Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007

HYBRID NICHE GENETIC ALGORITHM FOR SET COVERING PROBLEM

YOU-LIAN ZHENG[1], DE-MING LEI[2]

[1]School of Mathematic and Computer Science, Hubei University
[2]School of Automation, Wuhan University of Technology
E-MAIL:deminglei11@163.com

Abstract:

Set covering problem (SCP) is a well-know combinatorial optimization problem. This paper presents a hybrid niche genetic algorithm to solve set covering problem. The various genetic representation strategies of SCP are analyzed and an effective non-binary representation method is proposed. The method of search space compression is then presented and the hybrid niche genetic algorithm is designed. Finally, the hybrid algorithm is tested by using some SCP instances and the computational results demonstrate the good performance of hybrid genetic algorithm on SCP.

Keywords:

Set covering problem; Non-binary encoding; Niche; Genetic algorithm

1. Introduction

Set covering problem (SCP) is the problem of covering the rows of a $m \times n$ zero-one matrix $A = (a_{ij})$ by a subset of columns at minimal cost. The problem can be described in the following way:

$$\begin{aligned} &Min \sum_{j \in J} c_j x_j \\ &Subject\ to \sum_{j \in J} a_{ij} x_j \geq 1 \ \ for\ i \in I \\ &\qquad x_j \in \{0,1\},\ for\ j \in J \end{aligned} \tag{1}$$

Where $I = \{1, 2, \cdots n\}$. $J = \{1, 2, \cdots m\}$. If column j is in the solution, $x_j = 1$; otherwise $x_j = 0$.

SCP is important in practice. It has been used to model a large range of problems arising from scheduling, manufacturing, service planning, information retrieval etc. The delivery and routing problem and the airline crew scheduling problem are two famous instances. A number of heuristics and meta-heuristics have been presented to solve SCP. Haddadi proposed a simple Lagrangian heuristic based on Lagrangian duality, greedy heuristic, sub-gradient optimization and redundant covers. Lan et al. presented an effective and simple heuristic. Beasley et al proposed an effective genetic algorithm (GA) for SCP by introducing a new fitness-based crossover, a variable mutation rate et al. Aickelin et al developed an indirect genetic algorithm, in which the actual solution is found by an external decoder function. Solar et al proposed a parallel genetic algorithm, which performs better than sequential GA.

In this study, a new representation method is proposed and the hybrid niche genetic algorithm merging local search and fitness sharing GA is designed. The hybrid algorithm is applied to a set of instances and computational results are presented.

2. Hybrid niche genetic algorithm

2.1. Niche genetic algorithm

Genetic algorithm is a stochastic optimization method based on the mechanics of natural evolution and natural genetics. GA has shown its strong optimization ability to many complex problems; however, it also has some drawbacks such as premature convergence and the occurrence of slow convergence rate when its search approximating to the global optimal solution. To overcome these problems, various improvement strategies have been proposed and the introduction of niche is an importation method.

The basic idea of niching techniques is originated from the fact that the similar organisms with the similar features always survive together and species is a collection of these organisms. In natural ecosystem, a niche can be viewed as an organism's task, which permits species to survive in their environment. By analogy, in niche GA**,** a niche is commonly referred to as the location of each optimum in the search space, the fitness representing the resources of that niche. The organisms in a niche can be defined as similar individuals in terms of similarity metric.

Fitness sharing is one of two general niching techniques.Sharing decreases the fitness of the individual in a niche by an amount related to the number of individuals

1-4244-0973-X/07/$25.00 ©2007 IEEE

in the niche. Specially, the shared fitness $f_i^{'}$ of individual i in a niche is equal to its prior fitness f_i divided by its niche count m_i.

$$f_i^{'} = f_i / m_i$$

An individual's niche count is the sum of the sharing function values between itself and each individual in the population. The sharing function of an individual i is given by the following equation.

$$sh(d_{ij}) = \begin{cases} 1-(d_{ij}/\sigma_{share})^{\alpha} & d_{ij} < \sigma_{share} \\ 0 & otherwise \end{cases}$$

Where d_{ij} denotes the distance between individual i and j. σ_{share} is the predetermined threshold of dissimilarity, α is the constant that regulate the shape of sharing function.

$$m_i = \sum_{j=1}^{N} sh(d_{ij}), \quad i = 1,2\cdots N \tag{3}$$

Where N is population scale.

2.2. Representation methods

A number of effective representation methods have been proposed to solve SCP using genetic algorithm. The usual 0-1 binary representation is an obvious choice for the SCP since it represents the underlying 0-1 integer variables. The chromosome consists of n bits. A value of 1 for ith bit implies that column i is in the solution. The binary representation is illustrated in Fig.1(a).

When binary genetic algorithm is used, the feasibility of new solutions obtained by crossover and mutation must be checked and the redundant columns need to be eliminated. On the other hand, if some non-feasible solutions occur, these solutions need to be converted to the feasible ones by a simple heuristic or the punishing function is added to the objective function of these solutions.

