BB problems using EDA-based GA

We can investigate the performance of binary GAs using a visual representation of the distribution changes over the GA run. In [10] the following approach have been proposed. After the given GA finishes the run, we create a diagram for each component of the probabilities vector. The vertical axis contains probability values; the horizontal axis contains numbers of generation. An example of a component changes inside a GA run is presented on Figure 2.

Analyzing many diagrams for different BB optimization problems we can find that the components of the probabilities vector frequently converge to one-value if optimal solution contains one in corresponding position and or to zero-value if the optimal solution contains zero. Our hypothesis is

that probability converges to one (or zero) if corresponding gene of the best-found solution is equal to one (or zero). The higher algorithm performance the more often probabilities converge to the correct value.

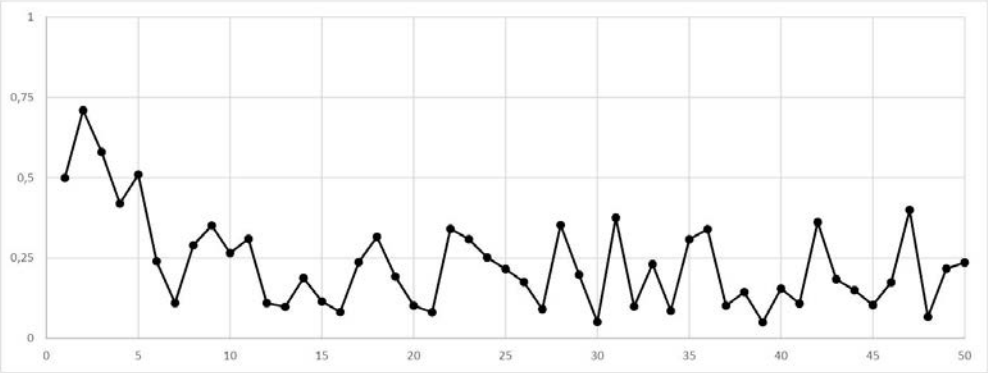

Figure 2. Example of a probabilities vector component changes.

We can propose the following strategies of implementing the convergence feature in GAs:

- *Straight-forward optimal solution prediction.*

When applying a GA to an optimization problem, the distribution of one-values in populations are estimated and collected. Then we visually analyze the convergence of each component of the probabilities vector and define genes of the optimal solution as:

$$x_i^{optimal} = \begin{cases} 1, if\ p_i \to 1 \\ 0, if\ p_i \to 0 \end{cases} \quad (2)$$

The predicted value can precise or match the best-found individual.

- *Integral criterion for the prediction.*
-

$$x_i^{optimal} = \begin{cases} 1, if\ \sum_{g=1}^{MaxGen}\left(p_i^j - 0.5\right) > 0 \\ 0, otherwise \end{cases}, \quad (3)$$

where MaxGen is maximum number of generations.

The predicted value can precise or match the best-found individual.

We can also evaluate prediction on early generations and include the predicted solution into current population (for example, replacing the worst individual) to enhance search process.

- *Weighted integral criterion for the prediction.*

The main idea of the given modification is that the probability values on the later iterations have the greater weights as algorithm collects more information about search space.

We have estimated the performance of the standard conventional GA, the GA using the distribution for updating next population instead of using a crossover operator and the GA with the optimal solution prediction using the weighted integral criterion at the final generation. We have included the following well-studied benchmark problems, that are known as complex for a GA optimizer: Rastrigin and Shifted Rotated Rastrigin, Rosenbrock, Griewank, De Jong 2, “Sombrero” (“Mexican Hat”), Schekel, Katkovnik and other. We have also included some deceptive “trap” problems like Ackley Trap, Whitley Trap, Goldberg Trap, Trap-4, Trap-5, DEC2TRAP, Liepins and Vose.

The numerical experiments have shown that the GA with prediction outperforms the standard approach. More detailed information on the experimental results can be found in [10].

3.2. Problem decomposition for the LSGO problems using EDA-based GA

As increasing dimensionality of the BB optimization problems decrease the performance of a GA, we will focus on the LSGO problem.

