A Novel Particle Swarm Optimization for the Steiner Tree Problem in Graphs

Wen-Liang Zhong, *Student Member IEEE*, Jian Huang and Jun Zhang (corresponding author), *MIEEE*

***Abstract*—The Steiner tree problem (STP) in graphs is a special but essential case of multiple destination routing (MDR) problems, which focuses on finding a minimal spanning tree (MST) that connecting the source and destinations. It has been proved to be an NP-hard problem. Particle swarm optimization (PSO) is an important swarm intelligent algorithm with fast convergence speed and easy implementation. In this paper, a novel discrete PSO for the STP (DPSO-STP), with the concept that the particle is guided by social and self cognition, is proposed. Different from the standard PSO, the DPSO-STP includes four parts: 1. two preprocessing operations are introduced, which are to construct a complete graph and to calculate each node's total distance from itself to the source and destination nodes; 2.the position of a particle is represented as a binary string, where 1 stands for the selected nodes and 0 denotes the opposite; 3. several novel update operations, including new mutation factor c_3, are adopted for the binary string; 4. when generating a MST from a binary string, a modified Prim's algorithm and a trimming strategy are employed. The experiments based on the benchmarks from category B, C of STP in the OR-library have been carried out to demonstrate the effectiveness of the proposed algorithm. Compared with traditional heuristic algorithms, such as shortest path heuristic (SPH), average distance heuristic (ADH), etc, the DPSO obtains more promising results. And it also performs better than the other iteration based algorithm, with much less computation. The discussion to extend the algorithm to other MDR problems is also given.**

I. Introduction

WITH the development of Internet, more and more complicated applications require figuring out the proper routing to transfer data from one source to several destination nodes, such as network meeting and video-on-demand services, etc. Analyses on this are called multiple destination routing (MDR) problems [1]. The Steiner tree problem in graphs (STP) is one of the most important MDR problems, which is NP-hard [1]. Given a network graph, the object of STP is to find out a minimal spanning tree (MST) that connecting the source and all the destination nodes (see section II for more detail).

Many heuristic approaches for STP have been represented in the last twenty years [1]-[9]. The SPH algorithm [2] creates the MST by adding the nearest node one by one until the tree contains the source and all destination nodes. The distance network heuristic algorithm [3] figures out the distance complete network first, and then creates and trims a MST on it. The ADH algorithm [4] focuses on seeking the intermediate nodes and performs well. The other feasible algorithms include dual ascent heuristic [5], direct convergence heuristic (DCH) [6], simulated annealing [7], genetic algorithm (GA) [1][8] and ant colony optimization [9], *etc*. The traditional algorithms [2]-[6] are very fast because they often generate a few, even only one, minimal spanning tree. But they often obtain poor results, especially when the network contains lots of nodes. On the other hand, the iteration based algorithms [1][7]-[9] achieve better results, but cost more computational time.

Particle swarm optimization (PSO), which was first proposed by Kennedy and Eberhart [10] in 1995, employs an iteration based search to locate global optimal solution. Simulating simplified social system of birds flock, the algorithm has been successfully applied to many continuous optimization problems as a swarm intelligence algorithm [11]-[15]. Due to its easy implementation and fast convergence, the PSO becomes more and more popular.

However, the standard PSO is designed for continuous problems, so it can't be applied to discrete problems directly. Since 1996, several discrete PSOs for binary optimization problems have been presented [16]-[18]. As to the MDR problems, the PSOs have obtained promising results [19]-[22], but none of them is for the STP.

In this paper, we propose a novel discrete PSO to solve the STP. Two important preprocessing operations are adopted. One is to construct the complete graph (any two nodes connected directly), and the other is to calculate the total distance for each node from itself to the source and destination nodes. The position of a particle is represented as a binary string, where 1 stands for the selected nodes and 0 for the opposite. Each binary string represents one or more MSTs with the same weight cost, which are the solution for STP. Consequently, the update equations are modified to fit this change. To prevent the algorithm from being trapped in a local optimal, a new parameter c_3 called mutation factor is introduced. When evaluating the fitness of a particle, a modified Prim's algorithm [23] is employed to generate a MST and a trimming operation is used to cut off the redundant nodes. In a word, the novel PSO is employed to optimize the binary string and two operations are used to generate corresponding MSTs. The results of the problems from category B and C in the OR-library indicate that the

This work was supported by NSF of China Project No.60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

Authors are with Department of Computer Science, SUN Yat-sen University, Guangzhou, P.R.China, (Jun Zhang is the corresponding author, email:junzhang@ieee.org)

978-1-4244-1823-7/08/$25.00©2008 IEEE

algorithm is promising. The proposed algorithm obtains better solutions than traditional heuristic algorithms and the GA [1].

