INFORMATION TECHNOLOGY, COMPUTER SCIENCE, AND MANAGEMENT
ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

UDC 681.3.681.5
https://doi.org/10.23947/1992-5980-2019-19-4-389-397

Genetic algorithm efficiency improvement in the course of set cover problem solution[*]

I. S. Konovalov[1], V. A. Fatkhi[2], V. G. Kobak[3]**
[1,2,3]Don State Technical University, Rostov-on-Don, Russian Federation

Повышение эффективности работы генетического алгоритма в процессе решения задачи покрытия множеств[***]

И. С. Коновалов[1], В. А. Фатхи[2], В. Г. Кобак[3]**
[1,2,3]Донской государственный технический университет, Ростов-на-Дону, Российская Федерация

Introduction. Practical tasks (location of service points, creation of microcircuits, scheduling, etc.) often require an exact or approximate to exact solution at a large dimension. In this case, achieving an acceptable result requires solving a set cover problem, fundamental for combinatorics and the set theory. An exact solution can be obtained using exhaustive methods; but in this case, when the dimension of the problem is increased, the time taken by an exact algorithm rises exponentially. For this reason, the precision of approximate methods should be increased: they give a solution that is only approximate to the exact one, but they take much less time to search for an answer at a large dimension.
Materials and Methods. One of the ways to solve the covering problem is described, it is a genetic algorithm. The authors use a modification of the Goldberg model and try to increase its efficiency through various types of mutation and crossover operators. We are talking about gene mutations, two-point mutations, addition and deletion mutations, insertion and deletion mutations, saltation, mutations based on inversion. The following types of crossover operator are noted: single-point, two-point, three-point and their versions with restrictions, uniform, triad. The effect of the stopping condition and the probability values of genetic operators on the accuracy of the solutions is investigated. It is shown how an increase in the number of individuals in a generation affects the efficiency of a solution.
Research Results. The experiment results allow us to draw three conclusions.
1) It is recommended to use a combination of gene mutation and single-point crossing.
2) With an increase in the number of individuals, the accuracy of the result and the time to obtain it increases. The average deviation from the exact result at a task size of 25 × 25 was 0%, at 50 × 50 – 0%, at 75 × 75 – 0.013%, at 100 × 100 – 0%, at 110 × 110 – 0% (the number of individuals was 500).
3) It is advisable to use the probabilities of the mutation and

Введение. Практические задачи (размещение пунктов обслуживания, создание микросхем, составление расписаний и пр.) зачастую требуют точного или приближенного к точному решения при большой размерности. Достижение приемлемого результата в данном случае требует решения задачи покрытия множеств — фундаментальной для комбинаторики и теории множеств. Точное решение можно получить с помощью переборных методов, однако в этом случае при повышении размерности задачи во много раз возрастает время работы точного алгоритма. По этой причине следует увеличивать точность приближенных методов: они дают решение, лишь приближенное к точному, однако затрачивают на поиск ответа намного меньше времени при большой размерности.
Материалы и методы. Описывается один из способов решения задачи покрытия — генетический алгоритм. Авторы используют модификацию модели Голдберга и пытаются повысить ее эффективность с помощью различных видов оператора мутации и скрещивания. Речь идет о генной мутации, двухточечной мутации, мутации добавления и удаления, мутации вставки и удаления, сальтации, мутациях на основе инверсии. Отмечены следующие виды оператора скрещивания: одноточечный, двухточечный, трехточечный и их версии с ограничениями, равномерный, триадный. Исследуется влияние условия останова и значений вероятностей генетических операторов на точность получаемых решений. Показано, каким образом увеличение числа особей в поколении влияет на эффективность решения.
Результаты исследования. Итоги экспериментов позволяют сделать три вывода.
1) Рекомендуется использовать сочетание генной мутации и одноточечного скрещивания.
2) При повышении количества особей растет точность результата и время его получения. Среднее отклонение от точного результата при размере задачи 25×25 составило 0 %, при 50×50 — 0%, при 75×75 — 0,013 %, при 100×100 — 0 %, при 110×110 — 0 % (количество особей — 500).
3) Целесообразно использовать вероятности оператора

[*] The research is done within the frame of the independent R&D.
[**] E-mail: xigorx92@mail.ru, fatkhi@mail.ru, valera33305@mail.ru
[***] Работа выполнена в рамках инициативной НИР.

crossover operator 100% and 100%, respectively.
Discussion and Conclusions. Recommendations are given to improve the efficiency of covering problem solution. To this end, a preferred combination of the genetic algorithm parameters, of types of crossover and mutation operators is indicated.