With respect to non-binary representation method, Beasley et al describe a method that the length of chromosome is m, the position of each gene corresponds to a row of matrix and the value of gene is the serial number of the column covering the row. This representation method is shown in Fig. 1(b). The advantage of this method is that the new solutions are always feasible and feasibility test is unnecessary.

Aickelin et al. presented an indirect representation method, in which the chromosome of each individual is a

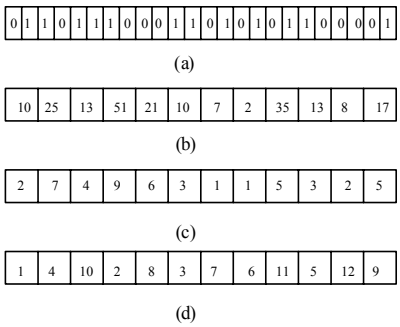

Figure 1 Comparisons among various representation methods

permutation from 1 to n shown in Fig. 1(d). The main feature of this method is that a decoding procedure must be used to convert the chromosome into the solution.

This paper presents a new representation method, which is similar to the method shown in Fig. 1(b). The length of chromosome is also m .Each gene in the chromosome corresponds to a row of matrix A and the gene value is also the serial number of the column covering the row; however, the serial number is obtained in the following way: for each row l, all columns covering the row are recorded in set Θ_l and ranked according to a principle and the serial number is obtained by the results of ranking. In this paper, the columns are ranked according to the value of c_j, the bigger c_j is, the bigger the serial number is. The presentation method is shown in Fig.1(c).

For $m \times n$ matrix, there are n columns and the gene value shown in Fig. 1(b) belongs to $[1,n]$, however, not all integers in interval $[1,n]$ can become the value of a gene. As shown in the Fig.1(c), gene value is contained in interval $[1,h_i]$, where h_i is the biggest number of column covering row i. All integers in $[1,h_i]$ can act as the value of the gene; as a result, the search space can be compressed and many existing mutation strategies can be used to enhance search efficiency.

In the following paragraphs, the decoding procedure is first introduced and then the approach to diminish search space is discussed.

The decoding procedure is described as follow.

1 For each gene corresponding to a row l, determine the column covering the row by finding an element of set Θ_l corresponding to the gene value. For those genes covered by a column, retain the value of one gene and

determine new value and the corresponding column for other genes.

2 For n columns, find the redundant columns from the column with the biggest c_j to the one with the smallest c_j, if a column is redundant, the column is excluded from the solution of SCP. When all redundant columns are eliminated, the solution consists of the remained columns.

If all 1 of a column are included in other columns, the column is redundant. The elimination of the redundant columns doesn't influence the feasibility of solution.

Unlink the procedure proposed by Aichelin et al., the above procedure doesn't choose some of n columns by using a principle but obtain the solution by eliminating redundant columns. It is easy to decide if a column is redundant, while the procedure of Aichelin et al need comprehensively considered many elements to evaluate each candidate column and decides if candidate column can be included in solution of SCP. The proposed procedure is simple and effective.

When all columns covering a row are ranked, the column with bigger c_j always has bigger serial number. Thus, when compressing search space, only the upper bound is considered. The new upper bound is obtained in the following way: for each gene i, $s_i = max\{s_{ij}, j \in S\}$, $s_i \leftarrow s_i + 0.2s_i$, $if\ s_i > h_i,\ then\ s_i \leftarrow h_i$. Where s_{ij} is the value of the gene i of individual j. S is the set of $g\left(\in [0.5N, 0.6N]\right)$ individuals of population with biggest fitness. If elitism strategy is used in GA, the elite individuals are also included in set S.

2.3. Hybrid algorithm for SCP

Hybrid algorithm merging fitness sharing genetic algorithm and local search is presented in this study. The search space is continuously compressed to make the proposed algorithm converging quickly to the optimal solution of SCP. The hybrid algorithm is described as follows.

1 Initialization, for each row, rank all columns covering the row and determine the interval of each gene. Stochastic generate initial population and make the solution with the optimal fitness as elite one.

2 Compute the niche count and the shared fitness for each individual and choose individuals into next generation.

3 Perform crossover and mutation operator, calculate the fitness of each individual and decide the new elite solution, if the new elite solution has bigger fitness than the stored elite, replace the old elite with the new one.

4 Determine the new interval for each gene and perform local search on elite and some individuals which have bigger fitness than any other solutions in population.

5 If the termination condition is met, stop search; otherwise go to (2).

With respect to local search, for the elite and some individuals of population, randomly choose some genes from their chromosomes, then determine a new value $s'_{ij} \in \left[1, s_{ij} + 1\right]$ and replace the old value with the new value for the chosen genes.

3. Computational Results

To test the optimization ability of the hybrid algorithm, four problem sets (4-6, A) including 30 instances are used and the hybrid algorithm is compared with BeCh and IGA.

Parameter settings of the hybrid genetic algorithm (HGA) are: population scale $N = 100$, crossover probability $p_c = 0.9$, $\sigma_{share} = 8$, $g = 0.6N$, mutation probability $p_m = 0.15$, $f_i = 3000 - \sum_{j \in J} c_j x_j$, f_i is fitness of solution i. Elitism strategy is used and local search is only acted on elite solution and three individuals which have bigger fitness than any other individuals in the population.

