A HYBRID ALGORITHM FOR SET COVERING PROBLEM[1]

ANTON V. EREMEEV, ALEXANDER A. KOLOKOLOV, LIDIA A. ZAOZERSKAYA

Omsk Branch of Sobolev Institute of Mathematics Siberian Branch of Russian Academy of Sciences, 13 Pevtsov st. 644099, Omsk, Russia
eremeev@iitam.omsk.net.ru, kolo@iitam.omsk.net.ru, zaozer@iitam.omsk.net.ru

Abstract. In this paper we propose a hybrid algorithm for the set covering problem (SCP), which combines L-class enumeration, genetic algorithm (GA) and Lagrangean heuristic. The lexicographical enumeration of L-classes in the linear relaxation of SCP ensures optimality of the resulting solution. Both heuristics are used for obtaining the initial approximate solution. The linear subproblems in L-class enumeration are also tested by means of Lagrangean relaxation. The results of computation experiments appear to be promising.

Key Words. Set covering problem, genetic algorithm, L-class enumeration

1. INTRODUCTION

The set covering problem (SCP) can be stated as follows. *Consider a set* $M = \{1, \ldots, m\}$ *and the subsets* $M_j \subseteq M$, *where* $j \in N = \{1, ..., n\}$. *A subset* $J \subseteq N$ *is a cover of* M *if* $\bigcup_{j \in J} M_j = M$. *For each* M_j *a positive cost* c_j *is assigned. The SCP is to find a cover of minimum summary cost.* The problem where $c_1 = c_2 = ... = c_n$ will be called the *unicost* SCP. In this paper we suppose that the costs are integer, although in case of real costs just a few changes in the algorithms would be required. Sometimes it is more convenient to consider SCP as an integer linear programming problem:

$$\min\{cx : Ax \geq e, x \in \{0,1\}^n\}, \qquad (1)$$

where A is an $m \times n$ matrix of 0s and 1s, c is an n-vector of costs, e is the m-vector of 1s, and $a_{ij} = 1$, if and only if $i \in M_j$. Using this formulation one can view SCP as a problem of optimal covering of rows by the columns in matrix A. In this paper we assume that the columns are ordered according to nonincreasing values of c_j-s.

Amoung the numerous applications of the SCP are the crew scheduling [2, **?**], emergency facilities allocation, production flow-lines optimization [5] etc. Different algorithms have been proposed for exact solving of SCP using branch and bound approach [2, 6, 17], cutting planes, L-class enumeration (LCE) [16, 18] and other techniques. However, SCP is an NP-hard problem and usually application of exact algorithms to the large-scale instances is very time-consuming. A number of heuristic algorithms are developed for approximate solving the large scale problems within relatively short running time. The solutions of good quality may be obtained using Lagrangean heuristics [7], neural networks [14] or meta-heuristics such as GA [4, 11] or ant colony algorithms [1]. Most of the successful versions of exact algorithms combine the exact techniques with heuristic methods (see e.g. [2, 6]).

2. L - CLASS ENUMERATION FOR SET COVERING PROBLEM

The papers [15, 18] and some other works are devoted to development of an approach for investigation and solving of integer programming problems, using some special (*regular*) partitions of space R^n, in particular, L*-partition.* In the framework of this approach the L-class enumeration method was proposed for the general integer linear programming problems. Here we will discuss this method in the way it is applied to the SCP.

Let $\Omega = \{x \in R^n : Ax \geq e, 0 \leq x \leq e\}$ be the relaxation set of SCP. The LCE method works

[1]The research was supported in part by RFBR Grant 97-01-00771, and INTAS Grant 96-0820.

on the principle that the polyhedron Ω may be split into lexicographically ordered subsets and the optimal solution can be found by checking some of these subsets. For detailed description of the LCE method the following definitions are needed.

