Minimum Spanning Tree Problem Research based on Genetic Algorithm

Hong Liu[1,2]
[1]College of Information Engineering, Zhejiang University of Technology
[2] College of Computer science and Information Engineering Zhejiang Gongshang University
Hangzhou, P.R.China
LLh@mail.hzic.edu.cn

Gengui Zhou
College of Business and Administration
Zhejiang University of Technology
HangZhou, P.R.China
ggzhou@zjut.edu.cn

Abstract—**Minimum Spanning Tree (MST) problem is of high importance in network optimization, but it is also difficult for the traditional network optimization technique to deal with. In this paper, self-adaptive genetic algorithm (GA) approach is developed to deal with this problem. Without neglecting its network topology, the proposed method adopts the Prüfer number as the tree encoding and self-adaptation is used to enable strategy parameters to evolve along with the evolutionary process. Compared with the existing algorithm, the numerical analysis shows the efficiency and effectiveness of such self-adaptive GA approach on the MST problem.**

Keywords- Genetic algorithms, Minimum spanning tree, Prüfer number

I. INTRODUCTION

The Minimum Spanning Tree (MST) problem is to find a least cost spanning tree in an edge weighted graph. This problem has been well studied and widely applied to many combinatorial optimization problems such as transportation problem, telecommunication network design, distribution systems and so on. With the development of computer technology, many efficient polynomial-time algorithms for this problem have been developed by Dijkstra[1], Kruskal[2], Prim[3] and Sollin[4]. More recently, as to the MST problems, varieties of heuristics including genetic algorithms (GAs) methods have been used, such as the probabilistic MST by Bertsimas [5] and the quadratic MST by Xu [6]. These GA approaches, in particular, have shown to be highly effective in solving the probabilistic MST.

In this paper, we present an approach by using the genetic algorithms (GAs) based on Self-Adaptive. For adaptation to the evolutionary process, we develop a tree-based genetic representation to encode the candidate solutions of the MST problem. It has a tree structure and guarantees that the candidate solutions are always feasible solutions after some modification on the number of leaves. This tree encoding has the property of inheriting good properties from parents and thus provides a great chance of evolving to a highly fit structure or solution in the evolutionary process. In taking Prim's result as its lower bound , and Fernandes and Gouveia's result as its upper bound, the GA results on numberical experiments clearly show the high effectiveness of the proposed GA approach on the MST problem.

The paper is organized as follows: In Section 2 the MST problem is defined. Some techniques about it are described in Section 3. Section 3 also discusses in detail our GA approach for the MST problem. In Section 4 the numerical experiments on the proposed method are studied and the conclusion follows in Section 5.

II. PROBLEM DESCRIPTION

Consider a connected graph G=(V, E), where V={1,2,…,n}is a finite set of vertices representing terminals or telecommunication stations etc., and $E=\{(i,j)|i,j\in V\}$ is a finite set of edges representing connections between these terminals or stations. Each edge has an associated positive real number denoted by $W=\{w_{ij}|edge(i,j)\in E\}$ representing distance, cost and so on. For a subset of node $S(\subseteq V)$ we define $E(S)=\{(i,j)|\ i,j\in S\}$ to be the edges whose end points are both in S. Also, we define the following binary decision variables for all edges $(i,j)\in E$.

Let $X=\{x_{ij}\}$be defined as follows:

$$x_{ij} = \begin{cases} 1, & \textit{if edge}(i,j) \textit{ is selected in a spanning tree} \\ 0, & \textit{otherwise} \end{cases}$$

