JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 24, 1081-1094 (2008)

DOI:10.1688/JISE.2008.24.4.5

Ant Colony Optimization Approaches to the Degree-constrained Minimum Spanning Tree Problem

YOON-TECK BAU, CHIN-KUAN HO AND HONG-TAT EWE
Faculty of Information Technology
Multimedia University
Cyberjaya, 63100 Malaysia

This paper presents the design of two Ant Colony Optimization (ACO) approaches and their improved variants on the degree-constrained minimum spanning tree (d-MST) problem. The first approach, which we call p-ACO, uses the vertices of the construction graph as solution components, and is motivated by the well-known Prim's algorithm for constructing MST. The second approach, known as k-ACO, uses the graph edges as solution components, and is motivated by Kruskal's algorithm for the MST problem. The proposed approaches are evaluated on two different data sets containing difficult d-MST instances. Empirical results show that k-ACO performs better than p-ACO. We then enhance the k-ACO approach by incorporating the tournament selection, global update and candidate lists strategies. Empirical evaluations of the enhanced k-ACO indicate that on average, it performs better than Prüfer-coded evolutionary algorithm (F-EA), problem search space (PSS), simulated annealing (SA), branch and bound (B&B), Knowles and Corne's evolutionary algorithm (K-EA) and ant-based algorithm (AB) on most problem instances from a well-known class of data set called structured hard graphs. Results also show that it is very competitive with two other evolutionary algorithm based methods, namely weight-coded evolutionary algorithm (W-EA), and edge-set representation evolutionary algorithm (S-EA) on the same class of data set.

Keywords: ant colony optimization, degree-constrained minimum spanning tree, Kruskal, metaheuristic, NP-hard, Prim, swarm intelligence

1. INTRODUCTION

The ant colony optimization (ACO) is a metaheuristic approach for solving hard combinatorial optimization problems. The inspiring source of ACO is the pheromone trail laying and the behaviour of artificial ant following the foraging behaviour of real ants which use pheromones as a communication medium [1]. An ant, while going from the colony to the food source lays a chemical substance, called pheromone. When the ant returns from the food source, it reinforces the pheromones on the path that it had used. Pheromone trail laying is used to attract other ants to follow a particular path. When a large number of ants forage for food, the minimum cost path to the food source will eventually contain the highest concentration of pheromones, thereby attracting all the ants to use that minimum cost path. The pheromones on higher cost path which is not reinforced often enough will progressively evaporate. ACO algorithms are modelled after this behaviour, and have been used to solve minimum cost path problems and problems that can be reduced to a kind of shortest path problems [1].

Received August 9, 2006; revised December 6, 2006; accepted January 23, 2007.
Communicated by Chin-Teng Lin.

This work proposes ACO algorithms to find the degree-constrained minimum spanning tree (d-MST). The problem of finding a d-MST of a graph is a well-studied NP-hard problem and important in the design of telecommunication networks, design of networks for computer communication, design of integrated circuits, energy networks, transportation, logistics, sewage networks and plumbing for maximum network reliability, and optimality such as rerouting of traffic in case of vertex failures, and improve the network performance by distributing the traffic across many vertices [2]. There might be constraints imposed on the design such as the number of vertices in a subtree, degree-constraints on vertices, flow and capacity constraints on any edge or vertex, and type of services available on the edge or vertex.

The d-MST problem is obtained by the modification of the MST problem from a given connected, edge weighted, undirected graph G, such that no vertex of the spanning tree has degree greater than d. It is because when $d = 2$, the MST meeting the constraint will take the form of a path. This is the path of least total weight which includes every vertex in the graph. In other words, this is a Hamiltonian path. Hence an algorithm which solves the d-MST problem also solves the Hamiltonian path problem, which is NP-complete. Therefore, the d-MST is NP-hard problem. The d-MST is in P if $d = |V| - 1$, whereby $|V|$ is the number of vertices. When $d = |V| - 1$ there is no degree constraint and this is equal to MST problem that could be solved using a polynomial amount of computation time.

The d-MST problem was first studied by Deo and Hakimi in 1968 [3]. Since computing a d-MST is NP-hard for every d in the range $2 \leq d \leq |V| - 2$, a few heuristics had been introduced to solve the d-MST problem such as ant colony optimization [4, 5], branch and bound [2], evolutionary algorithms [2, 6-8], genetic algorithms [2, 9-11], Lagrangean relaxation [2, 12], parallel algorithms [13], problem space search [2], simulated annealing [2] and variable neighbourhood search [14]. Recently, Soak, Corne and Ahn [6] have proposed a new encoding based on tree construction rule for d-MST to be used with EAs. The d-MST problem has also been studied for the complete graphs of points in a plane where edge costs are the Euclidean distance between these points coordinate. Euclidean problems are relatively simple to solve [2]. For these Euclidean problems there always exists a MST with degree no more than five and it is also being described in [15]. Using exact algorithms such as branch and bound and Lagrangean relaxation as described by Krishnamorrthy *et al.* [2], one can find optimal solutions even for large problem instances including several hundred vertices in polynomial time. This showed that there exist effective polynomial-time heuristics for finding d-MST in the plane.