We say that points $x, y \in R^n$ $(x \succ y)$ belong to the same class of L-partition if no integer vector z exists that $x \succeq z \succeq y$. Here $\succ, \succeq$ are the symbols of lexicographical order. Each point $z \in Z^n$ forms a separate class of partition; other classes that contain only noninteger points are called *fractional*. We denote the factor - set, induced by L-partition for a set $X \subset R^n$ as X/L. The elements of X/L are called L-classes. This partition has some important properties, for example,

1) if X is bounded, then X/L is finite;

2) for X/L a linear order may be introduced: for any nonempty $V', V'' \in X/L$ we write $V' \succ V''$ if $x' \succ x''$ for any $x' \in V'$ and $x'' \in V''$.

Let Ω_0 be a set of all optimal solutions for (1), $z^* = \text{lexmin}\, \Omega_0$. The LCE algorithm generates a sequence of current points $\sigma = \{x(t)\}$ from Ω, $t = 1, \ldots, \theta_L$ with the following properties:

a) $x(t) \prec x(t+1)$, $t = 1, \ldots, \theta_L - 1$,

b) all points $x(t)$ belong to different L-classes;

c) σ contains an integer subsequence $\sigma' = \{x(t_k)\}$, $k = 1, \ldots, q$, such that $z^* \in \sigma'$ and if $k = 1, \ldots, q$, and $t > t_k$, then $(c, x(t)) < (c, x(t_k))$.

It is not difficult to see, that $|\sigma| \leq |\Omega/L|$, and in case of SCP this sequence is proved to be finite [15]. The current points $x(t)$ are constructed by solving special linear programming problems.

The aim of the LCE is to find the integer optimum, moving along the set $\Omega/L = \{V_1, ..., V_p\}$. It is not necessary to visit all of the L-classes: an additional condition (record inequality) such as $(c, x) \leq (c, \zeta) - 1$, where $\zeta = x(t_k)$ is the best integer solution found before the current step $t > t_k$,, would exclude some unnecessary L-classes. The value of the goal function $r = (c, \zeta)$ will be called a record on the current step. Before the LCE is started, the record is assumed to be $+\infty$, or it may be determined by means of some heuristic algorithm (for example, in the LCE algorithm described below the initial admissible solution and the corresponding record value are obtained by the well-known Chvatal greedy algorithm [8]). Every time when a new integer point is found, the LCE excludes it by the record inequality. The LCE can be started from any point $x(0) \in \Omega$. The process stops if it is impossible to find the next suitable L-class. After this ζ is the optimal solution to the SCP, given that $x(0) \preceq \Omega_0$.

Let ζ^0 be the approximate solution found by the greedy heuristic on the preliminary stage. On the next stage L-class enumeration is began accourding with lexicographically increasing order, starting from the point lexmin Ω. Due to *alternating L-structure* property of SCP (see e.g. [15]), this starting point is an integer vector. The L-class enumeration algorithm [18] for the SCP has the following outline.

The general scheme of LCE

Step 0. Find an approximate solution $\zeta^0 \in \mathbf{Z}^n$.
0.1 Find $x' = \text{lexmin}\, \Omega$.
0.2 Set $r := \min\{(c, x'), (c, \zeta^0)\}$.
0.3 Set $p := \max\{j : x'_j = 1, j = 1, ..., n-1\}$.

Step 1. Find $\varphi = \max\{j : x'_j = 0, j = 1, ..., p-1\}$.
1.1 If such φ does not exist, go to step 4.

Step 2. Set $x'' := x'$. Solve the linear subproblem:

$$x' = \text{lexmin}\{x \in \Omega : (c, x) \leq r - 1, \quad x_1 = x''_1, ..., x_{\varphi-1} = x''_{\varphi-1}, x_\varphi = 1\}. \tag{2}$$

2.1 If the subproblem (2) has no solutions, and $\varphi = 1$, go to step 4.
2.2 If the subproblem (2) has no solutions, and $\varphi > 1$, then set $p := \varphi$ and go to step 1.
2.3 If $x' \in \mathbf{Z}^n$, and $r > (c, x')$, then set $r := (c, x')$.
2.4 If $x' \in \mathbf{Z}^n$, then set $p := \max\{j : x'_j = 1, j \leq n-1\}$, and go to step 1.

Step 3. Find $\varphi := \min\{j : x'_j \neq \lfloor x'_j \rfloor, j = 1, ..., n\}$, and go to step 2.

