Genetic Algorithms for Problems of Logical Data Analysis in Discrete Optimization and Image Recognition

R. M. Sotnezov

Moscow State University, Computational Mathematics and Cybernetics Department, Leninskie gory 1, Moscow, 119992 Russia
e-mail: rom.sot@gmail.com

Abstract—Genetic algorithms for the search for minimal covering of a Boolean matrix are developed and studied. This problem arises in image recognition if methods of combinatorial (logical) analysis of information are used to synthesize recognizing procedures.

DOI: 10.1134/S1054661809030122

1. INTRODUCTION

In this paper one of the central problems of discrete mathematics, the problem of covering a set by a system of its subsets, is considered; this problem is formulated as the problem of finding the minimal covering of a Boolean matrix. The covering problem is related to the class of *NP*-complete problems; therefore, the known algorithms for the search for an exact solution possess exponential numerical complexity and are poorly applicable in practice. Approximate solutions are sought for problems with large dimensions. As a rule, the gradient algorithm gives good results. However, in a number of cases, for example, for sparse (with respect to the number of unities) matrices, the quality of the solution produced by the gradient algorithm sharply worsens. Therefore, the problem of developing a fast heuristics which yield good approximate solutions to complex problems is topical.

Upon construction of recognition and clusterization procedures, discrete optimization tools can be used in a number of cases. One example is synthesis of elementary classifiers in logical recognition procedures upon construction of a logical corrector based on elementary classifiers. Another example is recognition algorithms based on calculation of estimates. The degree of correctness of these algorithms depends on the choice of parameters characterizing the representative character of each of the learning objects and each of the reference sets. The problem is reduced to solving systems of linear inequalities and if the system is incompatible, to the choice of its maximal compatible subsystem, which in turn is equivalent to the maximization of the sum of Boolean variables in which the ith variable is equal to 1 if the ith inequality is valid, and 0 in the opposite case, which is known to be reducible to a covering problem.

The objective of this paper is the construction and study of genetic algorithms for the search for an approximate solution to the covering problem. Two genetic algorithms have been developed: an algorithm with binary problem representation and an algorithm with integer problem representation. In the first case, the binary vector is used to describe the matrix covering (specimens of a population), and in the second case, the integer vector. Both algorithms seek the minimal covering among irreducible coverings. The weight of the corresponding covering is used as the estimate of applicability of the solution (degree of fitness of the specimen).

Nonstandard crossing operators taking into account weights of the used columns and relative fitness of parents and mutation operators with a variable number of mutating genes are used. The number of mutating genes increases with the evolution of the population. An original method of optimizing the irreducible covering to obtain a covering with a lower weight is proposed.

The efficiency of the constructed algorithms is estimated using test problems from the electronic Beasley's OR Library. These problems consist of 65 sparse (with respect to the number of unities) matrices divided into 11 classes. For each problem, the algorithms were run ten times each and then evaluated with respect to three quantities: best solution, average deviation from the optimal solution, and frequency at which the optimal solution is obtained.

The results of testing demonstrated that at least one of the algorithms finds an optimal solution in 61 problems. In the four remaining problems, the weight of the best covering differs from that of the optimal solution by unity. Note that for the problems considered, the gradient algorithm constructs coverings whose weight is larger by 10–15% than that of the optimal solution.

Received February 26, 2008

2. STATEMENT OF THE PROBLEM

Let $L = (a_{ij})_{m \times n}$ be a Boolean matrix which does not contain rows of the form (0, 0, ..., 0). The set of columns H is called the covering if in the submatrix of the matrix L formed by this set of columns there is no zero row. A covering is called irreducible if in the case of elimination of any its elements this covering stops being a covering. Let each column j of the matrix L be related to the weight $c_j > 0$. It will be assumed that columns of matrix L are ordered by increasing weights. If column weights are equal, ordering is performed by the number of covered rows in decreasing order. The weight of the covering is the sum of weights of its elements. It is necessary to construct a covering with minimal weight.

Let us introduce the notation used hereinafter after the structure of genetic algorithms has been constructed.