In practice, the costs associated with the graph's edge are arbitrary and need not satisfy the triangle inequality [7]. For example, where edge costs are defined to be communications costs between vertices, physical distance can be very minor factor in comparison to others such as the type, capacity, quality of line, maintainability, speed, corporate provider of the link, and *etc*. In this case, a MST may have degree up to $|V| - 1$. Exact approaches and existing heuristics have no guaranteed bounds on the quality of the solutions and it becomes ineffective for graphs with large number of vertices. However, in this paper we present the design of two Ant Colony Optimization (ACO) approaches and their improved variants on the degree-constrained minimum spanning tree (d-MST) problem. The first approach, which we call p-ACO, uses the vertices of the construction graph as solution components, and is motivated by the well-known Prim's algorithm for constructing MST. The second approach, known as k-ACO, uses the graph edges as solution components, and is motivated by Kruskal's algorithm for the MST problem. The proposed approaches are evaluated on two types of non-Euclidian graph problems for d-MST problem. We then enhance the k-ACO approach by incorporating the tournament selection, global update and candidate lists strategies.

2. PROBLEM FORMULATION FOR d-MST

The degree-constrained minimum spanning tree (d-MST) problem can be stated as follows: Let graph $G = (V, E)$ be a connected weighted undirected graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is a finite set of vertices, and $E = \{e_{ij} \mid i \in V, j \in V, i \neq j\}$ is a finite set of edges representing connections between these vertices. Each edge has a nonnegative real number denoted by $W = \{w_1, w_2, \ldots, w_{|E|}\}$, representing weight or cost. Note that in a complete graph having $|V|$ vertices, the number of edges, $|E|$, is $|V|(|V| - 1)/2$, and the number of spanning trees is $|V|^{|V|-2}$. A spanning tree always consists of $|V| - 1$ edges. Any subgraph of G can be described using a vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ where each x_i is a binary decision variable defined as:

$$x_i = \left\{ \begin{matrix} 1, & \text{if edge } e_{ij} \text{ is part of the subgraph} \\ 0, & \text{otherwise} \end{matrix} \right\}. \tag{1}$$

Let S be a subgraph of G. S is said to be a spanning tree in G if S:

(a) contains all the vertices of G and the vertices can be in non-order form;
(b) is connected, and graph contains no cycles.

Now let T be the set of all spanning trees corresponding to the simple graph G. In the MST problem, if we assume that there is a degree constraint on each vertex such that the degree value d_j of vertex j is at most a given constant value d, the number of edges incident to each vertex is constrained. Then the problem is denoted as a d-MST and can be formulated as follows:

$$\min \left\{ z(\mathbf{x}) = \sum_{i=1}^{|E|} w_i x_i \mid j \in V, d_j \leq d, x \in T \right\}. \tag{2}$$

3. RELATED WORK

ACO have been applied to solve constrained MST problems such as the capacitated MST [16] and generalized MST [17] problems. However, for solving the d-MST problem that we are doing, there are several approaches found in the literature. Among the approaches that were used for our comparison study were Prüfer-coded evolutionary algorithm (F-EA) [2], problem search space (PSS) [2], simulated annealing (SA) [2], branch and bound (B&B) [2], Knowles and Corne's evolutionary algorithm (K-EA) [7], weight-coded evolutionary algorithm (W-EA) [9], edge-set representation evolutionary algorithm (S-EA) [8] and ant-based algorithm (AB) [5]. Krishnamoorthy *et al.* [2] presented F-EA, PSS, SA and B&B for the d-MST problem. F-EA employs Prüfer coding based evolutionary algorithm (EA) and uses standard single point crossover. PSS is metaheuristic which combines a simple constructive heuristic with a genetic algorithm. They use Prim's based heuristic to construct randomized minimum spanning tree. The SA approach provides a means to escape local optima by allowing some downhill moves during stochastic hill climbing in hopes of finding a global optimum. As the SA temperature parameter is decreased to zero, uphill moves occur less frequently to minimize the spanning tree cost. In their SA implementation, they set initial temperature at 1% of the standard deviation in edges weight over a random walk of length 20 neighbouring solutions. The temperature is reduced by 2% until either 3 successive chains produce the same result or when 300 temperature decrements have been performed. B&B is in general an exact technique. Since complete runs would have been too time-demanding, each run is terminated after 10 minutes CPU time and the best solution found so far was reported as final solution. Knowles and Corne [7] described another EA for the d-MST problem. In their algorithm, chromosomes are sequences of integer values that influence the order on which edge vertices to connect to form the growing spanning tree. Raidl and Julstrom [9] presented W-EA for the d-MST problem. In this W-EA approach, a feasible spanning tree is represented by a string of numerical weights associated with the vertices. During decoding, these weights temporarily bias the graph's edge costs. Raidl [8] also presented S-EA for the d-MST problem. In this S-EA approach, spanning trees in EA is represented directly as sets of their edges. Special initialization, crossover, and mutation operators are used to generate new, always feasible candidate solutions. Bui T. N. and Zrncic C. M. [5] presented AB for the d-MST problem. In their algorithm they use cumulative pheromone levels to determine candidate sets of edges from which degree-constrained spanning trees are built.