In section II, the background of standard PSO algorithm is introduced and the STP is defined formally. Section III describes the DPSO-STP and section IV gives the experiment results. Section V is the discussion to extend the algorithm for other MDR problems. And the last section draws a conclusion.

II. The Stand Particle Swarm Optimization and The Sterner tree problem in graphs

A. The standard particle swarm optimization

In the standard PSO model, the particle, which can be regarded as a bird, has a position X_i and a velocity V_i, both of which are D-dimensional vectors denoting a solution and a moving step, respectively. Here D is the number of dimension of the problem and the variable i stands for the i-th particle. Similar to the GAs, PSO evolves a population of n particles to search the best solution based on generations. Different from GAs, a particle adjusts its position X_i without crossover or mutation operators. It flies through the solution space attracted by *gbest* and *pbest*$_i$, which are the best positions that the whole population and the i-th particle have reached, respectively.

```
Initialization
while(termination criterion is not met)
  for($i$ = 1 to size)
    update ($V_i$, $X_i$);
    evaluating $f(X_i)$;
    if( $f(X_i) < f(pbest_i)$)
      $pbest_i \leftarrow X_i$
      if ($f(X_i) < f(gbest)$)
        $gbest \leftarrow X_i$
      end if
    end if
  end for
end while
Report the result
```

Fig. 1. The pseudo code of standard PSO for the minimal optimization

Fig. 1 is the pseudo code of the standard PSO for the minimal optimization problem. First, the parameters are initialized and every particle is given a random position and velocity. And then, the positions are saved as *pbest*$_i$, and evaluated by function f to elect *gbest*. During the generations, a particle is updated through these two equations:

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (pbest_i^k - X_i^k) + c_2 r_2 (gbest^k - X_i^k) \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

where k denote the k-th generation. The first equation can be divided to three parts by operator "+". The parameter ω, called inertia weight [11][12], helps to control the influence of the pre-velocity. The other two parts are self-cognitive and social-cognitive components. Both c_1 and c_2 are constant real positive number, which are used to direct the i-th particle to *pbest*$_i$ and *gbest*. And r_1, r_2 are random real numbers belonging to [0, 1]. After updated, the new positions will be evaluated again, and *pbest*$_i$ and *gbest* may be replaced if necessary.

B. The Sterner tree problem in graphs

The problem is formally defined as follow:

Definition 1: G= (V, E, C) is a weighted, undirected and connected graph, where V, E are the set of nodes (also called vertexes) and edges, respectively, and C is the weight function. That is, $C(e_{ij}) \to R^+$ is the weight of edge $e_{ij} \in E$, which connects nodes i and j.

Definition 2: V_d is the set of destination nodes and s is the source node. Because the STP only focuses on searching a tree connecting the source and all the destination nodes, V_d and s make no difference. So we call both of them objective nodes and treat them indiscriminatingly in the proposed algorithm.

Definition 3: The object of the STP is to find out a sub-tree T in graph G, which connects all objective nodes with the minimal cost of weight. That is,

$$f = \min \sum_{e \in T} C(e)$$

For any solution tree T that is not a MST, there is always at least one MST T' with the same nodes as T, and its weight cost is smaller than T. So it is obvious that the optimal sub-tree must be a MST. Because there are several algorithms to generate a MST in polynomial time, such as the Prim's and the Kruskal's algorithm [23], the key to solve STP is finding out the intermediate nodes (nonobjective nodes in the solution tree) as Leung *et al*. [1] pointed out.

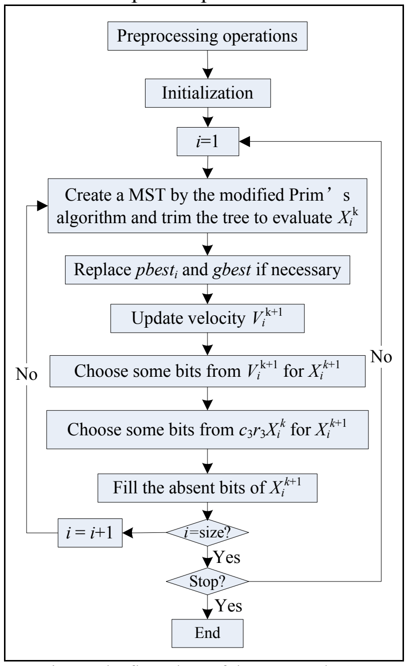
III. The Discrete particle swarm optimization for the Steiner tree problem

Because the DPSO-STP is more complexity than the standard PSO, we give its flowchart in Fig. 2. There are four key components in the proposed algorithm, which are the preprocessing operations, the representation, the update operations and the improving strategies, respectively.

