Journal of Marine Science and Technology

Volume 22 | Issue 3 Article 8

A PARTICLE SWARM OPTIMIZATION-LIKE ALGORITHM FOR CONSTRAINED MINIMAL SPANNING TREE PROBLEMS

Chun-Chao Yeh
Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C,
ccyeh@mail.ntou.edu.tw

Ying-Che Chien
Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C

Journal of Marine Science and Technology, Vol. 22, No. 3, pp. 341-351 (2014)
DOI: 10.6119/JMST-013-1119-2

A PARTICLE SWARM OPTIMIZATION-LIKE ALGORITHM FOR CONSTRAINED MINIMAL SPANNING TREE PROBLEMS

Chun-Chao Yeh and Ying-Che Chien

Key words: evolutionary algorithm, ant colony optimization, genetic algorithm, constrained minimal spanning tree.

ABSTRACT

Previous studies have discussed various constrained minimal spanning tree (MST) problems. In this paper, we propose an efficient algorithm for solving a class of constrained MST problems. The proposed PSO (Particle Swarm Optimization)-like strategy for solving constrained MST problems identifies optimal MSTs under degree and delay constraints. The solution quality and computation time of the proposed PLCMST (PSO-Like algorithm for Constrained MST problems) algorithm is compared with two other algorithms: one based on ant colony optimization, and the other based on a genetic algorithm strategy. Our experimental results show that the PLCMST outperforms the other two approaches, particularly when using dense graphs.

I. INTRODUCTION

Network optimization problems have been widely studied in various research fields, such as telecommunications and transportation sciences. Optimal routing problems, such as involved in constructing a minimal-cost broadcast tree covering all subscription nodes in a network, are typical optimization problems encountered in communications networks [10, 24, 43]. Depending on the type of service, various performance metrics and/or technical constraints must be considered simultaneously to identify a solution. For example, hop count, bandwidth, reliability, and traffic loading are widely used performance metrics in Internet routing problems. In communications systems, technical constraints, such as link capacity and transmission rate, are intrinsic because of the limitations of system components. In addition, various quality of service (QoS) requirements must be considered to accommodate specific applications or services. Typical QoS considerations associated with Internet multimedia traffic include minimal bandwidth, maximal delay, and maximal jitter guarantee. Moreover, QoS requirements define service level agreements between customers and service providers; accordingly, additional constraints should be imposed on network optimization problems as necessary.

Among the various types of network optimization problems, minimal spanning tree (MST) problems are one of the most critical models. Unconstrained MST problems can be solved effectively by using existing methods, such as the algorithms proposed by Prim [29] or Kruskal [19]. However, because complex systems may require some constraints to be imposed on the spanning trees; consequently, the problem typically become intractable. This paper focuses on a set of constrained MST problems, in which node-degree and path-delay constraints require consideration. Typical applications of such constrained MST problems include network multicast problems [24, 25, 41] and topology control in wireless communications networks [14, 22]. Similar situations can occur in logistics and transportation problems, when n objects must be transported from one location to m cities under the constraints of maximal path delay and the maximal number of route-link branches between any two locations.

MST problems with node-degree or path-delay constraints have been widely researched; however, few studies have considered these two constraints simultaneously. Tseng *et al*. proposed a genetic algorithm (GA) [39] and ant colony optimization (ACO) strategy [40] for solving constrained MST problems. Chen *et al*. [8] proposed a GA to solve a degree-and-delay-constrained (DDC) Steiner tree problem, which is a general MST problem. As a solution for constrained MST problems, this study proposes a novel evolutionary algorithm (EA) based on a particle swarm optimization (PSO) strategy. We compare the solution quality and computation time of the proposed method with those of two other algorithms based on GA [39] and ACO [40] strategies. The results show that our proposed method outperforms the other two algorithms in solving almost all problems in this study.

Previous studies have used soft-computing approaches (e.g.,

Paper submitted 08/30/12; revised 05/28/13; accepted 11/19/13. Author for correspondence: Chun-Chao Yeh (e-mail: ccyeh@mail.ntou.edu.tw).
Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung, Taiwan, R.O.C.