4. THE ACO APPROACHES

This section details two of our proposed ACO approaches and their improved variants on the degree-constrained minimum spanning tree (d-MST) problem. The first approach uses the graph edges as solution components, and is motivated by Kruskal's algorithm [18] for the MST problem, which we refer to as k-ACO's in this paper. The second approach uses the vertices of the construction graph as solution components, and is motivated by the well-known Prim's algorithm [19] for constructing MST, which we refer to as p-ACO in this paper. Then we detail our enhanced k-ACO approach by incorporating the tournament selection strategy, global update strategy and candidate lists strategy.

4.1 The k-ACO

The set of edges, $E = \{e_{ij} \mid i \in V, j \in V, i \neq j\}$ will serve as solution components from which ant will use to incrementally construct a degree-constrained spanning tree (d-ST) during each iteration of the algorithm. Firstly, we sort the edges in E into non-decreasing order by their edge weight. Then we let edge set E_T be a set of edges and let E_T be initially empty. The Kruskal's algorithm starts by selecting the first edge in the sorted edges. This edge will be the first minimum weight edge added to the E_T. At each step, Kruskal's algorithm only allows edges that connect any two different trees in the forest to be added to the current growing tree until E_T contained cardinality of $|V| - 1$ to form a spanning tree. This step implies that Kruskal's algorithm allows a spanning tree to be formed by more than one different tree components. The artificial ants follow this construction step, except that the selection of the next solution component is probabilistic to avoid the pitfalls of the greedy approach. To ensure the degree constraint is adhered to, we do not add next edges that violate the degree constraint.

The objective function f returns the cost of the degree-constrained spanning tree S_k found by ant k. Let w_{ij} be the weight between vertices i and j for edge e_{ij} and the amount of pheromone on the edge that connects vertex i and vertex j for edge e_{ij}. $\tau_{e_{ij}}$ is initially set to small value as $\tau_0 = 10^{-6}$ where τ_0 is the initial pheromone. The algorithm consists of a series of iterations:

(1) A set of *mAnts* artificial ants are initially located at randomly selected edges *mAnts* = $|V|$;
(2) Each ant, denoted by k, constructs a valid d-ST, selecting until $|V| - 1$ edges, always maintaining a set list J_k of feasible edges that remain to be visited. For this case, d-ST is valid if it adhere to its degree constraint. For efficient implementation of feasible edges we test whether two vertices are already connected via some edges or not by using a union-find data structures [20];
(3) At step r of iteration t, an ant will select an edge among the feasible edges that have not yet been visited. This edge will connect two different components of the forest in the graph that does not violate the degree constraint to be included in its partially constructed solution, according to probability:

$$p_{e_{ij}}^{k}(t_r) = \begin{Bmatrix} \dfrac{[\tau_{e_{ij}}(t)]^{\alpha}[\eta_{e_{ij}}]^{\beta}}{\sum_{l \in J_k}[\tau_{e_l}(t)]^{\alpha}[\eta_{e_l}]^{\beta}}, & \forall l \in J_k(e_{ij}),\ \eta_{e_{ij}} = \dfrac{1}{w_{ij}} \\ 0, & \text{otherwise} \end{Bmatrix} \tag{3}$$

where

$$w_{ij} = \begin{Bmatrix} w_{ij}, & \text{if } w_{ij} > 0 \\ \tau_0, & \text{if } w_{ij} = 0 \end{Bmatrix} \tag{4}$$

where α and β are two positive parameters which govern the respective influences of pheromone and distance visibility on ants' decision and η as inverse distance visibility measure.