A. The preprocessing operations

The solution of the STP is always a connected tree. However, because the graph G is not guaranteed as a complete graph, some binary strings may represent unconnected nodes. In other words, it is unable to generate a MST in the graph according a binary string occasionally. To overcome this obstacle, the Floyd's algorithm [23] is applied to figure out the shortest distance and the path between any two indirectly connected nodes in the graph. As a result, the graph has been transferred to complete graph.

The other preprocessing operation is that for each node, the total distance from itself to all the objective nodes is figured out. And then the nodes are sorted according to the total distance in descending order. This work is the preparation for *Modification* 7 in the update operations.



Preprocessing operations
Initialization
i=1
Create a MST by the modified Prim's algorithm and trim the tree to evaluate X_i^k
Replace $pbest_i$ and $gbest$ if necessary
Update velocity V_i^{k+1}
Choose some bits from V_i^{k+1} for X_i^{k+1}
Choose some bits from $c_3r_3X_i^k$ for X_i^{k+1}
Fill the absent bits of X_i^{k+1}
i=size?
$i = i$+1
No
Yes
Stop?
No
Yes
End

Fig. 2. The flow chart of the proposed PSO

B. *The representation*

As mention before, the key for the STP is to find out the proper intermediate nodes and it is easy to generate a MST based on a binary string in polynomial time. Consequently, the position of a particle is encoded as a binary string as $X_i^k = (x_{i1}, x_{i2}, \cdots, x_{in})$ in the proposed algorithm where x_{ij} is 0 or 1 and k, j denote the k-th generation and the j-th node in the graph, respectively. If x_{ij} =1, the j-th node is selected as an intermediate node or it is an objective node. As a result, the object of STP is equal to obtaining a proper binary string, so the STP has been converted to a 0/1 optimization problem.

Remark 1: The bits stand for the objective nodes are always set to 1.

The velocities are represented as

$$V = \begin{Bmatrix} v_1^0, v_2^0, \cdots, v_n^0 \\ v_1^1, v_2^1, \cdots, v_n^1 \end{Bmatrix},$$

where v_j^0 and v_j^1 are real numbers in the internal of [0,1] and denote the probabilities of the j-th bit to be 0 or 1. According *Modification* 5 below, the sum of v_j^0 and v_j^1 is not necessary to equal to 1.

C. *The update equations*

In the iteration, the particles will be updated and evaluated repeatedly until the termination criterion is met. Since the standard PSO is designed for the continuous problems, the update equations (1) and (2) in the DPSO-STP have to be redefined for the STP by following 7 modifications.

Modification 1: The result of subtracting operator "–" between two positions (binary string) X_1 and X_2 in (1) is defined as a velocity, in which v_j^b is set to 1 if the j-th bit in X_1 is b while it is not in X_2. Otherwise, v_j^b is set to 0.

Modification 2: In equation (1), the result V of subtraction is multiplied by $c_j r_j$ (j = 1, 2) , where c_j is a const real number and r_j is a random real number belonging to [0, 1]. It is the same in the proposed algorithm. As a result, each element is probably multiplied by a unique real number belonging to [0, c_j], because the r_j is spawn randomly.

Remark: if v_j^b is greater than 1.0 after multiplication, it will be limited to 1.

Remark: as to ωV, the result is that each element v_j^b in V is multiplied by the same ω, respectively.

Modification 3: The result of operator "+" in equation (1) between two velocities is a new velocity, that is $V = V_1 + V_2$. The v_j^b in V is the greater one between v_{1j}^b and v_{2j}^b.

Following the three modifications above, the first equation is totally redefined for the STP. An example is given as follow.

Assume ω =0.5, c_1=c_2 =2.

$pbest_i^k$ = 1,1,0,0,1,1,1,0 ,

$gbest^k$ = 1,1,1,0,1,0,1,0 ,

X_i^k = 1,1,1,0,1,1,0,0 ,

$$V_i^k = \begin{Bmatrix} 0.4,0, & 0,0.2,0,0,1,0 \\ 0.6,0,0.2,0.8,0,0,0,1 \end{Bmatrix}$$

so

$$V_{i1}^{k+1} = c_1 r_1 (pbest_i^k - X_i^k) = \begin{Bmatrix} 0,0,0.8,0,0,0,0,0 \\ 0,0, & 0,0,0,0,1,0 \end{Bmatrix},$$ suppose c_1r_1 = 0.8, 1.5 for each v_j^b ($v_j^b \neq 0$), respectively.

$$V_{i2}^{k+1} = c_2 r_2 (gbest^k - X_i^k) = \begin{Bmatrix} 0,0,0,0,0,0.7, & 0,0 \\ 0,0,0,0,0, & 0,0.6,0 \end{Bmatrix},$$ suppose c_2r_2 = 0.7, 0.6 for each v_j^b ($v_j^b \neq 0$), respectively.

and then

$$V_i^{k+1} = \omega V_i^k + V_{i1}^{k+1} + V_{i2}^{k+1} = \begin{Bmatrix} 0.2,0,0.8,0.1,0,0.7,0.5, & 0 \\ 0.3,0,0.1,0.4,0, & 0, & 1,0.5 \end{Bmatrix}.$$

The second update equation is divided to four modifications.