There exist a great variety of different LSGO techniques that can be combined in two main groups: non-decomposition methods and cooperative coevolution (CC) algorithms. The first group of methods are mostly based on improving standard evolutionary and genetic operations. But the best results and the majority of approaches are presented by the second group. The CC methods decompose LSGO problems into low dimensional sub-problems by grouping the problem subcomponents. CC consists of three general steps: problem decomposition, subcomponent optimization and subcomponent coadaptation (merging solutions of all subcomponents to construct the complete solution).

The problem decomposition is a critical step. There are many subcomponent grouping methods, including: static grouping, random dynamic grouping and learning dynamic grouping. A good survey on LSGO and methods is proposed in [3]. As we can observe in papers, almost all studies are focused on continuous LSGO, and there is a lack of techniques for binary (or other discrete) representations.

The main idea of the LSGO problem decomposition methods is based on the divide-and-conquer approach which decomposes the problem into single-variable or multiple-variable low dimensional problems. In this case, only part of the variables are used in the search process; the rest are fixed and their values are defined using some strategy (for example, values from the best-found solution are used).

The finding of an appropriate decomposition is part of the general search process. It is obvious and has been presented in many studies that the best performance is achieved with separable LSGO problems. In the case of non-separable problems, the performance strongly depends on the decomposition strategy.

In this work, we will formulate the following requirements for the proposed decomposition method:

- The grouping should be dynamic to realize the “exploration and exploitation” strategy.
- The grouping should be random to avoid the greedy search and the local convergence.
- The grouping should be based on the past search experience of the whole population (to provide the global search options).
- The grouping should be adaptively scalable to provide efficient decomposition at every stage of the search process.

We will use previously discussed convergence property to define the values for fixing genes in chromosomes at the grouping stage. If the *i*-th position in a chromosome at the *t*-th generation is fixed, its value is defined by the corresponding value of the probability vector:

$$x_i\ (t) = \begin{cases} 0,\ if\ p_i(t) < (0.5 - \delta) \\ random,\ \ if\ (0.5 - \delta) \le p_i(t) \le (0.5 + \delta) \\ 1,\ if\ p_i(t) > (0.5 + \delta) \end{cases} \quad (4)$$

where δ is a threshold (a confidence level), $\delta \in (0, 0.5)$.

We will explain the proposed approach using Figure 3. The diagram visualizes an arbitrary component of the probability vector for an arbitrary run of a GA on a benchmark BB problem. For the chosen gene the corresponding value of the optimal solution is equal to zero. As we can see from Figure 3, the GA starts with random initialization, thus the value of the probability vector is equal to 0.5. At the first generations the GA actively explores the search space and number of 1’s and 0’s genes are almost equal, thus the value of the probability vector is still about 0.5. After that, the GA locates a promising region in the search space and increases the number of 0’s in this position, thus the value of the probability vector decreases towards zero.

The confidence level δ is a parameter that defines a threshold for the probability value around 0.5, where we cannot make a decision about the gene value.

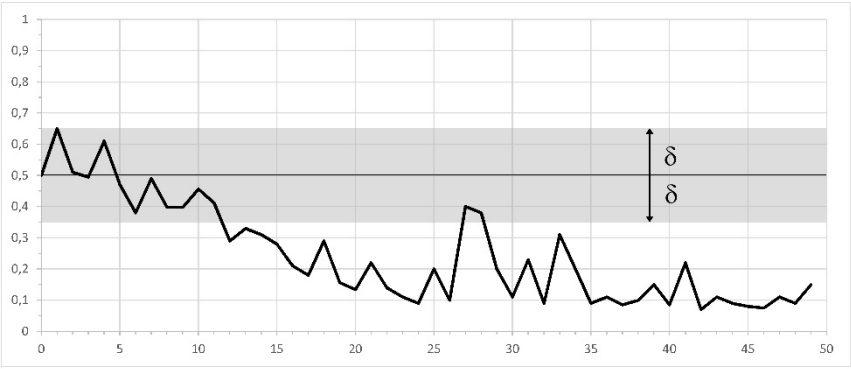

Figure 3. Making decision about a gene's values fixation using the analysis of the dynamic of the probability vector component.

Although a decision about fixed variables is made by stand-alone components, the estimated distribution contains information about the problem solving in general. Thus the method is not focused only on separable LSGO problems.