(4) When every ant has completed a d-ST, pheromone trails are updated:

$$\tau_{e_{ij}}(t+1) = (1-\rho)\ \tau_{e_{ij}}(t) + \sum_{k=1}^{mAnts} \Delta\tau_{e_{ij}}^{k} \tag{5}$$

where

$$\Delta\tau_{e_{ij}}^{k} = \rho\tau_0, \text{ as local update} \tag{6}$$

$$\Delta\tau_{e_{ij}}^{k} = \begin{cases} Q/L^{gb}, & \text{if } e_{ij} \in d\text{-}ST^{gb} \\ 0, & \text{otherwise} \end{cases} \text{ as global update} \tag{7}$$

where ρ is the evaporation rate $\tau_0 = 10^{-6}$ is the initial pheromone, L^{gb} is the weight of global-best degree-constrained spanning tree $d\text{-}ST^{gb}$ and Q is a positive integer.

Pheromone evaporates at a fixed rate after all ants have constructed their d-ST. $\tau_{e_{ij}}$ is the amount of reinforcement received by edge e_{ij}. $\Delta\tau_{e_{ij}}$ is proportional to the quality of the solutions in which edge e_{ij} was used for construction. While ant k is building a solution a local update rule is applied [21]. The local update rule was needed to yield better performance by encouraging ants' exploration. Only the globally best degree-constrained spanning tree L^{gb} by ant k is allowed to deposit additional amount of pheromone. This is used as the elitist strategy [22]. The ACO approaches proposed in this paper follows that of the Ant Colony System in [21], and incorporates the elitist strategy from Ant System [22].

4.2 The p-ACO

The p-ACO uses the vertices of the construction graph as solution components. During each iteration of the p-ACO, every ant will use these solution components to construct a degree-constrained spanning tree (d-ST) incrementally. Initially we let vertex set *CS* be an unordered list of vertices and let *CS* be initially empty. The Prim's algorithm starts from an arbitrary root vertex in graph. This arbitrary root vertex will be the first vertex added to the *CS*. At each step, Prim's algorithm select edge e_{ij} whereby the vertex i contains in the *CS* and the second unconnected vertex j does not contains in the *CS* to be added to the current growing tree. *CS* is grown until it contains all the vertices in V to form a spanning tree. This step implies that Prim's algorithm always form a spanning tree by a single tree component. The artificial ants follow this construction step, except that the selection of the next solution component is probabilistic to avoid the pitfalls of the greedy approach. To ensure the degree constraint is adhered to, we do not add next vertices that violate the degree constraint. The p-ACO description had been given in our previous paper [4].

4.3 The Enhanced k-ACO

We enhanced k-ACO by incorporating three strategies. Strategy 1 is the tournament selection strategy. Strategy 2 is the global update strategy. And lastly, strategy 3 is the candidate lists strategy.

4.3.1 Strategy 1: tournament selection

An ant selects a set of other feasible edges using Eq. (3) with tournament selection of size 2 [23] instead of using typical roulette wheel selection [24, 25] method. Fig. 1 shows our enhanced k-ACO incorporated with tournament selection of size 2 pseudocode. Tournament selection can be likened to the natural process of individuals

```
Algorithm    Tournament selection of size 2
Input: Feasible edges, $E^*$ of ant $k$ from Eq. (3) into the tournament
Output: Only one feasible $e_{ij}$ as a winner
begin
Set max_probability ← 0
for count ← 1 to $|E^*|/2$ do
    1. Randomly select two edges from $E^*$ to compete
    2. Compare the probability of the two edges with max-probability: the edge with higher
       probability compare to max_probability will be the winner and then update max_prob-
       ability as winner probability
return winner $e_{ij}$
end
```

Fig. 1. Pseudocode for tournament selection of size 2.

competing with each other in order to become winner. Selection pressure can be easily adjusted by changing the tournament size.

4.3.2 Strategy 2: using global pheromone trail update rule

We adapt the global pheromone trail update rule $\Delta\tau^k_{e_{ij}} = Q/L^{gb}$ by defining L^{gb} as follows:

$$L^{gb}(e_{ij}) = L^{gb} + \frac{\text{degree}(i) + \text{degree}(j)}{2 \times d} L^{gb}. \tag{8}$$

This strategy introduces degree constraint knowledge into global pheromone trail update instead of solely using the total weight of global-best degree-constrained spanning tree, L^{gb}. The Eq. (8) decreasing some amount of pheromone for higher degree constraint edge e_{ij} vertices.

4.3.3 Strategy 3: using candidate lists

For each ant, a list containing the N least expensive edge yet to be included in the partial solution. Our candidate lists comprise a top N size of the sorted feasible edges, E^* set by its weight. The reason we apply the candidate lists strategy is to reduce large neighbourhood in our solution construction caused by the large number of edges in the graph. This allows our enhanced k-ACO algorithm to focus on more promising edges of the current state. We apply candidate lists of size 30 in our experiment. This value is obtained from the parameter tuning process.