Step 4. The enumeration is finished: the best obtained integer solution is optimal.

The linear subproblems on step 2 we can solve, for example, using the lexicographical dual simplex method. Often it turnes out that these subproblems have no feasible solutions. To save the computation time the linear subproblems are examined by the following group testing heuristic, which allows to test more than one L-class at a time.

The group testing heuristic is applied on step 2 in case an L-class where $x_\varphi = 1$, and $x_j = x''_j$, for all $j = 1, \ldots, \varphi - 1$ is not found. Then we solve the following subproblem:

$$x' = \text{lexmin}\{x \in \Omega : (c, x) \leq r - 1, \quad x_1 = x''_1, ..., x_{j_0-1} = x''_{j_0-1}, \sum_{j=j_0}^{j_0+n_0-1} x_j \geq 1\}, \tag{3}$$

where j_0 and $n_0 > 0$ are such that:

$$j_0 = \min\{j < \varphi : x''_j = x''_{j+1} = ... = x''_{j+n_0-1} = 0, \text{ and no } k \in [j + n_0, \varphi - 1] \text{ exists that } x''_k = 0\}.$$

If in the set $\{x \in \Omega : (c, x) \leq r - 1\}$ there is no such L-class $V \succeq x''$ that for all $x \in V$ we have $x_j = x''_j, j = 1, ..., j_0 - 1$, then problem (3) has no feasible solutions. Otherwise the next L-class would be found, and the process would continue from it.

Before solving the problem (3) we first check the existance of admissible solutions to it. Let us consider the following supplementary linear subproblem: find

$$\sum_{j=1}^{n} c_j x_j \to \min \tag{4}$$

subject to

$$x \in \Omega, \sum_{j=j_0}^{j_0+n_0-1} x_j \geq 1, \tag{5}$$

$$x_j = x''_j, j = 1, ..., j_0 - 1. \tag{6}$$

If the optimal goal function value for (4) - (6) exceeds $r-1$, then the problem (3) has no solutions. In order to bound the optimum of (4) - (6) from below, one can use an approximate solution to the dual problem. This solution may be obtained for example by the knapsack greedy algorithm. The LCE always finds the optimum of SCP and finishes enumeration after visiting not more than $|\Omega/L|$ L-classes.

3. GENETIC ALGORITHM

In our hybrid algorithm one of the heuristics applied in for the search of initial approximate solutions is the GA [11]. The genetic algorithms were proposed in the works of J.Holland and further developed by D.Goldberg and many other authors (see e.g.[9, 13]). These algorithms are based on modelling of the selective breeding in nature and of the random changes which take place during mutation and crossover. The GA *population* consists of *individuals*, and each individual is a pair of *genotype* g and *phenotype* $x(g)$ corresponding to a search point in the space of solutions D. Here g is a fixed length string of symbols (called *genes*) from alphabet $\mathbf{A}$. The function $x(g)$ maps g to its phenotype $x(g) \in D$, thus defining a *representation* of solutions in GA. The search is guided by the results of evaluation of the *fitness* function $\Phi(g)$ for each genotype in population. $\Phi(g)$ usually depends monotonically on the goal function for feasible solutions and it may also be used as a penalty function for infeasible ones.

The genotypes of the initial population are randomly generated according to some a priori defined probability distribution. The GA discussed here implements the *steady-state* reproductive strategy [9], and a single new individual is generated at each iteration. The genotype of a new individual replaces the least fit one in the population, thus the population size s remains constant during the run of GA. Usually the stopping criterion is the limit on the number of iterations t_{max}. The best solution found is returned as the final result when GA stops.

Each new genotype (offspring) is constructed from a pair of parents chosen out of the current population by a probabilistic *selection* operator. Then the *crossover* operator replaces some genes of one parent genotype with genes of the other. The resulting string undergoes *mutation*, where a randomly chosen subset of genes is substituted with random symbols from $\mathbf{A}$. In this paper we consider a GA where a gene is mutated with a fixed probability p_m. Usually p_m is chosen empirically for the specific class of optimization problems.