Table 1 Comparison among three algorithms

problem	Sol.	Ave.			problem	Sol.	Ave.		
		HGA	BeCh	IGA			HGA	BeCh	IGA
4.1	429	429	429.68	429	5.6	213	213	213	213
4.2	512	512	512	512	5.7	293	293	293	293
4.3	516	516	516	516	5.8	288	288	288.8	288
4.4	494	494	494.8	494	5.9	279	279	279	279
4.5	512	512	512	512	5.10	265	265	265	265
4.6	560	560	560	560	6.1	138	138	138	138
4.7	430	430	430.21	430	6.2	146	146.3	146.2	146
4.8	492	492	492.09	492	6.3	145	145	145	145
4.9	641	641	643.11	641	6.4	131	131	131	131
4.10	514	504	514	514	6.5	161	161	161.3	161
5.1	253	253	253	253	A.1	253	253	253.2	253
5.2	302	302.2	303.5	302	A.2	252	252	252	252
5.3	226	228	228	226	A.3	232	232	232.5	232
5.4	242	242	242.1	242	A.4	234	234	234	234
5.5	211	211	211	211	A.5	236	236	236	236

Two-point crossover is considered, two positions is randomly chosen and alter the genes of two parents between two chosen positions. Polynomial mutation is used. Some genes is first chosen from an individual, then polynomial mutation with $\eta_m = 20$ is performed on those genes and the integer part of the obtained result is the new value of gene.

When hybrid algorithm has search 1000000 solutions, the search terminate. The algorithm runs 10 times for each instance. The final computational results are shown in table 1, where Ave. denotes the average value of the objective function of all optimal solution obtained in 10 times. The computational results of BeCh and IGA are respectively taken from [4]，[5]。.

As shown in table 1, HGA obtains the same computational results as IGA on instances 4.1-9. Two algorithms also find the optimal solution of these instances. However, BeCh just obtains the optimal solutions of 4.2-3 and 4.5-6, for other 5 instances, BeCh cannot converge to the optimal solutions in at least one of 10 times. For instance 4.10, both HGA and BeCh converge to the know optimal result, the new optimal result is obtained by HGA and the new solution is {0,1,2,,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,21,22,24,25,27,2829,34,39,40,41,42,46,47,48,49,50,54,55,57,58,59,62,67,68,70,76,81,82,84,85,87,95,106,109,118,122,141,144,154,190,210,234,246,254,256,299,479}.

For 10 instances of problem set 5, the optimal result of IGA in 10 times runs is equal to the known optimal value. HGA and BeCh cannot converge to the optimal solution of 5.3. However, HGA obtains the optimal solution of 5.8 in each run while BeCh only find the optimal solution of 5.8 in two of 10 times. For other instances, HGA and BeCh obtain the optimal results. For all instances of problem set 6 and A, three algorithms converge to the optimal solutions of, however, IGA obtains the best average value and BeCh obtains the worst average value.

For 30 instances, HGA converges to the optimal solution of 29 instances and the new optimal solution of 4.1. The average computational results of HGA are contiguous to those of IGA and better than those of BeCh. Three algorithms with difference representation method obtain similar results

4. Conclusions

Genetic algorithm with the representation method shown in Fig. 1(b) may be inferior to binary coded genetic algorithm. However, a new representation method shown in

Fig. 1(c) is introduced, the hybrid algorithm merging local search and fitness sharing genetic algorithm is proposed. The search space compression is also used in the hybrid algorithm. Computational results show that this kind of non-binary coded genetic algorithm is very competitive with other genetic algorithm using other representation method and the effectiveness of the new representation method is demonstrated.

References

[1] Haddadi S, Simple Lagrangian heuristic for the set covering problem, European Journal of Operational Research, 97:200-204 1997

[2] Caprara A Fischetti M. Toth P, A heuristic method for the set covering problem, Operations Research, 47 (5):730–743. 1999

[3] Lan G. DePuy G.W. Whitehouse G.E, An effective and simple heuristic for the set covering problem, European journal of operational research, 176: 1387-1403, 2007

[4] Beasley J.E. Chu P.C, A genetic algorithm for set covering problems, European journal of operational research, 94, 392-404 1996

[5] Aickelin U, An indirect genetic algorithm for set covering problems, Journal of the Operational Research Society, 53(4): 1118-1126 2002

[6] Solar M. Parada V. Urrutia R, A parallel genetic algorithm for the set-covering problems, Computers and Operation Research, 29: 1221-1235, 2002

[7] Iwamura K. Horiike M. Sibahara T, Input data dependency of a genetic algorithm to solve the set covering problem, Tsinghua Science and Technology, 8(3): 14-18, 2003

[8] Goldberg D E Richardson J. Genetic algorithms with sharing for multimodal function optimization. In: Proc. of 2nd International Conference on Genetic algorithms, 41-49 1987

[9] Deb K, Goldberg D E. An investigation of niche and species formation in genetic function optimization, In: Proc. 3rd International Conference on Genetic Algorithms. 42-50, 1989