мутации и скрещивания 100 % и 100 % соответственно.
Обсуждение и заключения. Даны рекомендации, позволяющие повысить эффективность решения задачи покрытия. С этой целью указано предпочтительное сочетание параметров генетического алгоритма, типов операторов скрещивания и мутации

Keywords: genetic algorithm, set cover problem, Goldberg model, stopping condition, crossing, mutation.

Ключевые слова: генетический алгоритм, задача покрытия множеств, модель Голдберга, условие останова, скрещивание, мутация.

For citation: I.S. Konovalov, et al. Genetic algorithm efficiency improvement in the course of set cover problem solution. Vestnik of DSTU, 2019, vol. 19, no. 4, pp. 389–397. https://doi.org/10.23947/1992-5980-2019-19-4-389-397

Образец для цитирования: Коновалов, И. С. Повышение эффективности работы генетического алгоритма в процессе решения задачи покрытия множеств / И. С. Коновалов, В. А. Фатхи, В. Г. Кобак // Вестник Донского гос. техн. ун-та. — 2019. — Т. 19, № 4. — С. 389–397. https://doi.org/10.23947/1992-5980-2019-19-4-389-397

Introduction. Many of practical problems require an exact or approximate to exact solution with high dimensionality. Among these tasks there are the location of service points, the creation of microcircuits, scheduling. In this case, achieving an acceptable result requires solving a set cover problem, which is fundamental for combinatorics and the set theory. An exact solution can be obtained using exhaustive methods (for example, the branch-and-bound method). Naturally, with an increase in the problem dimension, the time taken by the exact algorithm rises exponentially. For this reason, the accuracy of approximate methods should be increased: they give a solution that is only approximate to the exact one, but they take much less time to find an answer with high dimensionality.

The following practical task can also serve as a good example. Assume, you need to assemble a team of specialists for a ship. Crew members should possess in aggregate all the required skills, but the number of co-workers should be minimal. This is an unweighted covering problem, that is, the "scales" of group members are the same and therefore not important. If to assign a certain value - weight (for example, working experience) to each member of the team, then the task will be balanced. An actual practical problem is to solve this problem in a shorter time, which provides achieving a result that is as close as possible to the exact one.

Materials and Methods

Research Objective. Given a population U of n elements and an aggregate of subpopulations U, $S = \{S_1,\ldots, S_k\}$. Each subpopulation S_i is associated with some non-negative cost $c: S \rightarrow Q^+$. $S' \subseteq S$ is a covering if any element of U belongs to at least one element of S' [1, 2].

The task can be presented in two versions: weighted and unweighted. The weighted covering problem involves finding an aggregate of subpopulations that covers the whole population U and has minimum weight. In the unweighted version, the resulting population should have the smallest possible number of subpopulations.

Problem-solving techniques. Genetic algorithm. Covering problems are solved using heuristic methods, approximate algorithms with a priori estimate, and exact algorithms [3, 4].

Exact algorithms (the best-known of them is the branch-and-bound method) give an exact solution, but are useless in large-dimensional problems, because they take too much time. If the accuracy of the solution can be neglected to a certain extent, it is recommended to use approximate algorithms [5] which solve the problem in an acceptable time. We are talking about algorithms with a priori estimate (for example, the greedy algorithm [6]) and probabilistic heuristics (ant colony method [7, 8], neural networks, evolutionary calculations).

This paper discusses genetic algorithms (GA) and ways to improve their efficiency. In 1975, John Holland proposed a probabilistic GA based on the rules of natural selection and inheritance. The GA properties are studied in [10, 11]. A detailed description of the applicability of the genetic algorithm for solving a covering problem is given in [1]. GA application methods for this task are described in [12, 13].

The authors use the Goldberg model [14] which is modified as follows: various types of the mutation and crossover operator are used, protection against the appearance of "incorrect" covering under the variation of individuals is provided.

We describe basic parameters of this algorithm. Relating to the individual, binary coding is used ("0", "1"). The evaluation function can be expressed by the following formula:

$$\sum_{j=1}^{n} c_j x_j \rightarrow \min,$$

where x^k is n- dimensional vector for which the j-th element x^k_j is equal to 1 if the subpopulation Sj is an integral part of the covering and is equal to 0 otherwise; c_j is the cost of the subpopulation S_j.