3.1. General Scheme of the Non-Binary Genetic Algorithm

Let's denote the set of columns that cover the row i by $N_i = \{j \in N : a_{ij} = 1\}$. The non-binary GA (NBGA) described here is based on a non-binary representation where the genotype consists of m genes $g^{(1)}, g^{(2)}, \ldots, g^{(m)}$, such that $g^{(i)} \in N_i$, $i = 1, 2, \ldots, m$. Here each gene contains a column index that is assigned to cover the row. In this representation x maps a genotype g to the phenotype $x(g) \in \{0, 1\}^n$, where ones correspond to the columns present in genes of g. Obviously, it is a feasible solution to the problem (1) and no repair operator is needed.

For effective implementation of GA with non-binary representation an additional operator is required to eliminate the redundant columns from the solution after crossover and mutation. For this local improvement we use greedy heuristics that find approximate solutions to a corresponding reduced version P_g of the given SCP. The reduced problem P_g has a matrix that consists of the columns of A represented in genes of g. An improved genotype is added to the population only if there are no individuals with the same phenotype in it yet.

In genotypes of the initial population each gene $g_i, i = 1, ..., m$ is uniformly distributed over the set N_i. Let's consider the current population on iteration t as a vector of genotypes of its individuals $G = (g_1, g_2, \ldots, g_s)$. The fitness function on iteration t is defined as $\Phi_t(g) = cx(g_{l(t)}) - cx(g) + c_n$, where $l(t)$ is the index of the individual of the largest cover cost in the population on iteration t. The selection operator implements the proportional selection scheme, and

the probability to choose the k-th individual ($k = 1, 2, \ldots, s$) equals $p(g_k) = \Phi_t(g_k) / \sum_{l=1}^{s} \Phi_t(g_{l(t)})$.

The NBGA

1. While the initial population is not complete do
   1.1. Generate a random genotype g.
   1.2. Apply the column elimination procedure *Prime* to g and add g to population.
2. For t:=1 to t_{max} do
   2.1. Choose the parent genotypes g_u, g_v by the proportional selection.
   2.2. Produce an offspring g from g_u and g_v using crossover operator.
   2.3. Mutate each gene of g with probability p_m.
   2.4. Let g' be the best of results of procedures *Greedy* and *Dual Greedy* applied to g.
   2.5. If there are no individuals in G with phenotype $x(g')$, then substitute the least fit genotype $g_{l(t)}$ in population by g'. Otherwise substitute $g_{l(t)}$ by g.

The operators of crossover, mutation and the local improvement procedures *Prime*, *Greedy* and *Dual Greedy* will be described in the following sections.

3.2. Crossover and Mutation

Our crossover operator (called *LP-crossover*) is designed to find the best possible combination of the given parent genotypes g_u and g_v, if it is possible without extensive computations. Let's consider a problem of optimal crossover P_{oc}, which is a reduced version of the initial SCP where the covering subsets restricted by the set of indices $N' = \{j | x(g_u)_j = 1\} \cup \{j | x(g_v)_j = 1\}$.

First, in *LP-crossover* a simple reduction is applied to P_{oc}. Denote $S = \{i \in M : |N_i \cap N'| = 1\}$. Then each row $i \in S$ may be covered by a single column $j(i)$ in P_{oc}. Let's call $Q = \{j | j = j(i), i \in S\}$ a set of fixed columns. For each $i \in \bigcup_{j \in Q} M_j$, one of the columns that cover the gene $g^{(i)}$ in Q is assigned. Let's refer to these genes as fixed too. As a result of the reduction we obtain a new subproblem P_r, that consists of the rows and columns that were not fixed during this procedure. Second, the dual simplex method is used to solve the linear relaxation of P_r. If the solution x' obtained by simplex method is integer, then it is used to complete the genotype g. To avoid time-consuming computations we don't start the simplex method if the number of rows in P_r exceeds a certain threshold μ (we use $\mu = 150$). If this is the case or if the number of simplex iterations exceeds its limit (equal to 300 in our experiments), or if the solution x' is fractional, then *LP-crossover* returns an unchanged genotype g_u.