Modification 4: The next position X_i^{k+1} is initialized as an empty string.

Modification 5: Generate a random number $\alpha \in [0,1]$, and if v_j^b in V_i^{k+1} is greater than α, the j-th bit of X_i^{k+1} is set to b. If both v_j^0 and v_j^1 are greater than α, the j-th bit is set to 0 or 1 randomly. And if both v_j^0 and v_j^1 are smaller than α, the j-th bit is set to "-", which stands for the absent bits. After these two steps, X_i^{k+1} is a string with several absent bits.

Modification 6: A new parameter c_3 is added to equation (2) to keep balanced between exploitation and exploration. That is, the second update equation is modified as

$$X_i^{k+1} = V_i^{k+1} \otimes c_3 r_3 X_i^k \tag{3}$$

For each absent bit in X_i^k, a random real number r_3 belonging to [0, 1] is spawned, and if $c_3 r_3 > \alpha$, the corresponding bit in X_i^k will be copied to X_i^{k+1}. The reason of not copying all absent bits from X_i^k is that, after some generations, $pbest_i$, $gbest$ and X_i will become more and more similar, like other evolutionary algorithms. As to the standard PSO, X_i will keep tiny moving as local search to get higher precision, especially in unimodal function optimization. However, in the DPSO-STP, the number of v_j^b=0 in V_j^{k+1} will increase rapidly. So if choosing as many as possible bits in X_i^k, almost all elements in X_i^{k+1} will come from X_i^k, and then the algorithm stagnates. With c_3, some absent bits in X_i^{k+1} won't be filled until the operation in *Modification* 7, like mutation. So c_3 is called mutation factor. What the algorithm benefit from it is showed by experiments in the next section.

Modification 7: Because several bits are perhaps absent in X_i^{k+1}, a strategy to fill the blanks is adopted. For the *i*-th absent bit a random real number t belonging to [0, 1] is generated, if $t < rand_i / n$, where $rank_i$ is the rank of the *i*-th node that figured out in the preprocessing operations, and n is the number of nodes. Otherwise the *i*-th is set to 0. So for each node, the closer to all the objectives it is, the more opportunity to be selected it will have.

Remark: The bits stand for the objective nodes are always set to 1.

Assume α = 0.5, c_3=2, and V_i^{k+1} is given in the example after *Modification* 3. So a possible case is

$X_i^{k+1} = (-,-,0,-,-,0,0,1)$

Assume $c_3 r_3$ = 1, 0.5, 0.4, 0.7 for the 1st, 2nd, 4th, 5th bits, respectively, then the 1st, 2nd and 5th bits in X_i^k are copied to X_i^{k+1}, so

$X_i^{k+1} = (1,1,0,-,1,0,0,1)$.

Suppose $t < rank_4/n$, then X_i^{k+1} is $(1,1,0,0,1,0,0,1)$.

D. The modified Prim's algorithm and the trimming strategy

According to the seven modifications, the X_i^{k+1} is a new *n*-bit binary string. Because the solution of STP is a minimal spanning tree, we have to design a method to convert a binary string to a corresponding MST, and then the cost of the MST is the fitness of the particle. The Prim's algorithm [23] is an effective algorithm to do such a work in a connected graph. However, because it is uncertain that all selected node in X_i^{k+1} are connected directly, we modify the Prim's algorithm as follow:

Definition 4: A real edge is an edge connecting two nodes directly and given in the STP.

Definition 5: A virtual edge is the shortest path connecting two nodes, which contain at least one intermediate node. It is figured out by Floyd's algorithm [23] in the preprocessing operation. And restoring a virtual edge means using the intermediate nodes and real edges to replace it.

Definition 6: *S* is the set containing nodes already involved in the MST. And ~*S* is the set whose members are candidate nodes for the MST.

The modified Prim's algorithm is:

a. Choose a node randomly to *S*, and put the other nodes into ~*S*;

b. Find out the shortest real edge connecting one node in *S* and the other in ~*S*. If succeed, move the selected node from ~*S* to *S* and repeat this step until ~*S* is empty. If no real edges exist between *S* and ~*S*, go to c.

c. Find out the shortest virtual edge connecting one node in *S* and the other in ~*S*, and then move the selected node from ~*S* to *S* and record the virtual edge, then go to b. If ~*S* is empty, go to d.

d. Restore the recorded virtual edges and the modified Prim's algorithm finishes.