$M = \{1, ..., m\}$ is the set of numbers of rows of matrix L;

$N = \{1, ..., n\}$ is the set of numbers of columns of matrix L;

$M_j = \{i \in M | a_{ij} = 1\}$ is the set of numbers of rows of matrix L covered by the column with the number $j \in N$;

$N_i = \{j \in N | a_{ij} = 1\}$ is the set of numbers of columns of the matrix L covering the row with the number $i \in M$;

$N_q^q \subseteq N_i$ is the set consisting of q columns of the matrix L with the lowest weight covering the row with the number $i \in M$. If $q \geq |N_i|$, we assume that $N_i^q = N_i$.

3. DESCRIPTION OF GENETIC ALGORITHMS

The operation of genetic algorithms resembles the evolution of biological organisms. Initially, the set of approximate solutions to the considered problem, called the population, is formed; elements of the population are called specimens. Each specimen is described by some set of values called chromosomes. The quality of the approximate solution is characterized by the degree of fitness, which usually depends on the functional minimized and they formulated optimization problem. In the course of operation of the genetic algorithm, the population is replenished using genetic crossing and mutation operators, which are analogous of the corresponding biological processes. In this case, replenishment of the population takes place in such a way that the least fitted specimens gradually leave the population. Thus, the general scheme of the genetic algorithm can be described as follows.

(1) The initial population $P = \{J_1, ..., J_s\}$ with the given volume N is formed according to some rule. The degree of fitness of specimens of the population is estimated.

(2) Two specimens, for example, J_u and J_v, called *parents*, are chosen from the population P.

(3) Specimen J, called the *child*, is formed from the specimens J_u, J_v using the crossing operator.

(4) The child undergoes mutation (random change) with probability p.

(5) The degree of fitness of the child is estimated.

(6) The child replaces one of the specimens of the population that is worse fit than the child.

(7) The condition of completion of the genetic algorithm is verified. If this condition is not satisfied, the algorithm returns to step 2.

In developing genetic algorithms, it is necessary to take into account a specific features of the considered problem; therefore, to improve the general algorithm efficiency, the following set of subproblems was solved.

(1) Choosing the problem representation.

(2) Choosing the crossing operators.

(3) Determining the set of admissible solutions.

(4) The method for choosing parent specimens.

3.1. Representation of the Problem

In developing the genetic algorithm, it is necessary to choose the method for representing specimens in the population. For this purpose, it is necessary to determine the set of objects Ω among which the solution will be sought using the genetic algorithm. The set Ω is called the *set of admissible solutions*. Obviously, the lower the power of the set Ω, the higher the probability of finding the optimal solution. In this regard, for the problem of covering by the set of admissible solutions, a set of irreducible coverings of the matrix is assumed.

The coding function $\mathbf{g}(H)$ transforming the set of columns H into the set of chromosomes is determined on the set of admissible solutions. In this paper two specimen representation methods are implemented.

(1) **Binary representation.** The coding function $\mathbf{g}(H)$ transforms the set of columns H into the binary vector $\mathbf{g} = (g_1, ..., g_n)$, where the components of the vector are determined as

$$g_i = \begin{cases} 1, & i \in H \\ 0, & i \notin H. \end{cases}$$

(2) **Integer representation.** The coding function $\mathbf{g}(H)$ transforms the set of columns H into the integer vector $\mathbf{g} = (g_1, ..., g_m)$, where $g_i \in N_i \cap H$. Note that in the case of the integer representation, unlike the binary one, the mapping is not one-to-one, and different integer vectors g_1 and g_2 can correspond to the same set of columns H.

In both cases, the weight corresponding to this specimen of the covering is used as the estimate of the degree of fitness of the specimen.

3.2. Formation of the Initial Population

The procedure of formation of the initial population P takes several steps.

Step 1. The vector **g** is constructed; each component g_i of this vector is randomly and uniformly chosen from {0, 1} in the case of binary representation and from N_i in the case of integer representation.

For binary representation, the components of vector g are chosen in such a way that the corresponding set of columns is the covering of matrix L. For integer representation, the set of columns corresponding to the integer vector $(g_1, ..., g_m)$ automatically represents a covering, since any row $i \in M$ is covered by the column with the number g_i.

Step 2. The set of columns H is constructed for vector **g**.