Then a spanning tree of graph G can be expressed by the vector X. Let T be the set of all such vectors corresponding to spanning trees in graph G, the well-known MST problem can be formulated as:

$$Min\ Z(x) = \sum_{i=1}^{n-1} \sum_{j=2}^{n} w_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n-1} \sum_{j=2}^{n} x_{ij} = n-1 \tag{2}$$

$$\sum_{i\in S} \sum_{\substack{j\in S \\ j>1}} x_{ij} \le |S|-1,\ S \subseteq V\setminus\{1\},\ |S| \ge 2 \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} \ge 1,\ i = 1,2,...,n \tag{4}$$

$$x_{ij} = 0\ or\ 1,\ i = 1,2,...,n-1,\ j = 2,3,....,n \tag{5}$$

In this formulation, (2) is true of all spanning tree: a tree with n nodes must have n-1 edges. Equation (3) are some of the standard rank inequalities for spanning tree: if more than |S|-1 edges connect the nodes of a subset S, then that set of edges must contain a cycle. Unfortunately, there are $O(2^n)$ constraints for them, leading to a very large integer programming even for moderate values of n. Equation (4) gives the degree value on all vertices.

III. GA APPROACH

Unlike conventional combinatorial optimization techniques, the GA requires neither heuristics nor knowledge of key properties of the problems to be solved. By imitating the process of natural selection in nature, the GA starts from a set of candidate solutions of the problem instead of a single one, improves them step by step through biological evolutionary process like crossover, mutation, etc., and gets to the optimal solutions in most occasion. The whole process is operated randomly not deterministic. Thus, the GA is algorithm in performance of population and probability search in optimization and capable of dealing with any kind of objective and constraint functions.

A. *Chromosome representation*

For the GA approach on any problem, it is important to figure out the adequate chromosome representation of the problem. And how to encode a tree or what information should be stored on the chromosomes is also critical for the GA approach. In this paper we focus on the prüfer number encoding.

One of the classical theorems in graphical enumeration is Cayley's theorem that there are n(n-2) distinct labeled trees on a complete graph with n vertices. Priifer provided a constructive proof of Cayley's theorem by establishing a one-to-one correspondence between such trees and the set of ah strings of n-2 digits. This means that we can use only n-2 digits permutation to uniquely represent a tree with n vertices where each digit is an integer between 1 and n inclusive. This permutation is usually known as the Prüfer number.

For any tree in a complete graph, there are always at least two leaf vertices. By leaf vertex is meant that there is only one edge connected to the vertex. Based on this observation we can easily construct the encoding according to the following procedure: Here we build the permutation by appending digits to the right, and thus the permutation is built and read from left to right.

Procedure: Encoding

Step 1. Let vertex i be the smallest labeled leaf vertex in a labeled tree T.

Step 2. Let j be the first digit in the encoding if vertex j is incident to vertex i. Here we build the encoding by appending digits to the right, and thus the encoding is built and read from left to right.

Step 3. Remove vertex i and the edge from i to j, we have a tree with n - 1 vertices.

Step 4. Repeat the above steps until one edge is left and produce the Prüfer number or encoding with n-2 digits between integer 1 and n inclusive.

It is also possible to generate a unique tree from a Prüfer number via the following procedure:

Procedure: Decoding

Step 1. Let P be the original Prüfer number, be the set of all vertices not included in P, and be designated as eligible for consideration.

Step 2. Let i be the eligible vertex with the smallest label in .Let j be the leftmost digit of P. Add the edge from i to j into the tree. Remove i from and j from P. If j does not occur anywhere in the remaining part of P, designate j as eligible and put it into .Repeat the process until no digits are left in P.

Step3. If no digits remain in P, there are exactly two vertices, r and s, still eligible for consideration. Add the edge from r to s into the tree and form a tree with n - 1 edge.

An example is given to illustrate this encoding. The Prüfer number [2 5 6 8 2 5] corresponds to a spanning tree on a 8-vertex complete graph represented in Fig. 1. The construction of the Prüfer number is described as follows: locate the leaf vertex having the smallest label. In this case, it is vertex 1. Since vertex 2 (the only vertex) is incident to vertex 1 in the tree, assign 2 to the first digit in the permutation, then remove vertex 1 and edge (1,2). Now vertex 3 is the smallest labeled leaf vertex and vertex 5 is incident to it, assign 5 to the second digit in the permutation and then remove vertex 3 and edge (3,5). Repeat the process on the subtree until edge (5,8) is left and the Prüfer number of this tree with 6 digits is finally produced.