PSO, GAs, and ACO) to solve many complex problems, particularly nondeterministic polynomial-time (NP)-hard problems. Kuo *et al*. [20] showed that PSO algorithms can be used to solve global optimization problems efficiently. Jovanovic and Tuba [16] improved an existing ACO algorithm to solve a minimal connected dominating set problem. This type of problem is modeled as a solution for identifying a minimal set of relay nodes in a mobile ad hoc network. In [16], a pheromone correction strategy was incorporated into an ACO algorithm to prevent the algorithm from becoming trapped in local optima. Recently, a survey was conducted on applying ACO algorithms in telecommunications networks [5].

The remainder of this paper is organized as follows. Section II provides a detailed review of constrained MST problems, as well as other related problems. In Section III, the problems discussed in this paper are described and modelled. Section IV presents the proposed PSO-like algorithm (PLCMST) for solving constrained MST problems. The simulation results are summarized in Section V, and a conclusion is given in Section VI.

II. CONSTRAINED MINIMAL SPANNING TREE

This section provides a review of constrained MST problems and other relevant optimization problems. Although MST problems can be solved efficiently by using existing algorithms, the original setting of an MST problem may be too primitive for complex application scenarios; consequently, it may be necessary to constrain the problem. Previous studies have investigated various types of constrained optimization problems based on trees or paths [27, 28, 37]. The following subsections briefly review some of those problem models. This study focuses on models that are suitable for optimizing either spanning trees or Steiner trees. A Steiner tree connects a set of designated nodes, but not necessary all nodes in a network (as is the case with spanning trees). In a Steiner tree, the nodes designated for coverage are called terminal nodes; the nodes covered by the Steiner tree that are not terminal nodes are called Steiner nodes. Identifying a minimal-cost Steiner tree is a complex task that is considered an NP-hard problem [12].

1. Degree-Constrained Minimal-Cost Trees

Given a weighted undirected graph $G = (V, E)$ and positive integer K, a degree-constrained MST is a spanning tree in which each node degree is equal to or less than K. A degree-constrained MST problem is NP-hard [12], implying that a low probability exists for developing an efficient solution. To solve this type of problem, various heuristic algorithms have been proposed, such as those based on ACO [3, 7], GAs [36], and PSO [6]. Krishnamoorth *et al*. [18] compared three heuristic schemes for solving the problem; simulated annealing, GA, and a problem space search (PSS)-based method. They showed that the PSS method outperformed the other two schemes marginally. Behle *et al*. [4] employed branch-and cut algorithms to solve a degree-constrained MST problem. Narula *et al*. [26] proposed a branch-and-bound algorithm as well as a primal and dual heuristic procedure to solve this type of problem.

To solve degree-constrained minimal-cost Steiner tree problems, Ravi *et al*. [31] proposed an approximation algorithm for solving degree-constrained minimal-cost Steiner tree problems. The approximation algorithm for constructing a Steiner tree in which the degree of any node v in the output Steiner tree is $O(d(v) \log k)$, and the maximal cost of the tree is $(\log k)$ multiplied by the output of a minimal-cost Steiner tree obeying the degree bound $d(v)$ for each node v, where k is a constant related to a property of the input graph.

The minimal-degree-constrained MST (MDMST) is a similar problem model. An MDMST problem requires a minimal-cost spanning tree, such that each node is either a leaf, or it has a degree of at least d. Here, the input parameter d denotes the minimal node degree of nonleaf nodes in the tree. According to [2], the MDMST problem is NP-hard for cases where $d \geq 3$. Martinez *et al*. [23] proposed a parallel Lagrangian relaxation algorithm to solve a degree-constrained MST problem.

2. Delay-Constrained Minimal-Cost Trees

Given a weighted undirected graph $G = (V, E)$ and positive integer d, a delay-constrained MST is a spanning tree in which the path delay from root node s to each node v is equal to or less than d (the path delay is defined as the sum of all edge delays along the path from s to v). Salama *et al*. [33] proved that the delay-constrained MST problem is NP-hard by reducing it to a known NP-hard problem (i.e., the exact cover by 3-sets problem).

A general model of this type of problem is based on a Steiner tree, in which a set of designated nodes (not necessary all nodes in the tree) should be connected. A delay-constrained minimal-cost Steiner tree problem is a critical model for multicast routing applications in communications networks where communication delay requires consideration. Various approaches have been proposed to solve this type of problem, including the extended Prim-Dijkstra tradeoff [1], greedy randomized adaptive search procedure [35], branch-and-cut algorithm [21], PSO strategy [30], and hybrid scatter search method [42].