Step 3. The irreducible covering H' and $g' = g(H')$ are constructed for the covering H using the following algorithm. It is assumed that $H' = H$. Elements $j \in H'$ are considered in order of decreasing weights c_j for which it is verified whether the set of columns $H'\{j\}$ represents a covering. If it is a covering, we assume that $H' = H'\backslash\{j\}$.

Step 4. The degree of fitness of the specimen (H', g') is estimated.

Step 5. If the population P does not contain the specimen (H', g'), it is added to P.

Step 6. If the number of generated specimens is smaller than the chosen population size N, go to step 1. In the opposite case, the formation of the population is completed.

3.3. Choice of Parent Specimens

In this paper, the method of proportional choice is implemented for choosing parent specimens. According to this method, each specimen is assigned with the choice probability according to the value of the fitness function and the parents are chosen based on these probabilities. The choice probability for each specimen of the population is calculated using the formula

$$p_i = \frac{1/f_i}{\sum_{i=1}^{N} 1/f_i}, \quad (1)$$

where f_i is the degree of fitness of the ith specimen in the population and N is the population size.

Since the degree of fitness of the specimen in the population is the weight of the corresponding covering, for large weights of the optimal solution of the covering problem, the probabilities of choosing parents are approximately equal. For example, for $N = 3$, $f_{\text{opt}} = 500$ and $f_1 = 501, f_2 = 510, f_3 = 520$, $p_1 \approx p_2 \approx p_3 \approx 0.33$; i.e., each specimen is chosen with equal probability in spite of the closeness to the optimal solution.

In order to solve this problem, we propose using the relative degree of fitness calculated using the formula

$$f_i^r = f_i - f_{\text{opt}} + 1.$$

Since the value of the optimal degree of fitness is unknown, the following quantity is used as the relative degree of fitness:

$$f_i^r = f_i - \min_{j \in [1, ..., N]} f_j + 1. \quad (2)$$

3.4 Crossing Operators

To form new specimens in genetic algorithms, crossing operators are used; these operators, based on genes of parent specimens, form the child specimen. Upon creation of a genetic algorithm, mainly crossing operators are used that randomly mix the genes of parents.

In this paper, crossing operators taking into account the structure and relative degrees of fitness of parents are implemented for specimens with binary and integer representation.

Let f_1^r and f_2^r be the relative degrees of fitness of the specimens $g^1 = (g_1^1, ..., g_n^1)$ and $g^2 = (g_1^2, g_2^2, ..., g_i^2)$, respectively. For all i, $1 \le i \le n$, she maybe it is a little bit of it is that we stood there one day it was due to the least you could do with you would be shifted to debate it with you there was to shift as you would want us to give approval of the key we determine g_i as follows:

if $g_i^1 = g_i^2$, then $g_i = g_i^1 = g_i^2$;

if $g_i^1 \ne g_i^2$, then $g_i = g_i^1$ with the probability

$$p = \frac{f_2^r}{f_1^r + f_2^r} \text{ in the case of binary representation of specimens,} \quad (4)$$

$$p = \frac{c_{g_i^2} f_2^r}{c_{g_i^1} f_1^r + c_{g_i^2} f_2^r} \text{ in the case of integer representation of specimens,} \quad (5)$$

and $g_i = g_i^2$ with the probability $1 - p$.

3.5. Mutation Operator

During operation of the algorithm, the convergence of solutions to some local minimum can be observed. This may take place if there is a rather large number with an approximately similar genotype in the population. The crossing operator for such child specimens gives an object similar to the initial specimens, i.e., the object near some local minimum. In order to introduce a variety of types into the population, a mutation operator is used; this operator changes some set of genes in a specimen. Since the mutation operator should have an especially strong impact on the object in the case of process convergence, we propose increasing the number of mutated genes k(t) with time according to the formula

$$\mathrm{k}(t) = K\left(1 - \frac{1}{Ct+1}\right),$$

where K is the number of mutated genes at the last step of the algorithm and t is the step number. Quantity C characterizes the rate of variation of the number of mutating genes.