We designed the mutation operator similar to the proportional selection procedure. Suppose, i-th gene is to be mutated. Then the probability to assign a column $j \in N_i$ to it equals $p_i(j) = \frac{1}{c_j} / \sum_{k \in N_i} \frac{1}{c_k}$.

3.3. Redundant columns elimination

We use three greedy-type heuristics to exclude the redundant columns from the solution. The most simple heuristic *Prime* starts with a given cover discarding the columns in increasing order of indices. A column is discarded if the remaining solution is still a cover.

The second heuristic is the well-known *Greedy* algorithm [8]. This algorithm might find a solution which is not minimal, therefore *Prime* is run after it to eliminate the redundant columns.

The next heuristic is the *Dual Greedy* algorithm, which combines the successive columns elimination and the adaptive columns pricing. Let's denote the set of columns in the subproblem P_g by $N' := \{j \in N : x(g)_j = 1\}$. A cover J is obtained as follows.

The Dual Greedy Algorithm

1. Set $N'_i := N_i \cap N'$ for all $i = 1, ..., m$, $M' := M, J' := N', J := \emptyset$.
2. While $J' \neq \emptyset$ do
2.1. If there is $i \in M'$ such that $|N'_i| = 1$, then set $J := J \cup \{j\}, M' := M' \backslash M_j$, for $j \in N'_i$. Otherwise choose $j \in J'$ such that $j = \text{argmax}_{k \in J'} \frac{c_k}{|M_k \cap M'|}$
2.2. Set $J' := J' \backslash \{j\}$, $N'_i := N'_i \backslash \{j\}$ for all $i \in M_j$.

4. HYBRID ALGORITHM

The hybrid algorithm incorporates NBGA, L-class enumeration, Lagrangean heuristic and greedy algorithm [8]. The algorithm starts with finding an approximate solution ζ^0 by means of the greedy algorithm. After that the new solutions and the lower bound of the optimal goal function value are sought using Lagrangean relaxation (more details about the Lagrangean heuristic in use will be given below). In case the lower bound shows that the best found solution ζ^0 is optimal, the algorithm terminates its work. Otherwise the better solutions are sought NBGA. This finishes the initial stage, after which the LCE is performed.

The testing on step 2 of LCE is now performed both by the knapsack greedy and by Lagrangean heuristics. The Lagrangean relaxation includes all linear constraints of the SCP, and for each of them a Lagrangean multiplier is introduced. The multipliers are updated according to subgradient optimization algorithm [7]. The current best solution ζ^0 may be

improved in some cases by finding an approximate integer solution. To do this, after every γ iterations of Lagrangean heuristic on the basis of the current multipliers we construct a dual feasible solution y with the help of *DualFeas* procedure, proposed in [2]. The constant γ here is chosen experimentaly. Using the solution y with a help of reduced cost heuristic *RCH* [2] we obtain an approximate primal integer solution $\tilde{\zeta}$. If $(c, \tilde{\zeta}) < r$, then the goal functon record is updated: $r := (c, \tilde{\zeta})$. The Lagrangean heuristic stops if for the last τ iterations the lower bound was not improved, or if it turnes out that the lower bound exceeds $r - 1$.

In hybrid algorithm the knapsack greedy heuristic and the Lagrangean relaxation are applied not only in the group testing procedure, but every time before solving the supplementary subproblem on step 2. In case infeasibility of subproblem is established, the computationally expensive run of lexicographical simplex method is not needed. The same version of Lagrangean heuristic is used at the initial stage. The hybrid algorithm always finds an optimum in SCP within finite number of steps, and the number of L-classed considered is usually substantially smaller compared to the basic algorithm LCE.

4.1. Computational experiment

The hybrid algorithm was implemented in Borland Pascal and tested on Pentium Cleron computer (460 MHz, 32Mb RAM). The experiments were carried out on the randomly generated series **4** and **6** from OR-Library [3] ($n = 1000, m = 200, c_j \in [1, .., 100]$), and the series **47** of 10 unicost problems ($n = m = 100$, each constraint contains 4 to 5 ones). The series **47** is proposed by Y.Kochetov and it is available through the Internet at *http://dol.iitam.omsk.net.ru*.