The MST containing all selected nodes in X_i^{k+1} has been generated. Due to the restoring operation, some additional nodes may be also involved in the MST.

It is obvious that the MST may contain some redundant leaves, which are not objective nodes and increase some unnecessary cost. So tree is trimmed by follows:

a. Drop the redundant leaves.

b. If the number of dropped leaves is more than zero, go to a, otherwise the trimming is over.

As a result, a MST without redundant leaves is generated and its cost is the fitness value of the position X_i^{k+1}. As some nodes have been added in restoring or deleted in trimming, the binary string of X_i^{k+1} should been modified to respond it.

Fig. 3 (a) is the network of Steiner tree problem, where the blocks denote the objective nodes and the dots stand for routers. And (b) & (c) are the MST of X_i^{k+1} before and after restoring and trimming, respectively. The dashed in (b) is the virtual edge and the 6th node is selected in (c) because of restoring and the 8th one is deleted in trimming. The cost of MST in (c) is the fitness of X_i^{k+1} and the binary string is $(1,1,0,0,1,1,1,0)$ at the end.

According steps above, the PSO is applied to the STP and called the DPSO-STP. The experimental results are given in the next section.

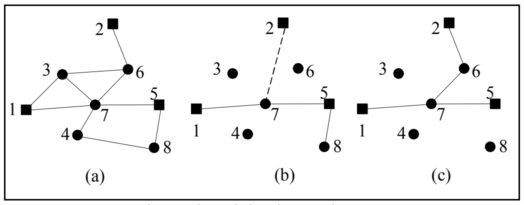


Fig. 3 The minimal spanning tree

IV. Experimental results

A. Relative setting

The experiments based on the STPs in graphs from category B and C in the OR-library have been carried out. (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). The details of them are given in Table 1 and 2. The B Steiner tree problems are based on the network with 50, 75 and 100 nodes. And the networks in C contain 500 nodes. In the two tables, $|V_G|$, $|E_G|$ and $|V_D|$ stand for the number of nodes, edges and objective nodes, respectively. And "OPT" denotes the optimal value for each test case given by the OR-library.

Table 1. The detail of Category B problems

No.	$\|V_G\|$	$\|E_G\|$	$\|V_D\|$	OPT
B01	50	63	9	82
B02	50	63	13	83
B03	50	63	25	138
B04	50	100	9	59
B05	50	100	13	61
B06	50	100	25	122
B07	75	94	13	111
B08	75	94	19	104
B09	75	94	38	220
B10	75	150	13	86
B11	75	150	19	88
B12	75	150	38	174
B13	100	125	17	165
B14	100	125	25	235
B15	100	125	50	318
B16	100	200	17	127
B17	100	200	25	131
B18	100	200	50	218

Table 2. The detail of Category C problems

No.	$\|V_G\|$	$\|E_G\|$	$\|V_D\|$	OPT
C01	500	625	5	85
C02	500	625	10	144
C03	500	625	83	754
C04	500	625	125	1079
C05	500	625	250	1579
C06	500	1000	5	55
C07	500	1000	10	102
C08	500	1000	83	509
C09	500	1000	125	707
C10	500	1000	250	1093
C11	500	2500	5	32
C12	500	2500	10	46
C13	500	2500	83	258
C14	500	2500	125	323
C15	500	2500	250	556
C16	500	12500	5	11
C17	500	12500	10	18
C18	500	12500	83	113
C19	500	12500	125	146
C20	500	12500	250	267

In the proposed PSO, the size of population is set to 20. c_1, c_2 and c_3 are set to 2.0, and $\omega = 0.5$. If the generation number reaches 1250 or the algorithm is unable to obtain any better MST in 250 continuous generations, the DPSO-STP stops. Each benchmark was carried out for 10 times.

To prove the efficiency of the proposed PSO, several results of heuristic algorithms [1][6], including the SPH, the ADH, the DCH and the GA, are given. The results of SPH and ADH are reported in [1], while the ones of the DCH are in [6]. The parameters of GA are set as follows: crossover probability $px = 0.7$, mutation probability $pm = 0.04$, size of population $size = 50$. The GA terminated when the generations came up to 500 or the algorithm stop finding better results for 100 generations. So there is at most 50*500 = 25000 MST generated in each run of GA, as many as the proposed PSO. Because the section to generate MSTs is the most complex part of the algorithms for STP, we reason to compare two algorithms by the number of generating the MST. B and C problems were also carried out on GA for 10 times.