Next we need to define the number of variables that will be fixed. There exist many strategies. For example, the splitting-in-half method divides an *n*-dimensional problem into two *n/2* subcomponents. In general, we will define the number of fixed variables as a percentage of the chromosome length and will denote it as α. The value of α can be constant or can change during the run of the algorithm. The variables and corresponding components of the probability vector are fixed for some predefined number of generations, which is called an adaptation period (denoted as t_{adapt}). The list of fixed components is randomly defined.

In this paper, the straight-forward approach is used, α and t_{adapt} are predefined and constant.

The main advantage of such EDA-based decomposition is that we do not lose the previously collected statistic as we fix components of the probability vector. The GA solves the problem of reduced dimensionality and updates the probability only for active components. After each adaptation period we will randomly fix other components, and the previously fixed components will continue updating their saved values.

We will describe the proposed LSGO algorithm in detail.

First, we need to encode the initial problem into a binary representation. The standard binary or Grey code can be used. A chromosome length *n* is defined. Next, specific parameters of the EDA-based decomposition and the chosen GA, maximum number of fitness evaluations (*MaxFE*) or maximum number of generations (*MaxGEN*) are defined. The maximum number of generations can be substituted with any other stop condition (for example, time-based condition).

Finally, the following algorithm is used (see Figure 4).

As is known, the island model GA can outperform the standard single-population GA for many complex optimization problems [11]. We can also decrease the computational time by implementing the island GA with a parallel multi-core or multi-processor computer.

Input: n, N, α, δ, t_{adapt}, $MaxFE$, the GA operators' parameters.

Initialization:
Randomly generate a population of N individuals of the length n.
Calculate $P(0)$ using formula (1).

Main loop.
Until $MaxFE$ is reached:
1. Problem decomposition stage: Start new adaptation period. Fix random α components in chromosomes and in the probability vector.
2. Subcomponent optimization stage: Run the GA for t_{adapt} generations:
   a. Fitness evaluation. Set values in fixed positions of chromosomes according to $P(t)$ using formula (2).
   b. Perform selection, crossover and mutation operations.
   c. Create next generation, update the probability vector $P(t)$ for active components.

Output: the best-found solution.

Figure 4. EDA-based decomposition GA for LSGO.

4. Numerical experiments

We have used 15 large-scale benchmark problems from the CEC'2013 Special Session and Competition on Large-Scale Global Optimization [12]. These problems represent a wider range of real-world BB LSGO problems. There are 3 fully-separable problems (denoted as f1-f3), 8 partially separable problems (f4-f7 with a separable subcomponent and f8-f11 with no separable subcomponents), 3 problems with overlapping subcomponents (f12-f14), and 1 non-separable problem (f15).

The experiment settings are:

- Dimensions for all problem are D=1000;
- The standard binary encoding is used with accuracies: ε=0.1 for f1, f4, f7, f8 and f11-15, ε=0.05 for f3, f6 and f10, and ε=0.01 for f2, f5 and f9;
- For each problem the best, mean, and standard deviation of the 25 independent runs are evaluated;
- Maximum number of fitness evaluations is MaxFE=3.0e+6;
- The performance estimation is performed for the number of fitness evaluations equal to 1.2e+5, 6.0e+5 and 3.0e+6.

The EDA-based decomposition GA settings are:

- Population sizes are N=1000 for the single-population version, N=500 for the island version with 3 islands, and N=400 for 5 islands;
- The adaptation period is t_{adapt}=100;
- The probability threshold is δ=0.05, 0.15 and 0.25;
- Numbers of fixed components are α=25%, 50% and 75% of the chromosome length.

All algorithms have been implemented in Visual Studio C++ using the OpenMP for parallel computing with multi-core PC. Free C++ source codes of the benchmark problems are taken from (http://goanna.cs.rmit.edu.au/~xiaodong/cec13-lsgo/, 2013).

We have carried out the above-mentioned experiments and have established the following. In the case of single population, the best performance on average is achieved with 50% fixed components and δ=0.15. In the case of the island model, the best results are obtained by the 5 island model with 75% fixed components and δ=0.15. Almost for every considered value of parameters, the island model outperforms the single population version of the algorithm.

The experimental results for the best found settings are presented in Table 2.