5. COMPUTATIONAL SETUP

This section first describes two different data sets containing difficult d-MST instances: structured hard (SHRD) graph and misleading graph (M-graph). After that, we describe how the parameters for both p-ACO and k-ACO are tuned for better solution quality.

5.1 Data Sets

The structured hard (SHRD) graphs are constructed by using non-Euclidean distance. These are difficult to solve optimally compared to other data sets such as Euclidean data sets of degree 3 or more [2]. The MST for SHRD is a star graph where one vertex has degree $|V| - 1$ and the all other vertices have degree 1. The misleading graph (M-graph) data set consists of randomly generated weights with a positive sign, greater than or equal to 0.0 and less than 1.0 attached to the edges in such a way that they will mislead greedy algorithms [7]. The M-graph is based on constructing the unconstrained MST to contain star patterns of degree d. Both data sets are complete graphs with undirected non-negative weighted edges.

5.2 Parameter Settings

The range for β was 2, 4, 6, 8, and 10, and ρ was 0.01, 0.02, 0.05, 0.10, and 0.20. For each β and ρ combinations used for p-ACO and k-ACO parameter settings, we record average solution costs over 20 independent runs. Each run is terminated after 30 iterations. The lowest average solution costs for each β and ρ combinations will be our p-ACO and k-ACO algorithms parameter values used in SHRD and M-graph data sets. This is because the lower the average solution costs indicate that the solution quality is higher. Tables 1 to 4 show parameter tuning results for p-ACO and k-ACO. The lowest value from the 5 × 5 parameter setting table is in bold print. It happened that for SHRD problem instance the k-ACO uses the same parameter values of $\beta = 10$ and $\rho = 0.01$ as p-ACO. Separate parameter values are used for p-ACO and k-ACO for the M-graph problem instances. The parameter values of $\beta = 4$ and $\rho = 0.02$ are chosen for p-ACO while values of $\beta = 2$ and $\rho = 0.10$ are chosen for k-ACO. Table 5 shows our final parameter setting for artificial ant k used for both p-ACO and k-ACO in our experiment on

Table 1. Parameter tuning for p-ACO average results, problem shrd305, iterations = 30, runs = 20.

	$\rho = 0.01$	0.02	0.05	0.10	0.20
$\beta = 2$	1590.50	1585.80	1599.65	1625.50	1680.60
4	1529.45	1533.00	1533.85	1540.00	1572.00
6	1518.95	1517.30	1521.90	1524.55	1545.15
8	1515.25	1515.65	1515.75	1518.25	1533.80
10	**1513.55**	1514.15	1514.95	1515.25	1526.75

Table 2. Parameter tuning for k-ACO average results, problem shrd305, iterations = 30, runs = 20.

	$\rho = 0.01$	0.02	0.05	0.10	0.20
$\beta = 2$	1609.45	1608.75	1618.35	1635.50	1700.70
4	1538.80	1541.35	1546.25	1550.85	1579.40
6	1526.10	1523.95	1525.90	1533.00	1548.90
8	1518.00	1518.40	1521.45	1524.55	1538.25
10	**1514.95**	1515.95	1517.40	1521.75	1531.00

Table 3. Parameter tuning for p-ACO average results, problem m50n1, iterations = 30, runs = 20.

	$\rho = 0.01$	0.02	0.05	0.10	0.20
$\beta = 2$	7.8744	7.9141	8.0041	9.0343	10.2777
4	8.0163	**7.7700**	7.9464	8.6402	9.7540
6	8.4264	8.3399	8.3438	8.8363	9.7434
8	9.0393	8.9501	9.0304	9.2558	9.6575
10	9.4282	9.4468	9.3318	9.3614	9.6826

Table 4. Parameter tuning for k-ACO average results, problem m50n1, iterations = 30, runs = 20.

	$\rho = 0.01$	0.02	0.05	0.10	0.20
$\beta = 2$	8.2915	7.9525	7.6415	**7.4913**	9.4632
4	7.9661	7.6310	7.5096	7.8329	8.5959
6	8.7173	8.2863	8.2706	8.2072	8.5624
8	8.9102	8.4881	8.6286	8.5700	8.8353
10	9.0634	9.0570	8.9409	8.7352	8.9876

Table 5. Final parameter setting artificial ant for p-ACO and k-ACO on SHRD and M-graph problem instances.

	p-ACO		k-ACO	
	SHRD	M-graph	SHRD	M-graph
mAnts	$\|V\|$	50	$\|V\|$	50
Q	1.0	1.0	1.0	1.0
α	1	1	1	1
τ_0	10^{-6}	10^{-6}	10^{-6}	10^{-6}
β	10	4	10	2
ρ	0.01	0.02	0.01	0.10

SHRD and M-graph problem instances. The maximum number of ants, *mAnts* is restricted to 50 because the cost of constructing solutions in our algorithms is quite high. The best parameter value of α is set to 1; it is consistent with other ACO research works [22, 26]. Parameter values of β and ρ used for enhanced k-ACO was the same with k-ACO.