Conversely, we can construct the corresponding tree using the Prüfer number, P=[2 5 6 8 2 5]. As the vertex 1, 3, 4 and 7 are not included in P, they are set as ={1,3,4,7}. Firstly, vertex 1 is the smallest digit in and vertex 2 is the leftmost digit in P. Add edge (1,2) to the tree, remove vertex 1 from and the leftmost digit 2 of P leaving P =[5 6 8 2 5]. Secondly, vertex 3 is now the smallest digit in and vertex 5 is the leftmost digit in remaining P. Then add edge (3,5) to the tree, remove vertex 3 from and the leftmost digit 5 from P leaving P=[6 8 2 5]. Repeat the process until P =[5] and vertices 2 and 8 are in . Add edge (2,5) to the tree, remove the last digit 5 and P and vertex 2 from . As vertex 5 now no longer occurs in the remaining P, it needs putting into leaving ={5,8}. Finally, P is empty and only vertices 5 and 8 are in . Thus add edge (5,8) to the tree and stop. The tree in Fig. 1 is formed.

It is clearly that the Prüfer number is a skillful encoding for spanning tree. The encoding length is only n-2, the search space size is nnÿ2 and the relation between Prüfer number and spanning tree is one-to-one mapping, therefore the probability to randomly produce a spanning tree is definitely 1, which means that this encoding is capable of equally and uniquely representing all possible trees and any initial population or offspring from crossover and mutation operation and still keep a tree. As to the MST problem, we can easily calculate each objective for each chromosome or individual according to its corresponding Prüfer number. Therefore, the Prüfer number encoding is adopted as the chromosome representation.

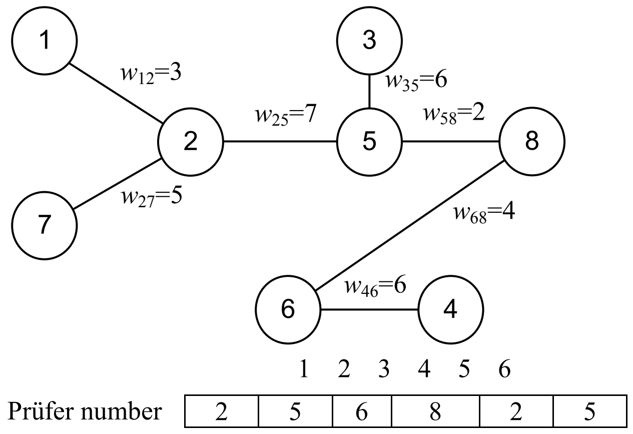

Figure 1. A tree and its Prüfer number

B. *Crossover and mutation*

Crossover and mutation are two crucial factors in the biological evolutionary process. In the GA approach they guarantee that the population or chromosomes have a great chance to be evolved to the optimal solution. Because Prüfer number encoding can always represent a tree after any crossover or mutation operations, here simply we use the uniform one-cut point crossover operator which has been shown to be superior to traditional crossover strategies for combinatorial optimization problem. Uniform crossover firstly generates a random crossover point and then exchanges relative genes between parents according to the cut point. The operation can be illustrated in Fig.2.

Mutation is a background operator, which produces spontaneous random changes in various chromosomes. A simple way to achieve mutation would be to alter one or more genes. Here the mutation is performed as random perturbation within the permissive range of integers from 1 to n(n is the number of vertices in graph) as illustrated in Fig. 3.