Table 2. Experimental results for the EDA-based decomposition GA with 5 islands and α=75%.

		f1	f2	f3	f4	f5	f6	f7	f8
1.2e5	Best	1.42E+07	9.63E+03	1.08E+02	1.39E+11	6.11E+14	2.90E+05	7.05E+08	3.97E+15
	Mean	5.50E+07	1.06E+04	4.52E+01	9.15E+11	7.17E+14	7.78E+05	2.76E+09	2.71E+16
	StDev	2.98E+07	1.53E+03	1.44E+01	5.63E+11	7.45E+08	2.61E+05	1.44E+09	6.77E+15
6.0e5	Best	6.89E+03	9.11E+03	3.04E+00	1.95E+10	3.07E+14	5.21E+05	2.01E+08	2.09E+14
	Mean	1.99E+04	1.25E+04	1.30E+01	9.07E+10	5.03E+14	6.05E+05	9.41E+08	2.18E+15
	StDev	1.68E+03	1.18E+03	6.32E-01	6.03E+10	2.50E+07	2.60E+05	7.56E+08	1.52E+15
3.0e6	Best	4.59E-05	1.82E+03	2.94E-05	6.60E+09	7.59E+14	6.25E+04	7.65E+07	4.49E+13
	Mean	5.68E-04	3.34E+03	4.81E-01	2.32E+10	9.75E+14	4.75E+05	2.53E+08	3.64E+14
	StDev	4.29E-04	2.54E+02	2.28E-01	1.14E+10	2.18E+06	3.35E+05	8.35E+07	5.21E+14
		f9	f10	f11	f12	f13	f14	f15	Average
1.2e5	Best	1.08E+09	8.87E+06	1.46E+11	3.87E+06	2.66E+10	1.88E+11	3.61E+07	3.05E+14
	Mean	1.80E+09	7.14E+07	3.47E+11	4.36E+08	2.98E+10	5.78E+11	2.69E+08	1.85E+15
	StDev	4.27E+08	1.57E+07	2.25E+11	7.89E+08	1.12E+10	3.67E+11	9.91E+07	4.52E+14
6.0e5	Best	6.42E+08	7.91E+06	1.34E+10	2.40E+03	6.28E+09	5.68E+10	1.80E+07	3.44E+13
	Mean	1.25E+09	1.38E+07	9.84E+10	6.66E+03	1.47E+10	1.03E+11	2.43E+07	1.79E+14
	StDev	5.21E+08	1.65E+07	1.18E+11	5.53E+03	4.83E+09	6.70E+10	8.68E+06	1.02E+14
3.0e6	Best	4.15E+08	6.18E+06	2.60E+10	7.72E+02	8.02E+09	1.42E+10	2.40E+07	5.36E+13
	Mean	8.06E+08	1.61E+07	7.01E+10	2.30E+03	1.27E+10	1.69E+11	3.05E+07	8.93E+13
	StDev	1.72E+08	7.89E+06	4.29E+10	2.41E+03	2.96E+09	4.81E+10	5.13E+06	3.47E+13

The summary results are compared with other techniques presented at the CEC'13 competition. The algorithms are DECC-G (differential evolution (DE) based cooperative coevolution (CC) with random dynamic grouping), VMO-DE (variable mesh optimization using differential evolution), CC-CMA-ES (Covariance Matrix Adaptation Evolution Strategy using Cooperative Coevolution), MOS (Multiple Offspring Sampling (MOS) based hybrid algorithm), and SACC (smoothing and auxiliary function based cooperative coevolution) [13]. We have averaged the performance estimates of all algorithms over all problems and have ranked algorithms by the Best and the Mean values. The results are in Table 3.

As we can see from Table 3, the proposed approach has taken 4th place by the Best criterion and 5th place by the Mean value. We should note that all algorithms except the proposed are specially designed for continuous LSGO problems. The EDA-based decomposition GA does not use any knowledge about search space. Moreover, the chromosome length in the binary algorithm is greater than in the case of the continuous space. Nevertheless, the EDA-based decomposition GA outperforms the CC-CMA-ES by two measures and the DECC-G by the Best value on average.

Table 3. LSGO approaches comparison.