The condition for stopping the algorithm is the number of generations of persistency of the solution.

The Goldberg model uses tournament elimination of individuals. The authors use the equal-probability random selection – the choice of two individuals of the generation to apply the crossover and (or) mutation operator to them.

In [15], a modification of this algorithm using the strategy of elitism is described.

Overview of the types of crossover operator. When two individuals are crossed, the offsprings descend a part of the genes from each of the parents, and thereby the search space is expanded. In the classic GA version, a single-point crossover is used. Scientists involved in genetic algorithms offer their versions of this operator [16, 17]. As mentioned earlier, the authors have proposed binary coding of an individual, rather than real, so only some certain versions of all can be used. Crossover of the individuals with real genes is described in [16]. Here is an overview of the types of crossover appropriate for the application in this GA.

Single-point crossover (Fig. 1). Two individuals are selected for crossover.

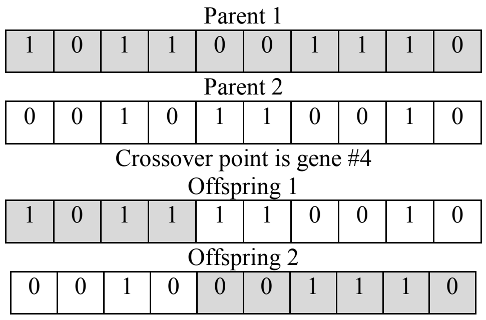

Fig. 1. Single-point crossover

The crossover point is played at random. A part of the genes of parent 1 is copied to offspring 1 to the crossover point, and a part of the genes of parent 2 is copied after the crossover point. Offspring 2 is created in a similar way, but vice versa.

Two-point crossover (Fig. 2). Two individuals are selected for crossover.

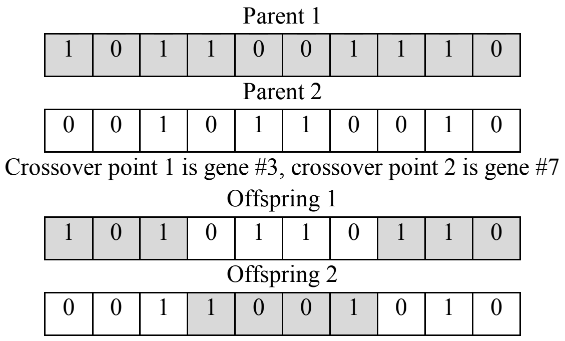

Fig. 2. Two-point crossover

Two different crossover points are played at random. A part of the genes of parent 1 to the crossover point 1, a part of the genes of parent 2 between the crossover points, and a part of the genes of parent 1 after the crossover point 2 are copied to offspring 1. Offspring 2 is generated in the same way, but vice versa.

A multipoint crossover and its special case, a three-point crossover, operate in like manner. The operators described can be modified, namely: verify, in addition, that the crossover points are selected only in those places where the genes of the individuals of the parents have different meanings. Thus, limited single-point, two-point, and three-point crossovers appeared.

Uniform crossover [16] (Fig. 3). A mask is randomly generated, a binary individual. In this case, a part of the offspring genes descends from one parent, and a part - from another.

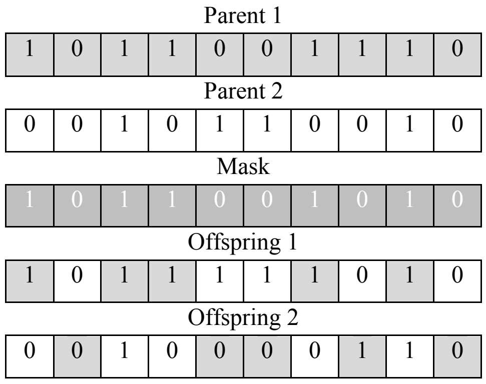

Fig. 3. Uniform crossover

Next, the mask is analysed. If it includes “1”, then the corresponding gene of parent 1 goes to the corresponding place of offspring 1. If otherwise, then offspring 1 descends the gene of parent 2.

Offspring 2 is generated in the opposite way. The gene is borrowed from parent 1 if there is “0” at the same place in the mask. If otherwise, then offspring 1 descends the parent gene.