The bandwidth-delay-constrained least-cost Steiner tree (BDC-LST) is a similar problem model, in which a least-cost Steiner tree connecting all designated nodes in an input graph is constructed under the constraints of path delay and path bandwidth. Similar to the definition of path delay, for each designated node v in the Steiner tree, the path bandwidth of the path from root node s to node v is defined as the minimal edge-bandwidth of all edges along the path from s to v. The BDC-LST bandwidth constraint requires the path delay of all designated nodes in the BDC-LST tree to be equal to or more than the bandwidth limit b. The BDC-LST problem model is typically used in multicast routing applications, particularly in communications networks where bandwidth and delay are

sensitive to communication quality (e.g., delivering a multicast stream from a source node to a set of subscriber nodes over interconnected networks). Ghaboosi and Haghighat [13] proposed a Tabu search scheme for solving delay-constrained MST problem.

3. Other Constrained Minimal-Cost Trees

In addition to degree and delay, various other constraints have been imposed on minimal-cost tree constructions, some of which are detailed as follows.

A. Bounded-diameter minimal spanning tree:

Given a weighted undirected graph $G = V, E$) and positive value D, a bounded-diameter MST (BD-MST) is a spanning tree with a diameter equal to or less than D. Here, the diameter $dia(G)$ of G denotes the longest path between any two nodes in G. The BD-MST problem is NP-hard for cases where $4 \le D \le (n-1)$ [12]. To solve a BD-MST problem, Gruber *et al*. [15] proposed a neighborhood search scheme to improve the local optima in variable neighborhood search (VNS), EAs, and ACO algorithms. Compared with the VNS and ACO approaches, an EA with an arc exchange neighborhood search procedure for local optimization yielded remarkably superior simulation results. To solve a BD-MST problem, Torkestani [38] proposed a decentralized learning automata-based algorithm that rewards trees with the smallest known weight (otherwise, the trees are penalized).

B. Leaf-constrained minimal spanning tree:

Given a weighted undirected graph $G = (V, E)$ and positive value L, a leaf-constrained MST (LC-MST) is a spanning tree with a minimum of L leaves. Deo and Micikevicius [11] showed that this type of problem is NP-hard. To solve the LC-MST problem, Julstrom [17] proposed two GA-based algorithms that were based on two chromosome coding schemes (blob and subset). The results of that study showed that the subset coding scheme-based algorithm consistently outperformed greedy heuristics-based algorithms. Singh [34] proposed an artificial bee colony algorithm that solved the LC-MST problem efficiently.

C. Capacitated minimal spanning tree:

Given a weighted undirected graph $G = (V, E)$ and positive value K, each node v_i (except the root node $s = v_0$) is associ ated with a demand d_i. A capacitated MST (Cap-MST) is a spanning tree where the total node demand of each subtree originating from root node s is equal to or less than K. The Cap-MST problem can be reduced to the partition problem, which is NP-hard [12]; accordingly, the Cap-MST problem is also NP-hard. Reimann and Laumanns [32] proposed a hybrid ACO-based algorithm for solving Cap-MST problems.

III. PROBLEM MODELS

1. General Description

In [39], constrained MST problems with node-degree and path-delay constraints were defined as DDC-MST problems. A DDC-MST problem is formulated to identify an MST in a weighted undirected graph $G = (V, E, C, D)$ under both node-degree and path-delay constraints, where V is a set of $n = |V|$ nodes $\{v_1, v_2, \ldots, v_n\}$; edge set $E = \{(i, j) \mid$ an edge exists between (v_i, v_j) in $G\}$; set $C = \{c_{i,j} | (i, j) \in E, c_{i,j} \ge 0\}$ defines the associated (nonnegative) edge costs over the edge set E; and set $D = \{d_{i,j} | (i, j) \in E, d_{i,j} \ge 0\}$ defines the associated (nonnegative) edge delays over the edge set E. The DDC-MST problem is clarified as follows.