Algorithm	SACC	MOS	VMO-DE	DECC-G	CC-CMA-ES	EDA-GA
Best	9.80E+12	2.17E+11	4.90E+13	5.80E+13	6.25E+13	5.36E+13
Ranking by Best	2	1	3	5	6	4
Mean/ StDev	8.0E+13/ 5.08E+13	5.33E+11/ 2.04E+11	5.32E+13/ 4.81E+12	7.7E+13/ 1.02E+13	8.58E+13/ 2.39E+13	8.93E+13/ 3.47E+13
Ranking by Mean	4	1	2	3	6	5

Although the proposed approach yields to some of LSGO algorithms, our hypothesis is that the proposed approach will be a good tool for solving complex real-world LSGO problems, which usually contain not only continuous variables, but mixed-type variables and can represent arbitrary complex structures. Further investigations of the algorithm structure and parameters can probably improve its performance. In particular, the α value can be adjusted adaptively during the algorithm run using information about the probability vector convergence.

5. Conclusions and further work

In this study we have presented and discussed an approach of combining the binary GA and the EDA algorithms to improve the GA performance via estimating and exploiting explicit statistics about one-values distribution in the GA's population. The estimated distribution can be used for many purposes. We have proposed the optimal solution prediction method which is based on the analysis of the distribution components' convergence. The results of numerical experiments have shown that the GA with such prediction outperform the standard GA. We have also proposed a novel LSGO algorithms with EDA-based decomposition which is also based on the distribution analysis for fixing genes values in chromosomes. The approach performs well on complex BB LSGO benchmark problems and demonstrates results comparable with the state-of-art techniques.

In further work, more detailed analysis of the EDA-based decomposition GA parameters will be provided. A self-configuration will be introduced into the algorithm. We also will try to apply the distribution analysis for other classes of optimization problems.

6. Acknowledgements

The research was supported by the President of the Russian Federation grant (MK-3285.2015.9).

References

[1] Audet Ch 2014 A Survey on Direct Search Methods for Blackbox Optimization and Their Applications *Mathematics Without Boundaries Springer New York* pp 31–56
[2] Srinivas M and Patnail L M 1994 Genetic algorithms: A Survey *Computer* **27 (6)** pp 17-26
[3] Mahdavi S, Shiri M E and Rahnamayan Sh 2015 Metaheuristics in large-scale global continues optimization: A survey *Information Sciences* **295** pp 407–428
[4] Hauschild M and Pelikan M 2011 An introduction and survey of estimation of distribution algorithms *Swarm and Evolutionary Computation* **1** pp 111-128
[5] Baluja Sh 1994 Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning *Technical Report Carnegie Mellon University* 41 p.
[6] Dong W, Chen T, Tino P and Yao X 2013 Scaling up estimation of distribution algorithms for continuous optimization *IEEE Trans. Evol. Comput.* **17 (6)** pp 797–822
[7] Wang Y and Li B 2008 A restart univariate estimation of distribution algorithm: sampling under mixed gaussian and lévy probability distribution *IEEE Congress on Evolutionary Computation CEC* pp 3917–3924
[8] Sopov E, et al. 2009 A Modified probabilistic genetic algorithm for the solution of complex constrained optimization problems *Vestnik of Siberian State Aerospace University* **5(26)** pp 31-36
[9] Sopov E and Sopov S 2011 On probabilistic genetic algorithm with adaptive mutation and Pareto-solutions prediction for complex multi-objective optimization problems *Vestnik of Samara State Aerospace University* **6(30)** pp 273-283 (in Russian).
[10] Sopov E and Semenkina O 2015 The optimal solution prediction for genetic and distribution building algorithms with binary representation *IOP Conf. Ser. Mater. Sci. Eng.*. **70** 10 p
[11] Gonga Y-J, et al. 2015 Distributed evolutionary algorithms and their models: A survey of the state-of-the-art *Applied Soft Computing* **34** pp 286–300
[12] Li X, et al. 2013 Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization *Technical Report Evolutionary Computation and Machine Learning Group RMIT University*.
[13] Li X, et al. 2013 Technical report on 2013 IEEE Congress on Evolutionary Computation Competition on Large Scale Global Optimization *Available at: http://goanna.cs.rmit.edu.au/~xiaodong/cec13-lsgo/competition/lsgo-competition-sumary-2013.pdf.*