5.3 Experiment Setup

The p-ACO, k-ACO and the enhanced k-ACO's approaches are implemented in Java v5.0 using the RePast v3.1 multi-agent simulation framework. For p-ACO, k-ACO and the enhanced k-ACO approaches, 50 independent runs were performed for each problem instance. Each run is terminated after 100 iterations. We have tried up to 300 iterations but the improvement found is too small to justify the additional computation time. All the results that we present in this paper were obtained on PCs with Pentium 4 3000 Mhz CPU running under Windows XP Professional.

Table 6. Average results (quality gains over d-Prim in %) on structured hard problem instances. Label SHRD153 means structured hard graph 15-vertex with degree constraint, *d* = 3 and so on.

Problem	F-EA avg.	PSS avg.	SA avg.	B&B avg.	W-EA avg.	S-EA avg.	AB avg.	p-ACO avg.	p-ACO best	k-ACO avg.	k-ACO best	k-ts-gu-cl-ACO avg.	k-ts-gu-cl-ACO best
SHRD153	13.66	16.62	14.93	18.03	14.20	18.03	20.11	16.28	19.84	16.57	16.60	**20.26**	21.19
SHRD154	10.83	12.99	11.61	14.76	11.42	15.35	14.96	9.30	12.01	11.13	12.01	12.29	15.35
SHRD155	4.00	9.60	9.07	9.60	3.53	9.60	9.60	1.33	1.33	7.58	9.60	8.95	9.60
SHRD203	11.32	10.91	10.43	10.91	12.29	12.43	11.07	11.45	11.87	11.84	12.12	11.72	12.12
SHRD204	6.82	7.05	5.57	7.05	8.50	8.78	9.48	9.20	9.26	9.23	9.48	9.22	9.48
SHRD205	6.28	7.30	7.74	7.30	7.96	8.44	7.74	7.99	8.47	8.10	8.32	8.17	8.47
SHRD253	13.07	15.40	14.73	15.40	16.51	16.75	19.08	19.69	20.40	18.76	20.31	**19.81**	20.40
SHRD254	4.84	6.79	5.56	6.79	6.83	7.69	5.99	6.41	6.72	6.15	6.72	6.41	6.72
SHRD255	5.37	6.74	5.19	8.29	9.01	9.01	7.92	9.00	9.02	8.97	9.02	8.91	9.02
SHRD303	6.51	11.27	9.53	11.27	12.50	12.17	11.55	11.49	12.26	11.37	11.92	12.30	12.46
SHRD304	7.30	10.58	8.45	10.58	11.76	10.80	11.43	11.41	11.71	11.38	11.75	11.61	11.80
SHRD305	2.18	4.74	2.50	4.74	5.77	4.79	5.77	6.33	6.52	6.31	6.58	**6.37**	6.58
Total Average:	7.68	10.00	8.78	10.39	10.02	11.15	11.22	9.99	10.78	10.62	11.20	11.34	11.93

Table 7. Average results (quality gains over d-Prim in %) on misleading problem instances. Label m50nl means first misleading graph 50-vertex with degree constraint, *d* = 5 and so on.

Problem	K-EA avg.	W-EA avg.	S-EA avg.	p-ACO avg.	p-ACO best	k-ACO avg.	k-ACO best	k-ts-gu-cl-ACO avg.	k-ts-gu-cl-ACO best
m50n1	27.59	42.76	43.59	39.87	43.33	41.98	43.31	42.85	43.36
m50n2	33.22	48.63	50.59	45.44	45.51	46.08	50.10	46.70	50.53
m50n3	26.98	29.25	33.33	25.50	32.31	28.23	32.73	31.13	33.43
m100n1	28.89	39.67	42.21	27.65	32.64	-1.23	4.76	24.66	28.56
m100n2	31.25	47.22	49.50	27.28	33.25	5.41	9.69	27.92	33.08
m100n3	26.51	46.04	49.02	22.76	28.32	-0.28	5.69	20.54	24.08
Total Average:	29.07	42.26	44.71	31.42	35.89	20.03	24.38	32.30	35.51

6. EMPIRICAL RESULTS

The solution quality is measured by the relative difference between the final objective value C obtained by a specific approach and the objective value $C_{d\text{-}Prim}$ of the solution found by the d-Prim heuristic where the first vertex is used as starting point in percent. This measure is called quality gain:

$$\text{quality gain} = (C_{d\text{-}Prim} - C)/C_{d\text{-}Prim} \cdot 100\%. \quad (9)$$

In other words, we use d-Prim as a reference algorithm and calculate relative quality improvements for the other approaches; where larger values indicate better results.