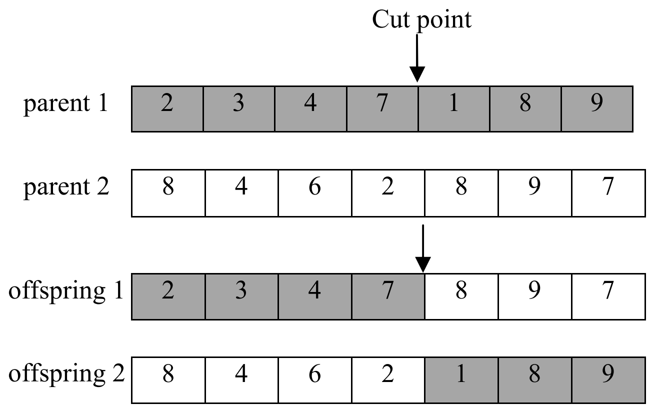

Figure 2. Illustration of crossover operation

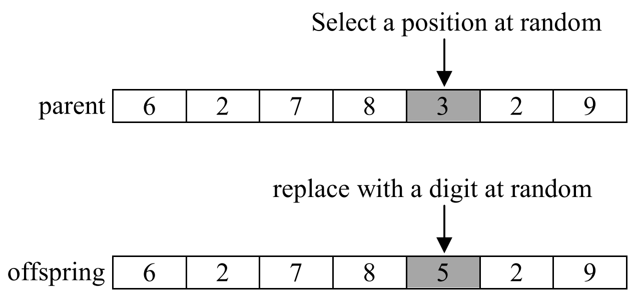

Figure 3. Illustration of mutation operation

C. *Evaluation and selection*

In the GA approach, the evaluation and selection play a very important role. The evaluation is to calculate the fitness for each chromosome and the selection is to select chromosomes for the next generation according to their own fitness values. The higher the fitness value of an individual, the higher its chances of survival and reproduction and the larger its representation in the subsequent generation If the crossover and mutation are regarded as the exploration for the GA to make the chromosomes be of diversity, then the evaluation and selection can be regarded as the exploitation for the GA to converge to the optimal or neat-optimal solution. In our GA approach for the MST problem, the evaluation procedure consists of two steps:

- Convert a chromosome into a tree, in which includes the transformation of the Prüfer number into a tree in the form of edge set;
- Calculate the total costs of the tree, in which the fitness for each chromosome can be directly calculated according to the objective in equation (1);
- Repeat the procedure on all individuals.

As to selection procedure, a mixed strategy with ($\mu+\lambda$)-selection strategy and roulette wheel selection is used. The ($\mu+\lambda$)-selection can enforce the best chromosomes into the next generation. The mixed strategy in our GA approach selects u best chromosomes from u parents and λ offspring.

Here the roulette wheel is constructed with the following steps:

Input: population P(t-1), C(t-1)

Output: population P(t), C(t)

Step 1: calculate the total fitness for the population

$$F=\sum_{k=1}^{popsize} eval(v_k)$$

Step2: calculate selection probability p_k for each chromosome v_k

$$p_k=\frac{eval(v_k)}{F}, \quad k=1,2,\cdots,popsize$$

Step 3: calculate cumulative probability q_k for each chromosome v_k

$$q_k=\sum_{j=1}^{k} p_j, \quad k=1,2,\cdots,popsize$$

Step 4: generate a random number r from the range [0,1]

Step 5: if $r \le q_1$, then select the first chromosome v_1; otherwise, select the kth chromosome $v_k(2\le k \le popsize)$ such that $q_{k-1} \le r \le q_k$.

D. MST Self-Adaptive Genetic Algorithm

The overall pseudo-code procedure for the MST problem is outlined as follows:

```
Procedure: Self-Adaptive Genetic algorithm for MST
begin
    t ←0;
    initialize P(t) by random generation based on system
    constraints;
    fitness eval(P);
    while (not termination condition) do
        crossover P(t) to yield C(t) by self-adaptive
        uniform crossover;
        mutation P(t) to yield C(t) by uniform mutation;
        fitness eval(C);
        if $u' < u + \lambda$
            select P(t+1) from P(t) and C(t)
        else
            select P(t+1) from P(t) and C(t) by roulette
            wheel;
        end
        t←t+1;
    end
    output best solution;
end
```

IV. Experiment Analysis

The performance of MST by the GAs approach is tested on seven numerical examples of the 10-vertex to 50-vertex, and 100-vertex, 200-vertex complete graph generated randomly, their runtimes are shown as table I, which are all run on matlab. The weights defined on edges for the graph are generated randomly and respectively distributed uniformly over [10,100]. The parameters for the GA are set as follows: population size pop_size =200; crossover probability pc =0.2; mutation probability pm= 0.05; maximum generation max_gen=500.