The efficiency of the mutation operator also depends on the set of genes which can mutate. It can be easily noticed that the probability of columns with large weights being included in the optimal solution is rather low. Therefore, for mutation it is necessary to choose the set of columns which have a large chance of being included in the optimal solution. This set can be determined, for example, using the following formula:

$$N^{\mathrm{mut}} = \bigcup_{i=1}^{m} N_i^q, \tag{6}$$

where, as was said above, N_i^q is the set consisting of q columns of matrix L with the lowest weights.

Thus, the probability of inverting the gene i for binary representation is

$$p_i = \begin{cases} \dfrac{\mathrm{k}(t)}{|N^{\mathrm{mut}}|}, & i \in N_i^q \\ 0, & i \notin N_i^q. \end{cases} \tag{7}$$

For integer representation, each component g_i is replaced by a randomly chosen value from the set $N_i^q \setminus \{g_i\}$ with the probability

$$p_i = \frac{\mathrm{k}(t)}{n}. \tag{8}$$

3.6. Recovery of Solution Admissibility

If the mixing and mutation operators are applied to specimens of the population, the admissibility of the solution may be violated (i.e., the obtained set of columns may not be the irreducible covering of matrix L). For binary representation, the case is possible when the obtained set of columns H is not even a covering.

To recover solution admissibility, the following heuristic algorithm is proposed, which is simultaneously the local optimization step for increasing the total efficiency of the algorithm.

Let (H, g) be a specimen of the population which does not satisfy the admissibility condition;

M' be the set of numbers of all rows of the matrix L uncovered by columns from H;

w_i be the number of columns from H covering the row with the number i, $i = 1, 2, ..., m$.

The procedure of recovery of solution admissibility can be described as follows.

Step 1. Initialize $w_i = |H \cap N_i|$, $i = 1, 2, ..., m$.

Step 2. Initialize $M' = \{i | w_i = 0, i = 1, 2, ..., m\}$.

Step 3. For each element $i \in M'$ in order of increasing find the column $j \in N_i$ which minimizes the functional $\dfrac{c_j}{|M' \cap M_j|}$. Add j in H, $w_i = w_i + 1$, $i \in M_j$, $M' = M' \setminus M_j$.

Step 4. For each $j \in H$ in order of decreasing weights if $w_i > 1$ for all $i \in M_j$, then

$$H = H \setminus \{j\}, \quad w_i = w_i - 1, \quad i \in M_j.$$

In the proposed algorithm, steps 1 and 2 determine the uncovered rows of the matrix. At step 3, the covering of the matrix is constructed in such a way that columns with the smallest ratio of the weight and number of covered rows are added first. At step 4, excessive columns are eliminated from the covering in such a way that the columns with the largest weights are eliminated first; i.e., at steps 3 and 4, a greedy heuristics of the search for an optimal solution is used.

It was noted above that for integer representation of the problem for any vector $\mathbf{g} = (g_1, ..., g_m)$, where $g_i \in N_i$, the corresponding set of columns H automatically represents a covering of matrix L. Thus, for integer representation, steps 2 and 3 of the proposed algorithms are unnecessary.

The admissibility recovery procedure is augmented by optimization of the weight of the obtained admissible solution. It is based on the search for such columns j in the set $N \setminus H$ that, upon recovery of admissibility of the specimen with the set of columns $H \cup j$, excessive columns $j_1, j_2, ..., j_k$, where $j_i \neq j$, $i = 1, ..., k$, are eliminated; for these columns the following relation is satisfied:

$$\sum_{i=1}^{k} c_{j_i} - c_j > 0. \tag{9}$$

We denote this set of columns by $\mathbf{r}(H)$ and the quantity on the right-hand side of inequality (9) by $\omega(j, H)$. Let us describe the procedure of optimization of the admissible solution.

Let (H, g) be the specimen of the population satisfying the admissibility condition.

Step 1. Form the set $\mathbf{r}(H)$. If it is empty, complete the algorithm.

Step 2. Find $j = \arg\max\{\omega(j, H) | j \in \mathbf{r}(H) \cup N^{\text{mut}}\}$, where N^{mut} is the set of columns determined according to (6).

Step 3. Assume $H = H \cup j$.

Step 4. Recover the admissibility of solution (H, g) by eliminating excessive columns $j_1, j_2, \ldots, j_k$, where $j_i \neq j$, $i = 1, \ldots, k$, for which (9) is satisfied. Go to step 1.

