ISSN 1512-1232

Solution of Problem of Set Covering by Means of Genetic Algorithm

Natela Ananiashvili
Faculty of Exact and Natural Sciences, I. Javakhishvili Tbilisi State University, Tbilisi

Abstract

Heuristic algorithm of solution of problem of covering minimal summary weight of given set with subsets of non-uniform values is offered. The algorithm is based on genetic algorithm with operators of crossover and mutation. The results of computational experiments are given on the basis of famous test problems.

Keywords: *minimal set covering, heuristic algorithm, genetic algorithm*

1. Introduction

The problem of set covering is a NP-complex problem of combinatorial optimization [1]. Algorithms of precise solution of this problem use techniques of branches and limits [2]. Time necessary for realization of such algorithms rapidly increases and when dimension of a problem also increases, it is impossible to get optimal values in real time [3-4]. This problem is a mathematical model for practical tasks, such as location of service centers, development of transport schedule [6], location of sources of power systems [6], etc. Therefore, it is very important to solve this problem in real time. When such problems are solved, heuristic algorithms are often used that find near optimal solutions in reasonable time interval. Approximate algorithms mainly imply partial selection of covering sets. In genetic algorithms, this process is similar to development of biological populations [7-8]. Therefore, terminology is similar to biological terminology.

2. Problem Formulation

The problem of set covering can be formally described as follows (for instance. as in [2]). Let us assume, we have finite set $R = \{r_1, r_2, \dots, r_m\}$ and set $\mathcal{L} = \{S_1, S_2, \dots, S_n\}$ of subsets $S_j \subset R$, where each S_j is assigned with positive weight $c_j > 0$. Any subset $\mathcal{L}' = \{S_{j_1}, S_{j_2}, \dots, S_{j_k}\}$ from $\mathcal{L}$ is called covering of set R, if the following condition is met: $\bigcup_{i=1}^{k} S_{j_i} = R$. Minimal covering means selection of subset from such subsets, when sum of weights $\sum_{j=1}^{k} c_{j_i}$ is minimal. It is the problem of weighted covering.

Let us introduce matrix $A = (a_{ij})_{m \times n}$ to write down a mathematical model of problem of set covering:

$$a_{ij} = \begin{cases} 1, if\ r_i \in S_j \\ 0, if\ r_i \notin S_j \end{cases}$$

It is presumed that r_i is in one of S_j. Let us introduce binary variables $x_j,\ j = 1,2,\dots,n$:

$$x_j = \begin{cases} 1, \text{if } S_j \text{ is in covering} \\ 0, \text{if } S_j \text{ isn't in covering} \end{cases}$$

Therefore, the problem of covering can be formulated in the following way. Let us minimize the following sum:

ISSN 1512-1232

$$f(x) = \sum_{j=1}^{m} c_j \cdot x_j , \qquad (1)$$

considering the following restrictions:

$$\sum_{j=1}^{m} a_{ij} \cdot x_j \geq 1, \quad i = 1, \dots, n, \quad x_j \in \{0, 1\}, \quad j = 1, \dots, m, \qquad (2)$$

3. The General Design of Genetic Algorithm

The proposed algorithm of solution of problem of minimal covering implies general principles of genetic algorithms [8]. It can be described informally in the following way:

Step 1. Selection of initial population.

Step 2. Determination of function of fitness (usefulness) and estimation of individuals' fitness from the corresponding population.

Step 3. Selection of parents on the basis of fitness. Usage of genetic operators, such as crossover and mutation for selected individuals and derivation of offspring individuals.

Step 4. Estimation of fitness of offspring individuals. Replacement of one parent with the best offspring individual from the initial population on the basis of fitness.

Step 5. If condition of completion of algorithm is met, then check if number of iterations has been expired. In other case, jump to Step 3.

Let us consider a design that is used in the proposed algorithm. The first step of genetic algorithm is selection of scheme of encoding. Binary encoding is selected, i.e. every chromosome is n-dimension vector $x^k = (x_1^k, x_2^k, \dots, x_n^k)$, where x_j^kth element is equal to 1, if set S_j is into *k*th chromosome and $x_j^k = 0$, if the same set isn't in the *k*th chromosome. It means that x^k values ($k = 1,2, \dots, s$) are chromosomes that correspond to the individuals. x_j^k Values are genes that can be equal to 0 or 1. S_j sets correspond to genotypes. Quantity s of individuals (chromosomes) in a population depends on the scale of problem. Generally, $25 \leq s \leq 100$ values are chosen.