6.1 Performance Comparisons on Structure Hard Graph (SHRD) Data Set

Table 6 shows results for the SHRD data set. The numbers of vertices are in the range 15, 20, 25, and 30 where the maximum degree was set to 3, 4 and 5. The results for F-EA, PSS, SA, B&B, W-EA, S-EA, AB and p-ACO are adopted from [4, 5, 8] and printed for comparison purposes only. Besides average gains, the gains of the best run are reported in Table 6. The enhanced k-ACO, incorporating the tournament selection of size 2, global update strategy and candidate lists of size 30 are referred to as k-ts-gu-cl-ACO.

We can conclude that k-ts-gu-cl-ACO has higher total average results compared to k-ACO. In turn, k-ACO has higher total average results compared to p-ACO. Overall our k-ts-gu-cl-ACO approaches attain the highest total average results compared to all other approaches such as F-EA, PSS, SA, B&B, W-EA, S-EA and AB. The k-ts-gu-cl-ACO gave the highest average gains for three problem instances, namely SHRD153 d = 3, SHRD253 d = 3 and SHRD305 d = 5 instances, where the improvement is better than all other non-ACO approaches. The figures of this are in bold print and we can refer it from the Table 6. For other problem instances, the k-ts-gu-cl-ACO is at least performed better than one of the other compared approaches. The better results produced by k-ts-gu-cl-ACO demonstrates the effectiveness of the enhancement strategies.

6.2 Performance Comparisons on Structure Hard Graph (SHRD) Data set

Table 7 shows results for misleading problem instances from Knowles and Corne [7]. The numbers of vertices are 50 and 100. The maximum degree was set to 5. The results for K-EA, W-EA, S-EA and p-ACO are adopted from [4, 8] and printed for comparison purposes only. Besides average gains, the gains of the best runs are reported in Table 7.

From the experiments, we can conclude that the k-ts-gu-cl-ACO has higher total average results compared to p-ACO and k-ACO. Among the ACO approaches, k-ts-gu-cl-ACO used here gave the best average results for graph of 50 vertices. On average, k-ts-gu-cl-ACO performed better than W-EA and K-EA for graph of 50 vertices. However, when the graph size is increased to 100 vertices, k-ACO without any enhancement performed very badly. One of the probable reasons is the increase of the number of edges in the graph (quadratic growth rate). Notice the drastic improvement, when we incorporated the three enhancement strategies for k-ACO. The three problem instances with 100 vertices namely, m100n1 average result increased from − 1.23 to 24.66, m100n2 average result increased from 5.41 to 27.92 and m100n3 average result increased from − 0.28 to 20.54.

7. CONCLUSIONS

We have presented the design of k-ACO, an ACO algorithm for the degree-constrained minimum spanning tree problem. Performance studies have revealed that k-ACO is competitive with a number of other metaheuristic approaches. The incorporation of enhancement strategies such as tournament selection, utilisation of candidate lists and the deployment of a global pheromone update strategy have improved the performance of k-ACO to obtain the best performance for three instances on the SHRD data sets. Despite not being able to produce the best results for the misleading graph data set, the use of enhancement strategies enabled k-ACO to achieve results very close to S-EA, and better than the other approaches for most of the problem instances under the SHRD class. We believe that the proposed ACO approaches extend naturally to the capacitated MST problem.

ACKNOWLEDGEMENTS

We thank A. T. Ernst, G. R. Raidl, and J. Knowles for providing us their test data sets.

REFERENCES

1. M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: algorithms, applications, and advances," F. Glover and G.A. Kochenberger, eds., *Handbook of Metaheuristics*, Kluwer Academic Publishers, 2003, pp. 250-285.
2. M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha, "Comparison of algorithms for the degree constrained minimum spanning tree," *Journal of Heuristics*, Vol. 7, 2001, pp. 587-611.
3. N. Deo and S. L. Hakimi, "The shortest generalized Hamiltonian tree," in *Proceeding of the 6th Annual Allerton Conference*, 1968, pp. 879-888.
4. Y. T. Bau, C. K. Ho, and H. T. Ewe, "An ant colony optimization approach to the degree-constrained minimum spanning tree problem," *Lecture Notes Artificial Intelligence*, Vol. 3801, 2005, pp. 657-662.
5. T. N. Bui and C. M. Zrncic, "An ant-based algorithm for finding degree-constrained minimum spanning tree," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 11-18.
6. S. M. Soak, D. Corne, and B. H. Ahn, "A new encoding for the degree constrained minimum spanning tree problem," *Lecture Notes Artificial Intelligence*, Vol. 3213, 2004, pp. 952-958.
7. J. Knowles and D. Corne, "A new evolutionary approach to the degree-constrained minimum spanning tree problem," *IEEE Transactions on Evolutionary Computation*, Vol. 4, 2000, pp. 125-134.
8. G. R. Raidl, "An efficient evolutionary algorithm for the degree-constrained mini-

mum spanning tree problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, 2000, pp. 104-111.