B. Experimental results

In Table 3, the relative errors of B problems, generated by the SPH, the ADH, the DCH, the GA and the DPSO-STP are presented. Due to only 50-100 nodes in B problems, each algorithm performs well. Both the SPH and the ADH achieved results whose relative errors are less than 5% in most cases. The relative error Err was defined as

$$Err = \frac{result - OPT}{OPT}.$$

The result is the average date of 10 runs. The DPSO-STP and the GA are the best two among these algorithms. The GA obtains the optimal fitness in every run of every case. So does the DPSO-STP.

Table 3. The relative errors of B problems

No.	SPH	ADH	DCH	GA	DPSO-STP
B01	0	0	0	0	0
B02	0	0	3.6	0	0
B03	0	0	0	0	0
B04	5.08	5.08	1.69	0	0
B05	0	0	8.17	0	0
B06	0	0	3.27	0	0
B07	0	0	5.40	0	0
B08	0	0	2.88	0	0
B09	0	0	0.90	0	0
B10	4.65	4.65	17.44	0	0
B11	2.27	2.27	4.54	0	0
B12	0	0	0.57	0	0
B13	7.88	4.24	14.54	0	0
B14	2.55	0.43	2.12	0	0
B15	0	0	3.14	0	0
B16	3.15	0	19.68	0	0
B17	3.82	3.05	6.8	0	0
B18	1.83	0	0.91	0	0

Table 4 shows the number of MSTs generated by the GA and the DPSO-STP before finding out the best results. It is obvious from Table 4 that the DPSO-STP obtains the optimal value much faster than GA. From B01 to B06, which are simplest, GA needed nearly two or three times computation of the DPSO-STP. With the increasing nodes, the gap between the GA and the DPSO-STP became larger and larger. For example, in the benchmark B10, the GA generates average 780 MSTs to find the optimum while the DPSO-STP does 72, which is only 10% of the GA. The situations of B14 and B15 are similar.

TABLE 4 THE NUMBER OF MSTS GENERATED BY GA AND PSO

No.	GA	DPSO-STP	No.	GA	DPSO-STP
B01	105	42	B10	780	72
B02	160	54	B11	585	142
B03	100	52	B12	260	144
B04	120	82	B13	1100	468
B05	130	50	B14	4020	342
B06	515	258	B15	1380	94
B07	275	42	B16	885	110
B08	185	48	B17	925	144
B09	275	56	B18	1250	338

The category C problems involve 500 nodes, and thousands of edges as Table 2 showed. Because the test data on category C is seldom reported, only the DCH and the GA algorithm were adopted for comparison.

TABLE 5 THE BEST, WORSE AND MEAN RESULTS OF CATEGORY C

No.	GA			DPSO-STP		
	Best	Worst	Mean	Best	Worst	Mean
C01	85	85	85	85	85	85
C02	144	144	144	144	144	144
C03	754	761	757.8	754	758	754.4
C04	1081	1091	1083.8	1079	1080	1079.2
C05	1579	1583	1580.6	1579	1579	1579
C06	55	55	55	55	55	55
C07	102	102	102	102	102	102
C08	510	522	514.8	509	512	509.9
C09	712	721	716	708	711	709.1
C10	1093	1103	1097.2	1093	1098	1094.5
C11	32	32	32	32	33	32.1
C12	46	47	46.2	46	46	46
C13	259	263	261.5	259	263	260.5
C14	325	331	327.6	324	326	324.8
C15	559	567	561.8	556	558	556.7
C16	11	12	11.3	11	12	11.4
C17	18	19	18.2	18	19	18.4
C18	117	120	118.4	115	119	116
C19	152	155	153.3	146	149	147.4
C20	268	276	272.4	267	267	267

TABLE 6. THE RELATIVE ERROR THE NUMBER OF GENERATING MST

No.	DCH	GA		DPSO-STP	
	Err	Err	MST	Err	MST
C01	3.5	0	380	0	66
C02	23.6	0	815	0	204
C03	7.16	0.50	15430	0.05	4706
C04	6.02	0.44	19480	0.02	5902
C05	1.25	0.10	17970	0	806
C06	14.06	0	1615	0	3160
C07	18.62	0	3255	0	3774
C08	9.03	1.14	16810	0.18	20502
C09	7.21	1.27	15965	0.30	25020
C10	2.92	0.38	18980	0.14	20648
C11	3.5	0	1410	0.31	12090
C12	23.6	0.43	3535	0	4156
C13	7.16	1.36	14390	0.97	25020
C14	6.02	1.42	12480	0.56	25020
C15	1.25	1.04	17730	0.16	21280
C16	14.06	2.73	2940	3.64	12780
C17	18.62	1.11	2450	2.22	11944
C18	9.03	4.78	10105	2.65	25020
C19	7.21	5	11160	0.96	24826
C20	2.92	2.02	13540	0	4388