First of all, let us select an initial population and then compute value of fitness function for chromosomes:

$$f_k = \sum_{j=1}^{n} c_j \cdot x_j^k , \quad k = 1,2, \dots, s \qquad (3)$$

Then we select the parents. When the problems are solved by means of genetic algorithms and the parents are selected, technique of roulette or some other techniques are used [8]. In the proposed article, selection is made with the following technique. At the odd iteration two individuals are selected from the population that has minimal values of fitness function. At the even iteration we select one chromosome with minimal value of fitness function and another chromosome with maximal value of fitness function. In this way we'll avoid rapid convergence towards some local minimum. Then we breed selected individuals with operator of one-point crossover. Individuals derived after crossover are mutated. Fixed value $\frac{1}{n}$of mutation operator is often used in genetic algorithms, but we choose variable value of mutation, because purpose of crossover and mutation operators is to derive individuals that are different from individuals of population. Besides, when we approach local extreme, individuals are slightly different from each other. Therefore, we must use mutation operator with variable value to avoid selection of only kindred individuals. This value will depend on individual, as well as characteristics of genotype of these individuals, particularly quantity of 1s and 0s in the genotype.

ISSN 1512-1232

From the chromosomes derived by means of crossover and mutation, we select the best one that has minimal value of function of fitness/usefulness. Then we look for individual with maximal fitness in the initial population and replace it with individual selected from offspring. The process of selection of parent chromosomes, crossover, mutation and replacement of parent individual with offspring individual is repeated until the end of iterations.

4. Algorithm

4.1. Formation of initial population

Let us assume that columns of matrix A are arranged according to growth of costs.

Quantity of individuals of population is denoted with s, L is the matrix of population with $s \times n$ size.

Algorithm 1: Formation of initial population.

Step 1. Let us take $L(k,\ j) = 0, k = 1,2,\dots,s, j = 1,2,\dots,n$; $\mathrm{R}(\ell) = 0, \ell = 1,2,\dots,\mathrm{m}$; $i = 1$; $t = 1$;

Let us form the first row of matrix L, i.e. the first chromosome of population in the following way:

Step 2. Let us find the first column with number j_1 of matrix A that covers r_ith node, i.e. $a(1, j_1) = 1$ and take it into covering. Let us assume, $L(1, j_1) = 1$, $\mathrm{R}(\ell) = \mathrm{R}(\ell) + \mathrm{a}(\ell, \mathrm{j}_1),\ \ell = 1,2,\dots,\mathrm{m}$.

Step 3. Let us find number i_1 of the first zero element of R. Let us take $i = i_1$ and jump to step 2. If zero element is absent, then jump to step 4.

The following rows of matrix *L*, i.e. remaining chromosomes of population are formed in the following way:

Step 4. Let us take $t = t + 1$. If $t > s$, then finish, otherwise take $\mathrm{R}(\ell) = 0, \ell = 1,2,\dots,\mathrm{m}$, $i = 1$.

Step 5. Let us find numbers $\{j_1, j_2, \dots, j_{h_i}\}$ of columns of matrix A that cover r_ith node. Then randomly select some number j_u from $\{j_1, j_2, \dots, j_{h_i}\}$, i.e. $a(i, j_u) = 1$ and take it into *t*-th chromosome: $L(t, j_u) = 1$. Let us assume $\mathrm{R}(\ell) = \mathrm{R}(\ell) + \mathrm{a}(\ell, \mathrm{j}_\mathrm{u}),\ \ell = 1,2,\dots,\mathrm{m}$.

Step 6. Let us find number i_1 of the first zero element of R. If such element exists, then assume $i = i_1$ and jump to step 5. If zero element doesn't exist, the jump to step 4.

When a population is formed with such technique, it is possible to find excessive genes in each population, i.e. after deleting some or several columns from each covering, set $R = \{r_1,\ r_2, \dots, r_m\}$ can be covered again. We call J dead-end covering, if $J \setminus \{j\}$ isn't covering for any $j \in J$. Therefore, it is necessary to delete excessive columns from each individual of given population and make them dead-end. For this purpose the following heuristic algorithm is used:

Algorithm 2: Deletion of excessive columns from *k*th individual.