A similar idea is used in the *triad crossover* [16]. The difference is that a randomly selected individual from a generation is used as a mask. Then, 10% of the mask genes undergo mutation. Further, if the gene of parent 1 matches the gene of the mask, then this gene proceeds to offspring 1, otherwise, the gene descends from parent 2. In offspring 2, at the places where offspring 1 descended the genes of parent 1, the genes of parent 2 are located, and vice versa.

Overview of the binary mutation operator types. What is the role of mutation in the evolutionary process? If only the crossover operator is used, in the end, the appearance of new individuals will be stopped. To qualitatively modify an individual, the mutation operator, which helps to increase genetic diversity, should be used.

In the classical GA, *single-point* mutation operator is used (Fig. 4): a mutation point is randomly selected in an individual — a gene which then swaps its value with the neighbouring gene.

Parent

1	0	1	1	0	0	1	1	1	0

Mutation point = gene #4

Offspring

1	0	1	0	1	0	1	1	1	0

Fig. 4. Single-point mutation operator

In addition to this mutation, several more types are considered.

Two-point mutation operator (Fig. 5) is a one-point mutation operator modification: two genes are randomly selected, and they exchange their values.

Parent

1	0	1	1	0	0	1	1	1	0

Mutation point = gene #4 and #7

Offspring

1	0	1	1	0	0	1	1	1	0

Fig. 5. Two-point mutation operator

Gene mutation (Fig. 6) is based on the fact that the value of one randomly selected gene is inverted.

Parent

1	0	1	1	0	0	1	1	1	0

Mutation point = gene #4

Offspring

1	0	1	0	0	0	1	1	1	0

Fig. 6. Gene mutation

Addition and deletion mutation [16] (Fig. 7) is obtained through combining two operations: adding a random gene to the chromosome tail and removing a random gene from the resulting chromosome.

Parent

1	0	1	1	0	0	1	1	1	0

Addition of gene «0» to individual tail

1	0	1	1	0	0	1	1	1	0	0

Deletion of gene #4

Offspring

1	0	1	0	0	1	1	1	0	0

Fig. 7. Addition and deletion mutation

Insertion and deletion mutation [16] is similar to the addition and deletion mutation: a random gene is added to a random chromosome position and a random gene is removed from the resulting chromosome.

Mutation based on mutation density [16]. Each gene of an individual mutates with a certain probability. The probability of gene mutation is usually selected so that 1% to 10% of the genes undergo modification.

Saltation [17] (Fig. 8) is mutation based on the inversion of *k*-genes of an individual.

Parent

1	0	1	1	0	0	1	1	1	0

k = 3, mutation of genes #4, #6, #7
Offspring

1	0	1	0	0	1	0	1	1	0

Fig. 8. Saltation

Inversion [17] (Fig. 9) is mutation of genes between two randomly selected change points.

Parent

1	0	1	1	0	0	1	1	1	0

Change points are gene #4 and #7
Offspring

1	0	1	0	1	1	0	1	1	0

Fig. 9. Inversion

Translocation [17] (Fig. 10) is mutation of genes which appear in two randomly selected parts of an individual.

Parent

1	0	1	1	0	0	1	1	1	0

Interval #1 = [gene #1; gene #3], Interval #2 = [gene #5; gene #7]
Offspring

0	1	0	1	1	1	0	1	1	0

Fig. 10. Translocation

Addition [17] is mutation in which an offspring individual is generated through inverting each gene of a parent individual.

Research Results

Analysis of the genetic algorithm performance using various "mutation + crossover" combinations. What combinations of types of binary mutation and crossover are used for more advantage to increase the GA efficiency? The authors have developed a software tool using the C Sharp language to compare genetic algorithms for optimal solutions and time consuming. For experiments, we used a personal computer with the Microsoft Windows 10 Pro × 64 operating system, Intel (R) Core (TM) i5-2500KCPU 3.30GHz processor, and 6 GB RAM.

100 experiments were carried out with $n \times m$ matrices, where n is the number of subpopulations of population U, m is the number of elements of the population U. The matrices are generated randomly. The following conditions are observed.

- Coefficient of the matrix for occupancy of subpopulations with units $p = 0.5$.
- Weights of subpopulations are randomly generated from the interval from 1 to 200.
- The number of subpopulations = 100, power of the population $U = 100$.

For GA, the following parameters are used.