The best, worst, mean cost, number of generating MSTs and relative error of the DPSO-STP in category C problems are illustrated in Table 5 and 6. According to the tables, the DPSO-STP performed much better than the GA. All benchmarks except C16 and C17, the DPSO-STP seemed to have more opportunity to get the optimal solution than GA. In 16 of 20 problems (C01 to C08, C10-C12, C15-C17, C19 and C20), the PSO succeeded in finding the best tree at lease once. With the number of nodes increasing, the PSO keeps efficient with relative errors less than 1% in most cases. However, in the C16, C17 and C18, the DPSO-STP showed a little weak. The three benchmarks are relative dense networks with a few objective nodes. As to sparse network that with less edges, the proposed algorithm is indeed effective. From Table 6, it is apparent that GA converges faster than PSO in category C. However, it should be noticed that the PSO achieved better results in most cases, which means that GA may be trapped in the local optimal area and stopped getting better results. So the GA generates much less MSTs according the terminate criterion.

C. Analysis of c_3

The DPSO-STP with and without parameter c_3 are applied to the C08 problem in category C for 20 times to find out how c_3 works. The algorithm without c_3 meant that all blanks were filled with bits from X_i^k in *Modification* 6.

Fig. 4 shows the convergence of the DPSO-STP with and without c_3 in C08 problem. Its vertical axis denotes the fitness of average $gbest^k$ in 20 runs and the horizon axis is the generation. The solid and dash curves represented the DPSO-STP with and without c_3, respectively. According to Fig. 4, both the fitness of the two DPSO-STP decreases rapidly in the first several generations, because the Prim's algorithm [23] and trimming are both greedy methods, which promises to obtain a suboptimal result very soon. However, PSO without c_3 is trapped in 10 generations and hard to find better results, while the algorithm with stayed c_3 performs much better in prophase and keeps going to the optimal point slowly in the later phase.

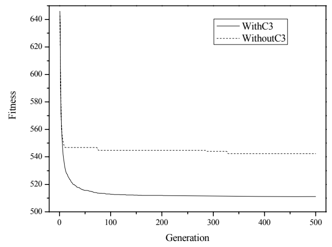


Fig. 4 The convergence of the DPSO-STP with and without c_3 in C08

As a result, the DPSO-STP with c_3 gets averaged fitness of 510.1, which was only a little greater than the optimal 509. And the result of the other DPSO-STP is only 542.3. The variation rate v_r, which is illustrated in Fig. 5, is defined as:

$$v_r = \frac{\sum_{i=1}^{runs}\sum_{j=1}^{size} m_{ij}}{runs \times size}$$

The *runs* and *size* are the number of runs and population, and m_{ij} is the number of different bits of the *j*-th particle in *i*-th run between the last and current generation. As Fig. 5 shows, the DPSO-STP with c_3 always keeps a variation rate at about 0.1, which indicates that several nodes in X_i^k are added or deleted all the time. As to the one without c_3, because all the particles become similar rapidly, few, even none, modifications happens as the dash line shows. It means almost all particles keep their binary string static and the algorithm is trapped. So c_3 is an important parameter for the proposed DPSO-STP.

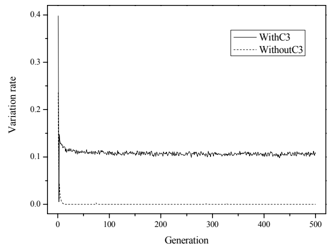


Fig. 5 The variation rate of the DPSO-STP with and without c_3

V. Discussion

As mention before, the Steiner tree problem in graphs is only a special case of multiple destination routing problems. However, the DPSO-STP algorithm can solve the extended MDR problems with several simple modifications. The two sorts of typical problems are discussed as follows.

The first kind is the dynamic multicast routing problem [24]. In this problem, the nodes are added or removed from the dynamic network during the running time. Thus the algorithm should detect the change of network and respond in time. In the DPSO-STP, the binary string should represent all the nodes, including the active and the dumb ones. If some nodes are removed, that is the active ones become dumb, their corresponding bits are forced to be to 0 before generating a MST from the binary string, as the objective nodes are always set to 1. When some nodes join into the network, they are not forced to be 0 any more, and always have the opportunities to be selected because of the mutation.

The other kind is the QoS multicast routing problem [19]. In this problem, the solution tree should meet several constraints, such as the total delay, the bandwidth, and so on. For the bandwidth constraint, another preprocessing operation is needed to delete the edges which can not fulfill the bandwidth requirement. Therefore the algorithm won't generate the illegal solutions. As to the total delay constraint, a validity check should be adopted in steps b and c of the modified Prim's algorithm. That is, if the shortest real (virtual) edge costs too much time to get a valid tree, the second shortest one will replace it. However, it can not insure to generate a valid MST, so the solution should be discarded or repaired.