Step 1. Let us define $j \in J$ for every $\ell = 1, 2, \dots, m$ as $\mathrm{R}(\ell) = \mathrm{h}_\ell$, where h_ℓ is quantity of columns that cover r_ℓth node in kth chromosome. It means that we must find quantity h_ℓ of covering columns $\{j_1, j_2, \dots, j_{\mathrm{h}_\ell}\}$ in kth chromosome that is represented in kth row of matrix *L*, when $a(\ell, j_\mathrm{p}) = 1, p = 1,2,\dots,\mathrm{h}_\ell$.

Let us assume $i = 1$.

Step 2. Let us find minimal number i_1: $i \le i_1 \le m$, when $\mathrm{R}(i_1) > 1$. If such number doesn't exist, then the process is finished.

ISSN 1512-1232

Step 3. If every $\mathrm{R}(l) - a(l,\ j_p) > 0,\ l = 1,2,\dots,m$, then deletion of column j_p from *k*th chromosome is possible. Let us assume $\mathrm{R}(l) = \mathrm{R}(l) - a(l,\ j_p),\ l = 1,2,\dots,m$ and $L(k, j_p) = 0$.

Step 4. Let us assume $\mathrm{p} = \mathrm{p} + 1$. If $\mathrm{p} \le \mathrm{u}$, then jump to step 3.

Step 5. Let us assume $i = i_1 + 1$. If $i \le m$ then jump to step 2, otherwise the process is finished.

By means of this algorithm we can delete excessive columns from *k*th individual $(k = 1, 2, \dots, s)$ of population *L* and find dead-end coverings.

4.2. Crossover

After formation of population, we can compute values of fitness function for each chromosome by means of equation (3). We select two parent chromosomes on the basis of rule that is described in the third paragraph. Let us assume these chromosomes are:

$$x' = (x'_1, x'_2, \dots, x'_n)$$
$$x'' = (x''_1, x''_2, \dots, x''_n)$$

Let us define location of crossover with random value $k \in \{1,2,3,\dots,n-1\}$. After crossover we'll get offspring chromsomes:

$$x^{ch_1} = (x'_1, x'_2, \dots x'_k, x''_{k+1}, \dots, x''_n)$$
$$x^{ch_2} = (x''_1, x''_2, \dots, x''_k, x'_{k+1}, \dots, x'_n)$$

After such crossover offspring chromosomes may not cover $R = \{r_1,\ r_2, \dots, r_m\}$. So, it is necessary to find and add covering sets for uncovered nodes of those chromosomes. We propose algorithm that guarantees this process:

Algorithm 3: Addition of subsets that cover uncovered nodes to chromosomes.

For legibility, chromosome x^{ch_1} is selected.

Step 1. Let us find numbers of non-zero elements in chromosome x^{ch_1}. $\{j_1, j_2, \dots, j_h\}$ are numbers of corresponding covering sets of this chromosome. To satisfy condition $\ell = 1,2,\dots,\mathrm{m}$ for every ℓth row of columns $\{j_1, j_2, \dots, j_h\}$ of covering matrix *A*, let us count number k_ℓ of columns that cover r_ℓth node, i.e. when $a\left(\ell, j_{i_p}\right) = 1, p = 1,2,\dots,k_\ell$ and $\left\{j_{i_1}, j_{i_2}, \dots, j_{i_{k_\ell}}\right\} \subset \{j_1, j_2, \dots, j_h\}$. Let us assume $R(\ell) = k_\ell$ and $i = 1$.

Step 2. Let us find minimal number i_1: $i \le i_1 \le m$ for which $\mathrm{R}(i_1) = 0$. If every element of R is non-zero, then the process is finished.

Step 3. Let us find numbers $\{j_1, j_2, \dots, j_v\}$ of those columns of matrix A that cover r_{i_1} node. Let us select randomly any number j_u from $\{j_1, j_2, \dots, j_v\}$, i.e. $a(i_1, j_u) = 1$; take column j_u into the current chromosome: $x^{ch_1}(j_u) = 1$.

Let us assume $\mathrm{R}(\ell) = \mathrm{R}(\ell) + \mathrm{a}(\ell, \mathrm{j_u}),\ \ell = 1,2,\dots,\mathrm{m}$.