9. G. R. Raidl and B. A. Julstrom, "A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem," in *Proceedings of the ACM Symposium on Applied Computing*, 2000, pp. 440-445.
10. G. Zhou, M. Gen, and T. Wu, "A new approach to the degree-constrained minimum spanning tree problem using genetic algorithm," in *Proceedings of the International Conference on System*, *Man*, *and Cybernetics*, Vol. 4, 1996, pp. 2683- 2688.
11. G. Zhou and M. Gen, "Approach to the degree-constrained minimum spanning tree problem using genetic algorithm," *Engineering Design and Automation*, Vol. 3, 1997, pp. 228-231.
12. A. Volgenant, "A lagrangean approach to the degree-constrained minimum spanning tree problem," *European Journal of Operational Research*, Vol. 39, 1989, pp. 325-331.
13. L. J. Mao, N. Deo, and S. D. Lang, "A parallel algorithm for the degree-constrained minimum spanning tree problem using nearest-neighbor chains," in *Proceedings of the 4th International Symposium on Parallel Architectures*, *Algorithms*, *and Networks*, 1999, pp. 184-189.
14. C. C. Ribeiro and M. C. Souza, "Variable neighborhood search for the degree-constrained minimum spanning tree problem," *Discrete Applied Mathematics*, Vol. 118, 2002, pp. 43-54.
15. C. Monma and S. Suri, "Transitions in geometric minimum spanning trees," *Discrete and Computational Geometry*, Vol. 8, 1992, pp. 265-293.
16. M. Reimann and M. Laumanns, "Savings based ant colony optimization for capacitated minimum spanning tree problem," *Computers & Operations Research*, Vol. 33, 2006, pp. 1794-1822.
17. S. J. Shyu, P. Y. Yin, B. M. T. Lin, and M. Haouari, "Ant-tree: an ant colony optimization approach to the generalized minimum spanning tree problem," *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 15, 2003, pp. 103-112.
18. J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," in *Proceedings of the American Mathematics Society*, Vol. 7, 1956, pp. 48-50.
19. R. Prim, "Shortest connection networks and some generalizations," *Bell System Technical Journal*, Vol. 36, 1957, pp. 1389-1401.
20. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* 2nd ed., The MIT Press, 2001.
21. M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, Vol. 1, 1997, pp. 53-66.
22. M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems*, *Man*, *and Cybernetics – Part B*, Vol. 26, 1996, pp. 29-41.
23. M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1999.
24. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd revised and extended ed., Springer Verlag, 1996.

25. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
26. T. Stützle and H. H. Hoos, "MAX-MIN ant system," *Journal of Future Generation Computer Systems*, Vol. 16, 2000, pp. 889-914.

Yoon-Teck Bau received the B.IT (First Class Hons.) degree in Information Systems Engineering from Multimedia University, Selangor, Malaysia, in 2002. He is a Tutor of Faculty of Information Technology, Multimedia University, Cyberjaya, Malaysia, since 2002. His research interests include ant colony optimization, artificial intelligence, Java technology and expert systems.

Chin-Kuan Ho received his B.Sc. (First Class Hons.) in Computer Science with Electronics Engineering from University College London in 1999, and M.Sc.(IT) by research from Multimedia University, Malaysia in 2002. He is currently a Lecturer with the Faculty of Information Technology, Multimedia University, and is pursuing his doctorate degree in the area of metaheuristics. His research work and interests focus on computational intelligence, in particular evolutionary algorithms, simulated annealing, and swarm intelligence. He has also been frequently invited as technical consultant for various software development houses.

Hong-Tat Ewe received his Bachelor of Engineering (Hons.) degree from University of Malaya, Malaysia, Masters of Science degree in Electrical Engineering and Computer Science from M.I.T., USA and Ph.D. degree from Multimedia University, Malaysia. He had worked before in Motorola Penang, Intel Penang, MIT's Research Laboratory for Electronics, and University of Malaya, and is currently an Associate Professor at the Faculty of Information Technology, Multimedia University. His research interest includes microwave remote sensing, satellite image processing, and wireless sensor network.