TABLE I. RUNTIME RESULT BY THE GA

Problem size(Vertices)	**Three algorithm runtime**		
	GA approach Runtime(s)	***Kruskal O(elog(e)) (machine unit time)***	***Prim $O(n^2)$(machine unit time)***
10	16.33	10	100
20	32.78	26.02	400
30	50.58	44.31	600
40	67.86	64.08	1600
50	84.92	84.95	2500
100	178.01	200	10000

Then, some numerical examples result of 10-vertex to 50-vertex are shown as Fig.4. to Fig.8.. In these figure, x coordinate axis means generation value, of which max_gen is 500, and y coordinate axis means fitness value of each generation. From the followed figure, we can see that such approach can get convergence

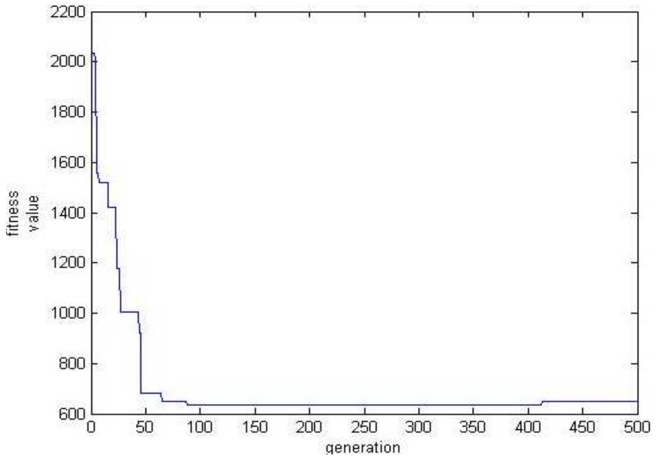

Figure 4. 10-vertexs result

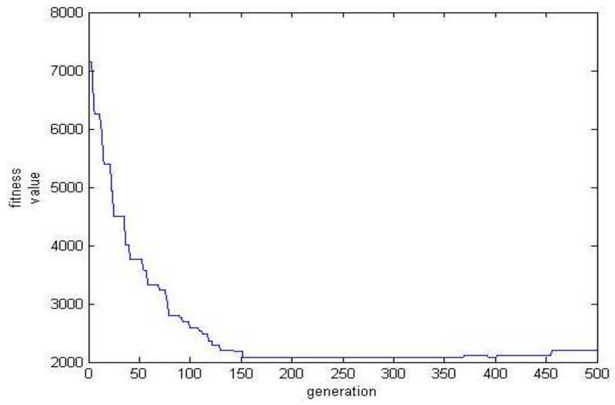

Figure 5. 20-vertexs result

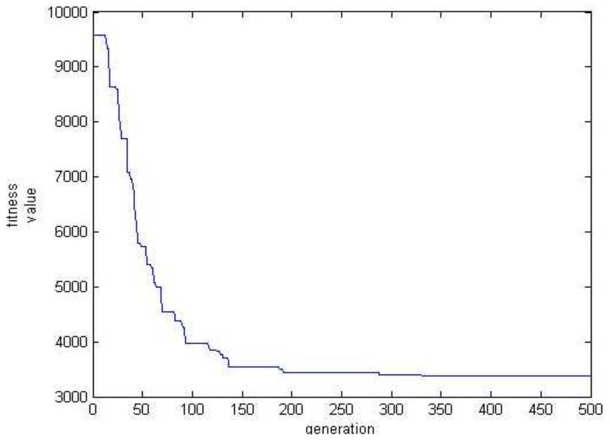

Figure 6. 30-vertexs result

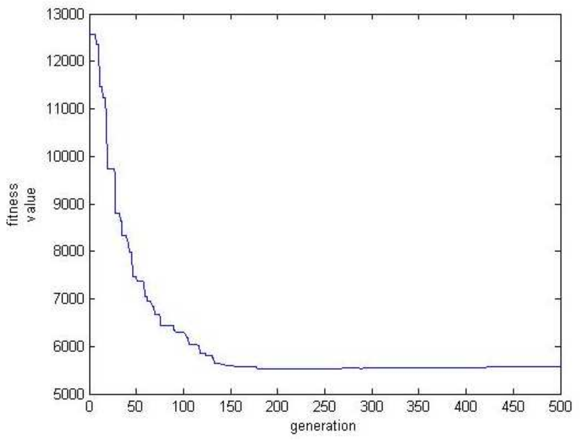

Figure 7. 40-vertexs result

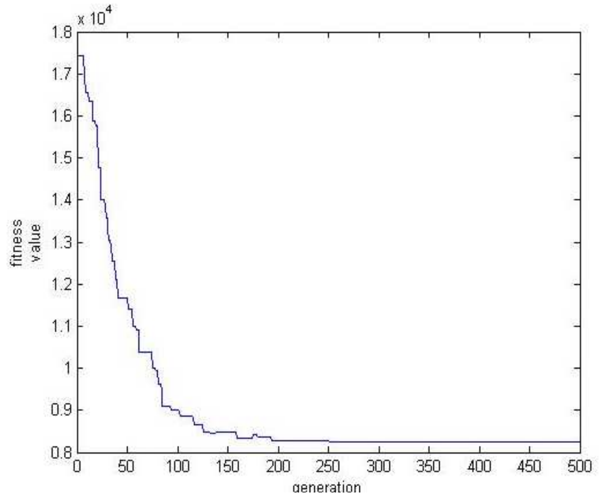

Figure 8. 50-vertexs result

V. Conclusion

This paper researched the self-adaptive GA approach for the MST problem. In order to encode a corresponding tree structure solution for the problem, we developed a tree-based permutation that is able to represent all possible rooted tree, such tree coding has the property that it inherits good properties from parent, and unlike traditional incremental approach to MST format, such approach directly performs a search on coding space and evolves them to the Pareto optimal solutions through the genetic operations. Combined with the GA techniques and self-adaptive technique, the results of the proposed method show its high effectiveness in dealing with the MST problem. Although this paper has only dealt with the classical MST, it is easy to extend the proposed method to solve those degree-constrained MST, stochastic MST, probabilistic MST and quadratic MST problems with multi-criteria.