3.7. Model of Population Change

The renewal of the current population by new "child" specimens can be performed in different ways. The method realized in this paper is called the sequential replacement method. According to this method, the child replaces one specimen with a higher degree of fitness randomly chosen from the population.

Another frequently used method is that in which a set of "child" specimens is created which partially or completely replaces the population of "parent" specimens.

The advantage of the sequential replacement method is that better solutions found by the genetic algorithm are always left in the population, and the created specimens can immediately participate in formation of "child" specimens.

As the population changes according to the rule of sequential replacement it is necessary to avoid repeated appearance of a specimen in the population. If this condition is rejected, it is possible that a large part of the population will consist of similar specimens, and as a consequence, premature convergence of the genetic algorithm to the local minimum will be observed.

3.8. Choice of Population Size

The population size should be chosen in such a way that the obtained initial population is sufficiently diverse for the genetic algorithm to finally converge to the global minimum, i.e., the minimal covering. Let us illustrate the choice of the population size in the genetic algorithm using figure, which shows the plot of deviation of the solution, averaged over five test problems, found by the genetic algorithm from the optimal solution,

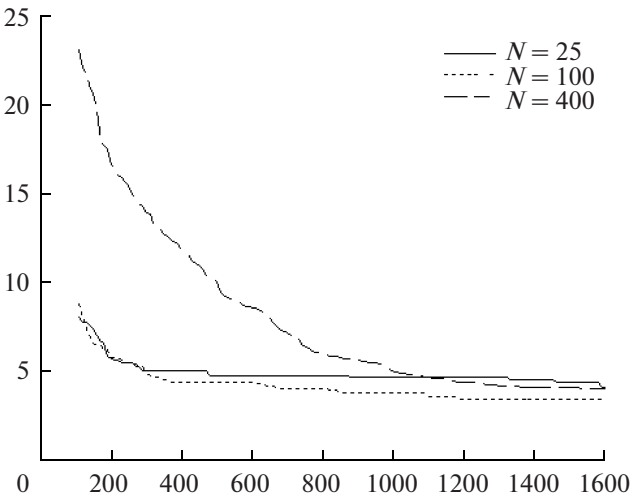

Figure.

$$\sigma = \frac{1}{5}\sum_{i=1}^{5}\frac{f_i - f_i^{\text{opt}}}{f_i^{\text{opt}}} \times 100\%;$$

here, f_i^{opt} is the best result for the ith problem and f_i is the weight of the best covering found by the algorithm for each of the five test problems with a size of 1000×10000. The average deviation σ characterizes the quality of the solution found by the algorithm.

Figure shows the dependence of the average deviation σ on the number of steps performed by the algorithm. It can be seen from the plot that if the population size is sufficiently large, for example, $N = 400$, the population is renewed too slowly and the convergence to the optimal solution is slow. If the population size is insufficient, for example, $N = 25$, premature convergence to the local minimum is observed. Thus, the optimal population size is $N = 100$.

4. DESCRIPTION OF THW GABINARY AND GANONBINARY ALGORITHMS

4.1 Schematic Diagram of the GABinary Algorithm

Step 1. Create the initial population of $N = 100$ specimens according to the algorithm of formation of the initial population for binary representation of the problem.

Step 2. Choose two specimens (H^1, g^1) and (H^2, g^2) from the population using the proportional choice method with relative degree of fitness (5).

Step 3. Apply crossing operator (3) with relative fitness function (5) to specimens (H^1, g^1) and (H^2, g^2) for creation of the child specimen (H, g).

Table 1. Characteristics of test problems

Class of problem	Number of rows in matrix	Number of columns in matrix	Number of problems in class	Fraction of unities in matrix, %
4	200	1000	10	2
5	200	2000	10	2
6	200	1000	5	5
A	300	3000	5	2
B	300	3000	5	5
C	400	4000	5	2
D	400	4000	5	5
E	500	5000	5	10
F	500	5000	5	20
G	1000	10000	5	2
H	1000	10000	5	5

Step 4. Apply the mutation operator with the mutation probability determined in (7).