The GA parameters in hybrid algorithm were set as follows: $N = 100$, $p_m = 0.2$, $t_{max} = 1000$. Before solving the non-unicost problems in GA we apply a core-refining reduction which keeps only the columns from the set $N_{core} = \bigcup_{i=1}^{m} \alpha_i^{10}$. Here α_i^k is the set of the greatest k indices in $N_i, i = 1, \ldots, m$. At the initial stage the Lagrangean heuristic is run one time, and after that the NBGA is run twise. In Lagrangean heuristic $\tau = 500$, and $\gamma = 10$.

The results of these experiments are shown in Tables 1-3. The column 'No' shows the problem number in series; T^* and T_{ex} contain the time when optimum is found and the total execution time of the program (in seconds). The column 'Prc' shows which procedure was the first to find the optimum (Grd - greedy algorithm; LR - Lagrangean relaxation at the initial stage; GA - genetic algorithm; LCE - L-class enumeration). The column θ_L gives the number of L-classes visited by the algorithm.

Table 1. The series **4**

No	LCE			hybrid algorithm			
	T_{opt}	T_{ex}	L	T_{opt}	T_{ex}	Prc	θ_L
1	96	365	18	3	3	LR	0
2	594	1461	86	2	2	LR	0
3	87	362	30	2	2	LR	0
4	318	947	59	16	16	GA	0
5	97	352	29	2	2	LR	0
6	149	842	49	23	57	GA	0
7	188	400	24	2	2	LR	0
8	164	764	32	16	79	GA	0
9	511	1485	126	100	128	LR	2
10	304	537	77	2	2	LR	0

Table 2. The series **6**

No	LCE			hybrid algorithm			
	T^*	T_{ex}	L	T^*	T_{ex}	Prc	θ_L
1	761	1354	52	33	288	GA	10
2	130	751	47	45	199	GA	14
3	183	543	23	26	128	LR	3
4	61	531	19	22	88	LR	2
5	1322	2120	187	29	304	GA	15

Table 1 contains the computational results for series **4**. In most of the cases the optimal solution is found already in the Lagrangean relaxation. The L-class enumeration was necessary only on one of these problems; on the rest of them the optimality of the obtained solutions followed from the Lagrangean lower bounds. The overall computation time of the hybrid algorithm on this series is approximately 26 times less than the time of pure L-class enumeration. This large speed up is connected with a small size of the duality gap in the problems of Series **4**.

Series **6** and **47** contain the problems with larger duality gap. Due to this fact these series are usually

Table 3. The series **47**

No	LCE			hybrid algorithm			
	T^*	T_{ex}	L	T^*	T_{ex}	Prc	θ_L
1	279	326	114	11	58	GA	11
2	248	253	74	6	15	GA	1
3	431	448	216	8	67	GA	16
4	114	123	46	6	27	GA	6
5	0	56	8	0	46	Grd	8
6	268	276	90	87	91	LCE	26
7	609	645	176	10	75	GA	15
8	174	323	71	26	148	GA	35
9	77	107	34	7	32	GA	6
10	117	171	40	8	67	GA	16

harder than **4** for the algorithms based on the linear relaxation. In all problems of Series **6** and **47**, except for problem **47.6**, the optimum was found by the heuristics before L-class enumeraton. The total number of the visited L-classes in comparisson with LCE reduced on series **6** on average 7 times and on series **47** this ratio was about 6. The computation time of the hybrid algorithm was on average 5.3 times less compared with LCE on series **6**, and 4.5 times less on series **47**.

A detailed description of computational experiments with NBGA solving the OR-Library benchmark problems is provided in [11]. For all of the 50 instances with random data and known optimal solutions with 1000 to 5000 variables and 200 to 500 constraints the GA found the optimum at least once.

Besides that we considered two sets of combinatorial unicost set covering problems. The series **Stein** consists of problems that arise from Steiner triple systems. These instances were proposed in [12] as the examples of hard problems that can be used for evaluation of the algorithms. During the experiments with these probems the NBGA found the optimal solutions in all cases where the optimum is known; the solution found for the instance **Stein.243** improves the best result known to us from the literature [17].