Though the experimental results of the extensive MDR problems are not given, the discussion shows that the algorithm is feasible for them.

VI. Conclusion

In this paper, a novel discrete particle swarm optimization is proposed for the Steiner tree problem in graphs. It is different from the standard PSO in 4 parts: 1. the preprocessing operations; 2. the representation; 3. the update operation; 4. the improving strategies. With the new parameter c_3, the DPSO-STP has achieved favorable results compared with other algorithms for the STP by the experiments on 38 test cases. Even in the large networks containing 500 nodes, the proposed PSO is still an effective algorithm. The future work is focus on simplifying the algorithm and extending it to other MDR problems.

References

[1] Y. Leung, G. Li and Z. B. Xu: "A genetic algorithm for multiple destination routing problems", *IEEE Trans. Evol. Comput.*, vol. 2, No. 4, Nov. 1998, pp. 150-161

[2] V. J. Rayward-Smith and A. Clare: "On finding Steiner vertices", *Networks*, vol. 16, 1986, pp. 283-294

[3] L. Kou, G. Markowsky and L. Berman: "A fast algorithm for Steiner tree", *Acta Informatica*, vol. 15, 1981, pp. 141–145

[4] F. Bauer and A. Varma: "Distributed algorithms for multicast path set up in Data Networks", *IEEE/ACM Trans. Networking*, vol. 4, No. 2, 1996, pp. 181–191

[5] S. Voβ: "Steiner problem in graphs: Heuristic methods", *Discr. Applied Math*, vol. 40, 1992, pp. 45–72

[6] C. Shampa, B. Arvind and R. Aman, "Directed Convergence Heuristic: A fast & novel approach to Steiner Tree Construction," *Int. Conf. Very Large Scale Integration*, pp. 255 – 260, 2006

[7] K. A. Downsland: "Hill-climbing, simulated annealing and the Steiner problem in graphs", *Eng. Optim*, vol. 17, 1991, pp. 91–107

[8] H. Esbensen: "Computing near-optimal solutions to the Steiner problem in a graph using a genetic algorithm", *Networks*, vol. 26, 1995, pp. 129–167

[9] G. Singh, S. Das, S. Gosavi and S. Pujar: "Ant colony algorithms for Steiner trees: An application to routing in sensor networks", *Recent Development in Biologically Inspired Computing*, Idea Group Publishing, 2005, pp. 181–206

[10] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 1942-1948.

[11] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 69-73.

[12] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. IEEE Int. Congr. Evol. Comput.*, vol. 1, 2001, pp. 101-106.

[13] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 58-73, Feb. 2002.

[14] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp.225-239, Jun. 2004.

[15] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of

multimodal functions," *IEEE Trans. Evol. Comput.*, vol.10, no.3, pp. 281-295. Jun. 2006.

[16] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of IEEE International Conference on System, Man, and Cybernetics*, pp 4104–4109, 1997.

[17] B. Al-kazemi and C. K. Mohan, "Discrete multi-phase particle swarm optimization," *Infomation Processing with Evolutionary Algorithms,* Springer Berlin Heidelberg, pp.306-326, 2006.

[18] G. Pampara, N, Franken, and A. P. Engelbrecht, "Combining particle swarm optimisation with angle modulation to solve binary problems," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 89-96, 2005.

[19] Z. Wang, X. Sun, D. Zhang, "A PSO-Based Multicast Routing Algorithm Natural Computation," *Int. Conf. Naetural Computation*, 2007, pp. 664 – 667

[20] C. Li, C. Cao, Y. Li, Y. Yu, "Hybrid of Genetic Algorithm and Particle Swarm Optimization for Multicast QoS Routing Control and Automation," *Int. Conf. Control Automation*, 2007, pp.2355 – 2359

[21] J. Wang, X. Wang, M. Huang, "An Intelligent QoS Multicast Routing Algorithm under Inaccurate Information", *Int. Conf. Comput. Intelligence and Security*, 2006, pp. 1073-1077

[22] P. Yuan, C. Ji, Y. Zhang, Y. Wang, "Optimal multicast routing in wireless ad hoc sensor networks," *IEEE Int. Conf. Networking, Sensing and Control*, 2004, pp.367-371

[23] H.C. Thomas, E.L. Charles, L. R. Ronald and S. Clifford, *Introduction to algorithms*, MIT press, 1990

[24] H.C. Lin, S.C. Lai, "VTDM-a dynamic multicast routing algorithm," *IEEE Inf. Proc. Comput. Commun. Societies*, pp. 1426-1432