- **Input:** A weighted undirected graph $G = (V, E, C, D)$, and two nonnegative values φ (node-degree constraint) and λ (path-delay constraint).
- **Objective function:** Identify a minimal-cost spanning tree $T = (V, E^T) \subseteq G$ under the following constraints. For spanning tree T, the cost $Cost(T)$ is defined as the summation of all edge costs for edges included in T; in other words, $Cost(T) = \sum_{(i,j)\in E^T} C_{i,j}$. Without loss of generality, we assume that T is associated with the root node $v_s = v_1$ (unless otherwise stated).
- **Constraints:**
  (i) *Path delay:* The path $P_t = (V_t, E_t) \subseteq T$ comprises a sequence of nodes V_t from root node v_s to leaf node v_t and the edges E_t between two adjacent nodes in the path. For P_t, the path-delay constraint $delay(P_t)$, or simply $delay(t)$, is defined as $\sum_{(i,j)\in E_t} d_{i,j}$. Furthermore, for T, the maximal path delay is defined as $delay(T) = \max\{delay(P_t) | t$ is a node in $T\}$. The path-delay constraint requires the path delay of each node in T to be equal to or less than λ; that is, $delay(T) \le \lambda$.
  (ii) *Node degree:* Each node in T has a maximum of φ child nodes.

2. Linear Programming Model

The aforementioned DDC-MST problem can be modeled as a binary integer linear programming problem, as follows [39].

Objective function:

$$\min \sum_{(i,j)\in E} c_{ij} x_{ij} \tag{1}$$

Subject to:

$$\sum_{v_j \in A(v_i)} x_{ij} \le \varphi, \forall v_i \in V \tag{2}$$

$$\sum_{v_i \in B(v_j)} x_{ij} = 1, \forall v_j \in V - \{v_s\} \tag{3}$$

$$delay(j) + M(1 - x_{ij}) \ge delay(i) + d_{ij} \tag{4}$$

$$0 \le delay(i) \le \lambda, \forall v_i \in V \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad (6)$$

where $x_{i,j}$ is a binary variable indicating whether the edge (i, j) is included in T (only where $x_{i,j} = 1$ is it included); set $A(v_i) = \{v_j | v_j \in \text{child\_node}(v_i)\}$ is a set of all child nodes in v_i; and set $B(v_j) = \{v_i | v_i \in \text{parent\_node}(v_j)\}$ is a set of all parent nodes of v_j in the G. The objective function shown in Eq. (1) is minimizes the total cost of T under constraints expressed in Eqs. (2)-(6). Specifically, the degree constraint in Eq. (2) specifies that all nodes in the spanning tree have no more than φ child nodes; Eq. (3) states that all nodes except the root node v_s have exactly one parent node; Eq. (4) states that the path delay from the root node v_s to node v_j is equal to or more than that from v_s to node v_i when edge (i, j) is included in the spanning tree (here, M is a sufficient large number; i.e., the sum of all edge delays in G); and Eq. (5) limits the path delay from v_s to v_i to a value equal to or less than the path-delay constraint λ.

3. Computational Complexity

For a DDC-MST problem, both node degree and path delay must be considered when constructing the MST. Although existing algorithms can solve unconstrained MST problems efficiently, an unconstrained MST problem could be difficult to solve when additional constraints are imposed. Previous studies have shown that MST problems with either node-degree or path-delay constraints are NP-hard [26, 33]. Consequently, the DDC-MST problem is equally difficult.

By definition, for an NP-hard problem such as the DDC-MST problem, identifying an efficient polynomial-time algorithm to solve the problem is difficult. Accordingly, the DDC-MST problem must be solved using heuristic (or meta-heuristic) algorithms.

IV. PROPOSED METHOD

The PLCMST (PSO-Like algorithm for Constrained MST problems) proposed in this study is an efficient algorithm for solving DDC-MST problems.

1. PLCMST Algorithm

Fig. 1 presents the framework of the proposed PLCMST algorithm. The key design components are the fast MST construction method (Line 6), fitness-evaluation function (Line 7), and evolutionary strategy (Line 18). In each iteration (Lines 4-18), the current fitness value F_T_{iBest} is first set to zero. Subsequently, the algorithm attempts to construct the MST T_k (Line 6) based on the current configuration of each particle k. Next, the fitness score of T_k is evaluated (Line 7), and T_{iBest} is updated with the optimal tree from the current iteration, and T_{gBest} is updated with a known optimal tree if one exists (Lines 8-15). Finally, the heuristic function $\eta(i, j)$ is updated after each iteration (Line 18). The heuristic function $\eta(i, j)$ plays a critical role; it is responsible for generating the evolutionary parameter matrix for constructing the spanning tree in each iteration for each particle.