Step 5. Apply the algorithm of recovery of the solution (H, g) admissibility for the binary representation. The result of operation of the algorithm is the specimen (H', g'), where H' is the irreducible covering. If (H', g') coincides with any specimen of the population, go to step 2.

Step 6. Calculate the degree of fitness f of specimen (H', g').

Step 7. If the population does not contain specimen (H', g'), renew the population (include specimen (H', g') according to the sequential replacement algorithm).

Step 8. If the population was renewed at step 7, increase the number of steps by 1. If the number of steps is smaller than $N_{\max} = 100000$, go to step 2. In the opposite case, complete the algorithm.

4.2. Schematic Diagram of Genetic GANonBinary Algorithm

Step 1. Create the initial population of $N = 100$ specimens according to the algorithm of formation of the initial population for integer representation of the problem.

Step 2. Choose two specimens (H^1, g^1) and (H^2, g^2) from the population using the proportional choice method with relative degree of fitness (5).

Step 3. Apply crossing operator (4) with relative fitness function (5) to specimens (H^1, g^1) and (H^2, g^2) for creation of the child specimen (H, g).

Step 4. Apply the mutation operator with the mutation probability determined in (8).

Step 5. Apply the algorithm of recovery of solution (H, g) admissibility for the integer representation. The result of operation of the algorithm is specimen (H', g'), where H' is the irreducible covering. If (H', g') coincides with any specimen of the population, go to step 2.

Step 6. Calculate the degree of fitness f of specimen (H', g').

Step 7. If the population does not contain specimen (H', g'), renew the population (include specimen (H', g') according to the sequential replacement algorithm).

Step 8. If the population was renewed at step 7, increase the number of steps by 1. If the number of stems is smaller than $N_{\max} = 100000$, go to step 2. In the opposite case, complete the algorithm.

5. NUMERICAL RESULTS

The efficiency of the constructed algorithms was evaluated using 65 test problems from the electronic Beasley's OR Library. These problems consist of sparse (with respect to the number of unities) matrices divided into 11 classes (the characteristics of these classes are given in Table 1). The algorithms were run ten times for each problem and then evaluated using three quantities: the best solution, the average deviation from the optimal solution, and the frequency at which the optimal solution is obtained.

The numerical results are given in Table 2. The following notation is used for columns of this table:

f^{opt} is the column with the optimal solutions for each problem; Greedy is the column with solutions found by the gradient algorithm;

$f_{\min}$ is the minimal value of the specimen fitness function found by the algorithm as the optimal solution;

$f_{\max}$ is the maximal value of the fitness function found by the algorithm;

σ is the average relative deviation of the solutions found by the algorithm from the optimal in the series of ten tests (calculated using the formula); and

ν is the frequency of finding the best result by the algorithm in ten tests.

It can be easily seen that for 61 problems at least one algorithm found the optimal solution. For the four remaining problems (F.5, G.2, G.3, H.1), the smallest length of the covering found by the algorithms differs from the length of the minimal covering by unity.