- Number of generations = 50.
- Crossover probability = 1.
- Mutation probability = 1.
- Stopping condition = 100 generations.
- Crossover operator:

Cr1 – single-point;
Cr2 - limited single-point;
Cr3 - two-point;
Cr4 - limited two-point;
Cr5 - three-point;
Cr6 - limited three-point;
Cr7 - uniform;
Cr8 - triad.

- Mutation operator:

Mut1 - gene;

Mut2 – single-point;
Mut3 - two-point;
Mut4 - addition and deletion mutation;
Mut5 - insertion and deletion mutation;
Mut6 - saltation;
Mut7 - addition;
Mut8 - inversion;
Mut9 – translocation.

Table 1 shows the average values of the comparison results of the algorithms for cover weights, and Table 2 - by operating time. Also, Tables 1 and 2 include the results of a genetic algorithm operation with 50 individuals proposed by Nguyen Minh Hang in [13].

Table 1

Comparison of the efficiency of crossover and mutation operator types by cover weights

Algorithm 100×100 50 individuals	Mut1	Mut2	Mut3	Mut4	Mut5	Mut6	Mut7	Mut8	Mut9	Nguyen Minh Hang GA
Cr1	41.78	60.35	45.02	60.12	55.07	67.46	67.46	67.46	67.46	46.23
Cr2	42.29	58.23	44.87	59.37	51.83	67.46	67.46	67.46	67.46	
Cr3	42.53	58.75	45.38	61.35	55.93	67.41	67.46	67.72	67.46	
Cr4	42.91	63.63	45.75	63.38	57.64	67.46	67.46	67.46	67.46	
Cr5	42.41	60.58	45.18	63.11	54.92	67.46	68.52	67.46	67.46	
Cr6	42.71	65.96	46.37	65.38	58.2	67.46	67.46	67.46	67.32	
Cr7	41.74	50.61	45.52	53.75	48.31	67.46	67.46	67.29	67.46	
Cr8	43.39	57.84	45.3	60.37	54.07	67.46	67.46	67.46	67.46	

Table 2

Comparison of the efficiency of crossover and mutation operator types by time costs (ms)

Algorithm 100×100 50 individuals	Mut1	Mut2	Mut3	Mut4	Mut5	Mut6	Mut7	Mut8	Mut9	Nguyen Minh Hang GA
Cr1	2418	1363	2028	1996	1777	1569	2251	1746	1853	1900
Cr2	2365	1399	1974	2175	1817	1571	2257	1756	1855	
Cr3	2485	1417	2111	2008	1824	1627	2325	1841	1935	
Cr4	2568	1416	2126	2145	1884	1626	2304	1809	1909	
Cr5	2537	1406	2131	1862	1825	1631	2338	1834	1927	
Cr6	2509	1422	2139	1902	1822	1636	2315	1820	1903	
Cr7	2679	1569	1970	2220	2124	1697	2410	1905	2008	
Cr8	2484	1443	2084	1950	1910	1654	2353	1866	1942	

Following from these results, to improve the GA efficiency, it is recommended to use the "uniform crossover + gene mutation" and "single-point crossover + gene mutation" combinations.

Impact of the mutation and crossover probability on the genetic algorithm efficiency. To study this problem, the software tool described above was applied. The "gene mutation + uniform crossover" combination was considered as the most efficient (along with "gene mutation + single-point crossover"). The problem dimension is 100 × 100, 50 individuals. The results are given in Tables 3 and 4.

Table 3

Comparison of the efficiency of crossover and mutation operator probabilities by cover weights

Crossover \ Mutation	0.2	0.4	0.6	0.8	1
0.2	60.31	59.45	59.37	54.65	44.35
0.4	58.2	58.09	57.51	54.71	45.01
0.6	57.98	57.58	54.67	52.67	44.75
0.8	54.03	55.18	55.07	51.11	44.68
1	52.17	50.95	50.28	48.89	44.33

Table 4. Comparison of the efficiency of crossover and mutation operator probabilities by time costs (ms)

Crossover \ Mutation	0.2	0.4	0.6	0.8	1
0.2	992	1038	1121	1293	2047
0.4	1038	1101	1196	1418	2115
0.6	1111	1205	1325	1504	2273
0.8	1237	1314	1420	1656	2338
1	1338	1448	1602	1858	2594

The fittest combination was specified: the mutation probability is 100% and the crossover probability is 100%.