Step 4. Let us assume $i = i_1 + 1$. If $i \le m$, then jump to step 2, otherwise the process is finished.

By means of described algorithm chromosomes x^{ch_1} and x^{ch_2} will cover again set $R = \{r_1,\ r_2, \dots, r_m\}$. After crossover offspring chromosomes are mutated.

4.3. Mutation

Let us select some number *k* from set $k \in \{1,2,3,\dots,n\}$ for offspring chromosomes x^{ch_1} and x^{ch_2} and mutate *k*th gene of this chromosome. Note that specification of gene in this problem is its weight, as well is number of 1s and 0s in corresponding genotype. Let us calculate quantity $p_1(j)$ of

ISSN 1512-1232

1s and quantity $p_0(j)$ of 0s for each *j*th columns of matrix A, when $j = 1,2,\dots,n$. Besides, let us calculate the following values for each column:

$$E(j) = -p_0(j) \cdot log(p_0(j)) - p_1(j) \cdot log(p_1(j)), \qquad j = 1,2,\dots,n$$

then develop vector of probability of mutation:

$$pmt(j) = \frac{\frac{1}{E(j)}}{\sum_{u=1}^{n} \frac{1}{E(u)}}, \quad j = 1,2,\dots,n$$

Let us select gene τ for mutation, where τ is random normalized integer from range $1 \le \tau \le n$. Let us define condition of mutation:

1. If the following conditions are met: $pmt(\tau) > \frac{1}{n}$, $x^{ch_1}(\tau) = 1$ and $p_0(\tau) > p_1(\tau)$, then gene τ mutates and we'll get $x^{ch_1}(\tau) = 0$.

2. If the following conditions are met: $pmt(\tau) > \frac{1}{n}$, $x^{ch_1}(\tau) = 0$ and $p_0(\tau) > p_1(\tau)$, then gene τ mutates and we'll get $x^{ch_1}(\tau) = 1$.

Similarly we can mutate the second chromosome x^{ch_2}. As in the case of crossover, after mutation offspring chromosomes may not cover set $R = \{r_1, r_2, \dots, r_m\}$. Let us use above mentioned algorithm #3 for x^{ch_1} and x^{ch_2}. Then add columns that cover every uncovered node. Chromosomes derived after addition may not represent dead-end coverings. Let us use algorithm #2 and delete excessive columns. This process is repeated until depletion of iterations.

5. The results of experiment

This algorithm is realized in Matlab. Test problems from famous library Or-Library [9] are used to check efficiency of proposed algorithm. The library includes problems of 11 classes. These classes are 4, 5, 6, A, B, C, D, E, F, G, H. The problems are derived randomly with weights $c_j \in [1,100]$ and matrices A, where number of 1s is from 2% to 20%.

Table 1. The names and sizes of problems

Problem class	Number of rows	Number of columns	Number of problems
4	200	1000	10
5	200	2000	10
6	200	1000	5
A	300	3000	5
B	300	3000	5
C	400	4000	5
D	400	4000	5
E	500	5000	5
F	500	5000	5
G	1000	10000	5
H	1000	10000	5

10 experiments were made for each problem and for each experiment new population was selected. The best results are given in Table 2 and 3. These results are compared to values of optimal weights of minimal covering. In sum, 65 problems were considered. Size of population was always $s = 100$. Number of iterations changed from 550 to 35250. For class 4-6 and A-D problems optimal values are known [10]. For large-scale class E-H problems the best known results are taken [11].

ISSN 1512-1232

Let us denote with F_{min} the best result of experiments.