Table 2. Numerical results

Number of problem	f^{opt}	Greedy	GABinary				GANonBinary			
			f_{min}	f_{max}	σ (%)	ν	f_{min}	f_{max}	σ (%)	ν
4.1	429	463	429	432	0.28	0.2	429	429	0.0	1.0
4.2	512	582	512	512	0.0	1.0	512	512	0.0	1.0
4.3	516	598	516	516	0.0	1.0	516	516	0.0	1.0
4.4	494	548	494	494	0.0	1.0	494	494	0.0	1.0
4.5	512	577	512	512	0.0	1.0	512	512	0.0	1.0
4.6	560	615	560	560	0.0	1.0	560	560	0.0	1.0
4.7	430	476	430	432	0.09	0.8	430	430	0.0	1.0
4.8	492	533	492	492	0.0	1.0	492	492	0.0	1.0
4.9	641	747	641	650	0.27	0.7	641	644	0.09	0.8
4.10	514	556	514	514	0.0	1.0	514	514	0.0	1.0
5.1	253	289	253	260	0.28	0.9	253	254	0.08	0.8
5.2	302	348	302	305	0.56	0.4	302	302	0.0	1.0
5.3	226	246	228	228	0.88	0.0	226	228	0.18	0.8
5.4	242	265	242	242	0.0	1.0	242	243	0.17	0.6
5.5	211	236	211	211	0.0	1.0	211	211	0.0	1.0
5.6	213	251	213	213	0.0	1.0	213	213	0.0	1.0
5.7	293	326	293	294	0.07	0.8	293	294	0.20	0.4
5.8	288	323	288	289	0.21	0.4	288	289	0.07	0.8
5.9	279	312	279	279	0.0	1.0	279	279	0.0	1.0
5.10	265	293	265	265	0.0	1.0	265	265	0.0	1.0
6.1	138	159	138	138	0.0	1.0	138	138	0.0	1.0
6.2	146	170	146	147	0.21	0.7	146	146	0.0	1.0
6.3	145	161	145	145	0.0	1.0	145	145	0.0	1.0
6.4	131	149	131	131	0.0	1.0	131	131	0.0	1.0
6.5	161	196	161	165	0.19	0.9	161	164	0.56	0.7
A.l	253	288	253	255	0.24	0.6	253	254	0.28	0.3
A.2	252	284	252	252	0.0	1.0	252	252	0.0	1.0
A.3	232	270	232	233	0.09	0.8	232	233	0.30	0.3
A.4	234	278	234	234	0.0	1.0	234	234	0.0	1.0
A.5	236	271	236	237	0.08	0.8	236	237	0.08	0.8
B.l	69	77	69	69	0.0	1.0	69	69	0.0	1.0
B.2	76	86	76	76	0.0	1.0	76	76	0.0	1.0
B.3	80	89	80	80	0.0	1.0	80	80	0.0	1.0
B.4	79	87	79	79	0.0	1.0	79	79	0.0	1.0
B.5	72	78	72	72	0.0	1.0	72	72	0.0	1.0
C.l	227	258	227	229	0.22	0.6	227	227	0.0	1.0
C.2	219	258	219	221	0.09	0.9	219	219	0.0	1.0
C.3	243	276	243	247	0.45	0.4	243	244	0.29	0.4
C.4	219	257	219	221	0.09	0.9	219	219	0.0	1.0
C.5	215	233	215	216	0.05	0.9	215	215	0.0	1.0
D.1	60	74	60	60	0.0	1.0	60	60	0.0	1.0
D.2	66	74	66	66	0.0	1.0	66	67	0.60	0.6
D.3	72	83	72	72	0.0	1.0	72	72	0.0	1.0
D.4	62	71	62	62	0.0	1.0	62	62	0.0	1.0

Table 2. (Contd.)

Number of problem	f^{opt}	Greedy	GABinary				GANonBinary			
			f_{min}	f_{max}	σ (%)	ν	f_{min}	f_{max}	σ (%)	ν
D.5	61	69	61	61	0.0	1.0	61	61	0.0	1.0
E.1	29	30	29	29	0.0	1.0	29	29	0.0	1.0
E.2	30	36	30	31	1.33	0.6	30	30	0.0	1.0
E.3	27	31	27	28	2.59	0.3	27	27	0.0	1.0
E.4	28	32	28	28	0.0	0.0	28	28	0.0	1.0
E.5	28	33	28	28	0.0	0.0	28	28	0.0	1.0
F.1	14	16	14	14	0.0	1.0	14	14	0.0	1.0
F.2	15	16	15	15	0.0	1.0	15	15	0.0	1.0
F.3	14	17	14	14	0.0	1.0	14	14	0.0	1.0
F.4	14	17	14	14	0.0	1.0	14	14	0.0	1.0
F.5	13	16	14	14	7.7	1.0	14	14	7.7	1.0
G.1	176	203	176	178	0.45	0.4	176	178	0.40	0.6
G.2	154	182	155	157	0.39	0.5	155	156	0.78	0.7
G.3	166	192	168	169	1.69	0.2	167	168	1.26	0.3
G.4	168	191	170	172	1.61	0.5	168	170	0.83	0.4
G.5	168	194	168	169	0.12	0.7	168	169	0.12	0.8
H.1	63	76	64	64	1.59	1.0	64	64	1.59	1.0
H.2	63	74	63	64	1.43	0.1	64	64	1.59	1.0
H.3	59	65	59	61	0.6	1.36	59	60	0.85	0.5
H.4	58	69	58	60	1.89	0.1	59	59	1.72	1.0
H.5	55	63	55	55	0.0	1.0	55	55	0.0	1.0