The generation dimension impact on GA efficiency. Tables 5 and 6 show the results with 50, 100, 200, 500, 1000 individuals and the problem dimension of 100 × 100 (GA1 - single-point crossover + gene mutation, GA2 - uniform crossover + gene mutation, GA3 - Nguyen Minh Hang’s GA).

Table 5

Generation dimension impact on cover weights obtained by genetic algorithm

Individuals	GA1	GA2	GA3
50	43.76	43.68	49.53
100	42.88	42.8	47.12
200	42.7	42.61	46.87
500	42.67	42.61	47.64
1000	42.61	42.61	50.35

Table 6

Generation dimension impact on the time required for genetic algorithm implementation (ms)

Individuals	GA1	GA2	GA3
50	2229	2377	1842
100	4175	4791	2219
200	8185	8722	2611
500	19109	20992	8440
1000	37588	41855	14581

Naturally, with an increase in the generation size, the operating time of the GA rises, and the problem solution accuracy increases.

The stopping condition impact on the problem solution efficiency. In the framework of the research presented, the number of generations of persistency of the fittest solution is used as a stopping condition. Tables 7 and 8 show the results of a comparative analysis of GA from a previous experiment with a stopping condition of 100, 200, 300, 500.

Table 7

Stopping condition impact on covering weights obtained by the genetic algorithm

Stopping condition	GA1	GA2	GA3
100	49.96	50.28	56.74
200	49.23	48.79	56.29
300	50.14	48.5	57.2
500	49.82	49.66	57.17

Table 8

Stopping condition impact on the time required to implement the genetic algorithm (ms)

Stopping condition	GA1	GA2	GA3
100	2264	2517	1834
200	3840	4251	3479
300	4994	5955	5001
500	7892	8429	8370

With an increase in the stopping condition, the algorithm running time increases. This is appropriate under the stopping condition of 200–250 individuals.

Discussion and Conclusions. The authors of this paper made an attempt to increase the GA efficiency as applied to a set cover problem. For this purpose, various types of the operator of mutation, crossover, and GA parametrization were used. The influence of the probabilities of genetic operators on the problem solution efficiency, the selection of the stopping condition and the number of individuals were investigated. The appropriate application scope of the GA and the branch-and-bound method are identified. Based on the results of the study, several conclusions can be drawn.

1) It is recommended to use a combination of gene mutation and single-point crossover.

2) If the number of individuals increases, the accuracy of the result and the time it is obtained increases. The average deviation from the exact result at the task dimension of 25 × 25 was 0%, 50 × 50 - 0%, 75 × 75 - 0.013%, 100 × 100 - 0%, 110 × 110 - 0% with 500 individuals.

3) It is efficient to use the probability of the mutation and crossover operator 100% and 100%, respectively.

References

1. Konovalov, I.S., Fatkhi, V.A., Kobak, V.G. Primenenie geneticheskogo algoritma dlya resheniya zadachi pokrytiya mnozhestv. [Application of genetic algorithm for the set-covering problem solution**.**] Vestnik of DSTU, 2016, no. 3, pp. 125–132 (in Russian).

2. Konovalov, I.S., Fatkhi, V.A., Kobak, V.G. Sravnitel'nyy analiz raboty zhadnogo algoritma Khvatala i modifitsirovannoy modeli Goldberga pri reshenii vzveshennoy zadachi nakhozhdeniya minimal'nogo pokrytiya mnozhestv. [Comparative analysis of work greedy algorithm of Chvatal and modified Goldberg models weighted in solving the problem of finding minimal coverings of sets.] Trudy SKF MTUSI, 2015, Part I, pp. 366–371, Rostov-on-Don: SKF MTUSI (in Russian).

3. Yeremeev, A.V., Zaozerskaya, L.A., Kolokolov, A.A. Zadacha o pokrytii mnozhestva: slozhnost', algoritmy, eksperimental'nye issledovaniya. [The set covering problem: complexity, algorithms, and experimental investigations.] Discrete Analysis and Operations Research, 2000, vol. 7, ser. 2, no. 2, pp. 22–46 (in Russian).

4. Yesipov, B.A., Muraviev, V.V. Issledovanie algoritmov resheniya obobshchennoy zadachi o minimal'nom pokrytii. [Study of algorithms for solving the generalized problem of the minimal covering.] Proc. of Samara Scientific Center RAS, 2014, no. 4 (2), pp. 308–312 (in Russian).