Table 2. The results of experiments for class 4-6 and A-D problems

Problem class	Optimal solution	F_{min}	Number of iterations	Computation time
4,1	429	432	850	5.748834
4,2	512	549	27250	189.521844
4,3	516	526	650	5.591801
4.4	494	509	5250	41.614141
4,5	512	527	650	7.285392
4,6	560	578	550	4.547840
4,7	430	437	650	4.874183
4,8	492	499	750	5.513872
4,9	641	665	15250	84.042923
4,10	514	537	750	5.267568
5,1	253	264	27250	184.503032
5,2	302	311	25250	160.479980
5,3	226	237	1250	10.480415
5,4	242	245	5250	38.623774
5,5	211	212	5250	36.555420
5,6	213	228	5250	35.234376
5,7	293	313	1250	11.987480
5,8	288	296	15250	106.514343
5,9	279	280	35250	254.069516
5,10	265	270	45250	345.253260
6,1	138	147	550	4.952619
6,2	146	150	35250	239.397738
6,3	145	150	5250	35.749140
6,4	131	132	5250	38.840746
6,5	161	169	1250	10.292615
A,1	253	255	1250	20.120029
A,2	252	267	850	15.381661
A,3	232	246	45250	699.561851
A,4	234	246	5250	79.033534
A,5	236	240	15250	279.40790
B,1	69	76	15250	260.158726
B,2	76	83	5250	114.132584
B,3	85	85	550	13.921434
B,4	79	83	5250	97.910171
B,5	72	75	650	12.822692
C,1	227	233	35250	886.189983
C,2	219	227	40250	835.990969
C,3	243	254	15250	367.848676
C,4	219	235	35250	844.615780
C,5	215	221	850	26.015785
D,1	60	60	35250	859.387968

ISSN 1512-1232

D,2	66	70	550	23.790084
D,3	72	77	15250	452.277166
D,4	61	64	850	28.991679
D,5	62	64	550	18.933103

Table 3. The results of experiments for class E-H problems

Problem class	Proposed minimal value	F_{min}	Number of iterations	Computation time
E,1	29	29	650	37.089270
E,2	30	32	1250	55.733434
E,3	27	29	5250	222.218509
E,4	28	31	15250	795.258351
E,5	28	30	550	31.467857
F,1	14	16	550	38.747741
F,2	15	17	650	36.459649
F,3	14	17	550	32.964627
F,4	14	16	850	45.287650
F,5	14	16	550	31.603698
G,1	179	185	35250	4163.728506
G,2	158	164	25250	2921.625042
G,3	169	175	5250	932.124276
G,4	172	176	25250	3059.695269
G,5	168	177	35250	5870.136848
H,1	64	69	5250	706.016420
H,2	64	66	35250	4681.564954
H,3	60	67	1250	226.269730
H,4	59	65	550	111.810822
H,5	55	60	850	148.935323

For test the standard computer was used with specifications Intel(R) Pentium (R) Dual CPU E2220 2.40 GHz, 2.00 GB of RAM. In most cases the problems were solved in seconds, but in some cases they required several minutes. Only class E-H problems requires more than hour. On the basis of experiment, we can conclude that proposed algorithm is sufficiently efficient.

References:

1. Garey M. R., Johnson D. S. *Computers and intractability. A guide to the theory of NP-completeness*. San Francisco: W. H. Freeman and Co.,1979.
2. Christofides Nicos, Graph Theory. An Algorithmic Approach, Computer science and applied mathematics, London, Academic Press, 1986.
3. Ananiashvili N., "About the one solution of the set partition problem". *V Annual international conference of the Georgian mathematical union, Batumi,* September, 2014. pp.60.
4. Ananiashvili N., "Solution of problems of minimal set partition and set covering". (2015)Bull. Georg. Natl. Sci., 9, 1: 38-42
5. Capara A. et al. Algorithms for Railway Crew Management // Mathematical Programming. 1997. 79. P. 125–141.
6. Minieka E., *Optimization Algorithms For Networks And Graphs,* New York : M. Dekker, c1978.
7. Randy L. Haupt, Sue Ellen Haupt, "Practical genetic algorithms"-2nd ed.Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 2004.

ISSN 1512-1232

8. Rutkovskaia D., Pilinski M., Rutkovski L., “Neironnie Seti, Geneticheskie Algoritmi i Nechetkie Sistemi”. Moscow, “Goriachaia Linia – Telekom”. 2006 (In Russian).
9. Beasley, J. E. , OR-Library: "Distributing Test Problems by Electronic Mail,The Journal of the Operational Research Society". Vol. 41, No. 11 (Nov., 1990), pp. 1069-1072.
10. Beasley J. E. and Jornsten K. Enhancing an algorithm for set covering problems //European Journal of Operational Research. 1992. 58. P. 293–300.
11. Jacobs L. W. and Brusco M. J. A simulated annealing-based heuristic for the set covering problem //Working paper, Operations Management and Infprmation Systems Department, Northern Illinois University, Dekalb, IL 60115, USA. 1993.

Article received: 2015-07-09