6. CONCLUSIONS

Two genetic algorithms were developed for the search for the solution of minimal covering of a Boolean matrix which use nonstandard crossing, mutation, and solution admissibility recovery operators. The efficiency of the constructed algorithms was evaluated using 65 problems of large size and complex structure. Optimal solutions were found for 61 problems, and the length of the coverings found in the remaining 4 problems differs from the minimal one by unity.

ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, project no. 07-01-00516 and the Council on Presidential Grants for State Support of Leading Scientific Schools, grant no. NSh-5294.2008.1.

REFERENCES

1. R. M. Sotnezov, "Genetic Algorithms in Problems of Discrete Optimization and Recognition," in *Proc. of International Conf. on "Pattern Recognition and Image Analysis: New Information Technologies"* (Nizhni Novgorod, 2008), Vol. 2, pp. 173–175.
2. R. M. Sotnezov, *Genetic Algorithms in the Covering Problem. Collection of Thesis's of the Best Graduation Papers of 2008 Year* (Izdatel'skii Otdel Fakul'teta VMiK MGU, Moscow, 2008), pp. 73–74 [in Russian].
3. D. I. Batishchev, V. E. Kostyukov, N. V. Starostin, and A. I. Smirnov, *Population–Genetic Approach to the Covering Problem. Handbook* (Izd–vo NNGU, Nizhni Novgorod, 2004) [in Russian].
4. M. Kh. Nguen, *The Use of the Genetic Algorithm for the Problem of Production Scheduling. Dynamic of Heterogeneous Systems* (KomKniga, Moscow, 2007), Issue 11, pp. 162–169 [in Russian].
5. J. E. Beasley and P. C. Chu, "A Genetic Algorithm for the Set–Covering Problem," European Journal of Operational Research **94**, 392–404 (1996).
6. J. E. Beasley and K. Jornsten, "Enhancing an Algorithm for Set Covering Problems," European Journal of Operating Research **58**, 293–300 (1992).
7. L. W. Jacobs and M. J. Brusco, "A Simulated Annealing–Based Heuristic for the Set Covering Problem," in *Proc. "Operations Management and Information Systems Department," Northern Illinois University, Deklab, IL 60115, USA, 1993* (Northern Illinois University, 1993).
8. M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP—Completeness* (Freeman, 1979).

9. J. E. Beasley, "OR–Library: Distribution Test Problems by Electronic Mail," Journal of the Operation Research Society **41**, 1069–1072 (1990).

10. A. Capara et al., "Algorithms for Railway Crew Management," Mathematical Programming **79**, 125–141 (1997).

11. S. Ceria et al., "Set Covering Problem," in *Annotated Bibliographies in Combinatorial Optimization*, Ed. by In Dell'Amico, F. Maffioli, and S. Martello (Wiley and Sons, 1997), pp. 415–428.

12. A. Capara, M. Fischetti, and P. Toth, "Algorithms for the Set Covering Problem," in *Working Paper, DEIS* (University of Bologna, 1998).

13. A. Capara, M. Fischetti, and P. Toth, "A Heuristic Method for the Set Covering Problem," Operation Research **47**, 730–743 (1999).

14. P. Galinier and A. Heztz, "Solution Techniques for the Large Set Covering Problem," Discrete Applied Mathematics **135** (3) (2007).

15. G. Lan, G. W. DePuy, and G. E. Whitehouse, "An Effective and Simple Heuristic for the Set Covering Problem," European Journal of Operating Research **176** (3), 1387–1403 (2007).

Roman M. Sotnezov was born on December 3, 1985 in Cheboksary. In 2003 entered the Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University; graduated from the university in 2008 with the speciality "Applied Mathematics and Informatics." After graduating the university, entered the post-graduate course of the Faculty of Computational Mathematics and Cybernetics. Scientific interests: discrete mathematics, image recognition, algorithms and data structures. 1 publication.