5. Kononov, A.V., Kononova, P.A. Priblizhennye algoritmy dlya NP-trudnykh zadac.h [Approximate algorithms for NP-hard problems.] Novosibirsk: Novosibirsk State University, 2014, 117 p. (in Russian).

6. Chvatal, V. A greedy heuristic for the set-covering problem. Mathematics of Operations Research, 1979, vol. 4, no. 3, pp. 233–235.

7. Lebedev, O.B. Pokrytie metodom murav'inoy kolonii. [Ant colony covering.] CAI-2010. XIIth National Conference on Artificial Intelligence with int. participation: Proc. Vol. 2. Moscow: Fizmatlit, 2010, pp. 423–431 (in Russian).

8. Lebedev, B.K., Lebedev, V.B. Pokrytie na osnove metoda roya chastits. [Particle Swarm Covering.] Neyroinformatika-2011: sb. nauch. tr. XIII Vseros. nauch.-tekhn. konf. Ch. 2. [Neuroinformatics-2011: Proc. XIII All-Russian Sci.-Tech. Conf. Part 2.] Moscow: Fizmatlit, 2011, pp. 93–103 (in Russian).

9. Holland, J. H. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1975, 245 p.

10. Stanovov, V.V., Semenkin, E.S. Issledovanie effektivnosti razlichnykh metodov samonastroyki geneticheskogo algoritma. [Study of efficiency of various methods of self-tuning of a genetic algorithm.] Actual problems of aviation and astronautics, 2012, no. 8, pp. 319–320 (in Russian).

11. Koromyslova, A.A., Semenkin, E.S. Issledovanie svoystva masshtabiruemosti geneticheskogo algoritma. [Investigation of the scalability property of a genetic algorithm.] Actual problems of aviation and astronautics, 2012, no. 8, pp. 305–306 (in Russian).

12. Yeremeev, A.V. Geneticheskiy algoritm dlya zadachi o pokrytii. [A genetic algorithm for the covering problem.] Discrete Analysis and Operations Research, 2000, ser. 2, vol. 7, no. 1, pp. 47–60 (in Russian).

13. Nguyen, M.H. Primenenie geneticheskogo algoritma dlya zadachi nakhozhdeniya pokrytiya mnozhestva. [Application of genetic algorithm to the problem of finding cover sets.] Dinamika neodnorodnykh system, 2008, vol. 33, iss. 12, pp. 206–219. Moscow: LKI, 2008 (in Russian).

14. Goldberg, D. E. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley, 1989, 432 p.

15. Konovalov, I.S., Fatkhi, V.A., Kobak, V.G. Strategiya elitizma modifitsirovannoy modeli Goldberga geneticheskogo algoritma pri reshenii zadachi pokrytiya mnozhestv. [Elitism strategy of modified Goldberg model of genetic algorithm in solving the set covering problem.] Herald of Computer and Information Technologies, 2016, no. 4, pp. 50–56 (in Russian).

16. Panchenko, T.V. Geneticheskie algoritmy. [Genetic algorithms.] Astrakhan: Astrakhan University, 2007, 88 p. (in Russian).

17. Batishchev, D.I. Geneticheskie algoritmy resheniya ekstremal'nykh zadach. [Genetic algorithms for solving extreme problems.] Voronezh: VSTU, 1995, 69 p. (in Russian).

Submitted 20.09.2019
Scheduled in the issue 20.11.2019

Authors:

Konovalov, Igor S.,
postgraduate student, Don State Technical University (1, Gagarin Square, Rostov-on-Don, 344000, RF),
ORCID: http://orcid.org/0000-0001-6296-3660
xigorx92@mail.ru

Fatkhi, Vladimir A.,
Head of the Computer Systems and Information Security Department, Don State Technical University (1, Gagarin Square, Rostov-on-Don, 344000, RF), Dr.Sci. (Eng.), professor,
ORCID: http://orcid.org/0000-0002-0373-7126
fatkhi@mail.ru

Kobak, Valery G.,
associate professor of the Computer and Automated Systems Software Department, Don State Technical University (1, Gagarin Square, Rostov-on-Don, 344000, RF), Dr.Sci. (Eng.), professor,
ORCID: http://orcid.org/0000-0002-1001-0574
valera33305@mail.ru