The NBGA was also tested on a set of problems **CLR** derived from a question of P.Erdös (see e.g. [10, 14]). For the instances **CLR.12** and **CLR.13** the NBGA found the new solutions of 23 elements and for the rest of the cases it found the solutions of previously known values (see e.g.[14]). The optimal solution value for the problems in this series was not known before. With the help of our hybrid algorithm we have established the optimality of the solution with 26 elements for the problem **CLR.9**.

On the basis of the computational experiments we conclude that the proposed hybrid approach is promising. A combination of Lagrangean relaxation, genetic algorithm and L-class enumeration allows to substantially shorten the time of search for the optimum as well as the total computation time.

5. REFERENCES

[1] Alexandrov D., Kochetov Y. Behavior of the Ant Colony Algorithm for the Set Covering Problem, Proc. of Symp. on Oper. Res., Springer Verlag, 2000, pp. 255-260.

[2] Balas E., Carrera M.C. A Dynamic Subgradient-Based Branch and Bound Procedure for Set Covering, Oper. Res., vol.44, No 6, 1996, pp. 875-890.

[3] Beasley J.E. OR-Library: Distributing Test Problems by Electronic Mail, Journ. of the Oper. Res. Soc., vol. 41, No 11, 1990, pp. 1069-1072.

[4] Beasley J.E., Chu P.C. A Genetic Algorithm for the Set Covering Problem, European Journ. of Oper. Res., vol. 94, No 2, 1996, pp. 394-404.

[5] Brauner N., Dhaenens-Flipo C., Espinouse M.-L., Finke G., Gavranovic H. Decomposition Into Parallel Work Phases With Application to the Sheet Metal Industry, Proc. of Intern. Conf. on Industr. Engin. and Prod. Manag., Glasgow, 1999, vol. 1, pp. 389-396.

[6] Beasley J.E., Jörnsten K. Enhancing an Algorithm for Set Covering Problems, European J. Oper. Res., vol. 58, 1992, pp. 293-300.

[7] Caprara A., Fischetti M., Toth P. Algorithms for the Set Covering Problem, Technical Report OR-98-3, DEIS-Operations Research Group, 1998.

[8] Chvátal V. A Greedy Heuristic for the Set Covering Problem, Mathematics of Operations Research, vol. 4, No 3, 1979, pp. 233-235.

[9] Davis L. Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[10] Erdös P. On a Combinatorial Problem, Nordisk Mat. Tidskrift, vol. 11, 1963, pp. 5-10.

[11] Eremeev A.V. A Genetic Algorithm with a Non-Binary Representation for the Set Covering Problem, Proc. of Oper. Res., Springer Verlag, 1999, pp.175-181.

[12] Fulkerson D.R., Nemhauser G.L., Trotter L.E. Two Computationally Difficult Set Covering Problems that Arise in Computing the 1-Width of Incidence Matrices of Steiner Triple Systems, Math. Progr. Study, vol. 2, 1974, pp. 72-81.

[13] Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning, Reading: Addison Wesley, 1989, 412 p.

[14] Grossman T., Wool A. Computational Experience with Approximation Algorithms for the Set Covering Problem, Eur. J. Oper. Res., vol. 101, No 1, 1997, pp. 81-92.

[15] Kolokolov A.A. Regular Partitions and Cuts in Integer Programming, Discr. Analysis and Oper. Res., Kluwer Academic Publ., 1996, pp. 59-79.

[16] Kolokolov A.A., Eremeev A.V., Zaozerskaya L.A. On Hybrid L-class Enumeration and Genetic Algorithm for Set Covering Problem, 15-th Conf. of Intern. Federat. of Oper. Res. Soc., Abstr, Pekin, 1999, P.117.

[17] Mannino C., Sassano A. Solving Hard Set Covering Problems, Oper. Res. Letters, vol.18, 1995, pp.1-5.

[18] Zaozerskaya L.A. On L-Class Enumeration Algorithm for Set Covering Problem, Proc. of 11-th Baikal Intern. School-Seminar "Optimization Methods and Their Applications", 1998, pp. 139-142 (in Russian).