References

[1] E.W. Dijkstra, A note on two problems in connection with graphs, Numerische Mathematik 1 (1959) 269–271.

[2] J.B. Kruskal, Jr., On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. ACM 7/1 (1956) 48–50.

[3] R.C. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal 36 (1957) 1389–1401.

[4] M. Sollin, Le trace de canalisation, in: C. Berge, A. Ghouilla-Houri (Eds.), Programming, Games, and Trans-portation Networks, Wiley, New York, 1965.

[5] D.J. Bertsimas, The probabilistic minimum spanning tree problem, Networks 20 (1990) 245–275.

[6] W. Xu, On the quadratic minimum spanning tree problem in: M. Gen, W. Xu (Eds.), Proceedings of 1995 Japan China International Workshops on Information Systems Ashikaga, Japan, 1995, pp. 141–148.

[7] İbrahim Akgün, Barbaros Ç. Tansel. Min-degree constrained minimum spanning tree problem: New formulation via Miller–Tucker–Zemlin constraints. Computers & Operations Research, In Press, Corrected Proof, Available online 24 March 2009.

[8] Raja Jothi, Balaji Raghavachari. Degree-bounded minimum spanning trees. Discrete Applied Mathematics, Volume 157, Issue 5, 6 March 2009, Pages 960-970.

[9] Alok Singh. An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Applied Soft Computing, Volume 9, Issue 2, March 2009, Pages 625-631.

[10] Guolong Chen, Shuili Chen, Wenzhong Guo, Huowang Chen. The multi-criteria minimum spanning tree problem based genetic algorithm. Information Sciences, Volume 177, Issue 22, 15 November 2007, Pages 5050-5063.