The following subsections detail the spanning tree construction function "RPrim_E()" (Line 6) and the update rule for the heuristic function $\eta(i, j)$.

```
Procedure PLCMST
    //Input: (G, λ, φ);
    //Output: MST T_gbest;
01: Initialization();
02: m = 0; // starting from run 0;
03: While (m <MaxRun)
04:     F_T_iBest = 0;          // reset current run value
05:    For k = 1 to K       //K is number of particles
06:       T_k = Rprim_E();  //Construct an MST using Prim method
07:       F_k = Fitness(T_k);  //Evaluate the fitness of T_k
08:       If(F_k > F_T_iBest)    //best solution of current run
09:          T_ibest = T_k;
10:          F_T_ibest = F_k;
11:       End-if
12:       If(F_T_ibest > F_T_gbest) //best solution up to now
13:          T_gbest = T_ibest;
14:          F_T_gbest = F_T_ibest;
15:       End-if
16:    End-for
17:    m++; //Increase the run
18:    update η(i,j) For each edges (i,j) //Update the system parameters
19: End-While;
20:Return T_gbest; //Return the global best solution
End.
```

Fig. 1. Framework of the PLCMST algorithm.

```
Procedure Rprim_E
    //Input: (G, λ, η);
   //Output: Spanning tree T;
01: Let p initially be a zero vector,V′ = {v_s}, and
02: V_c = {v_j | (s, j) ∈ E, d_sj < λ};
03: Let p(j) = s for all v_j ∈ V_c;
04: While V_c ≠ empty_set
05:    Select edge (a, b) from E_c according to the PRP rule, Ω = E_c
06:       where E_c = {(i, j) | v_i ∈ V′, v_j ∈ V_c, (i, j) ∈ E, delay(j) < λ};
07:    Concatenates (a,b) to the constructed tree T;
08:    Move v_b from V_c to V′;
09:    For each v_k ∉ V′, v_k adjacent to v_b, and delay(b) + d_bk < λ
10:     If  v_k ∉ V_c
11:          Add v_k to V_c; Let p(k) = b;
12:     Else
13:          Select edge (x, k) from {(b, k), (p(k), k) } according to
14:             the PRP rule With  Ω ={(b, k), (p(k), k) }; Let p(k) = x;
15:     End-if
16:    End-for
17: End-while
18: Return T;
End.
```

Fig. 2. Spanning tree construction algorithm.

2. Spanning Tree Construction

This study adopted spanning tree construction scheme presented in [39], where the evolutionary rules involved in the heuristic function $\eta(i, j)$ are different. Fig. 2 shows the spanning tree construction procedure expressed in pseudocode. First, vector $\boldsymbol{P}$ is used to track the parent node of node v_j in $p[j]$ (Line 1). The set V' is used to track the nodes currently included in the constructed spanning tree. The set V_c comprises a set of candidate nodes for inclusion in the spanning tree in the subsequent iteration. A node is considered to be a candidate node if the following conditions are upheld: 1) it is not yet included in the spanning tree; 2) it has an edge connected to a node that is included in the spanning tree (i.e., a node in set V'); and 3) the path delay of the node is less than the constraint value λ. In addition, a set of candidate edges is maintained as $E_c = \{(i, j) \mid v_i \in V', v_j \in V_c, (i, j) \in E, delay(j) < \lambda\}$.

The spanning tree construction algorithm starts by initializing the root node v_s (Lines 1-3), and then by incrementally constructing the MST T (Lines 5-16). The code in the ***for*** loop (Lines 9-16) maintains the candidate set V_c and vector $\boldsymbol{P}$ (i.e., the parent node information). The MST construction algorithm, which is similar to the one proposed by Prim [29], can be considered as a randomized version of Prim's algorithm. The main difference between the two algorithms is the edge-selection strategy; Prim's algorithm always selects the minimal-cost (or weight) edge from the set of candidate edges (without considering the path delay), whereas the proposed algorithm uses a pseudorandom proportional (PRP) rule to select an edge for inclusion in the spanning tree (Lines 5-8). The PRP rule assigns a high selection probability to the edges with comparatively lower edge cost, whereas those with relatively higher edge costs are allocated a lower selection probability (but not a zero probability).

3. Edge Selection

This study adopted the edge-selection rule proposed in [39]. The edge-selection rule in the spanning tree construction is based on the aforementioned PRP rule. The PRP rule proceeds as follows:

Step 1: set the selection probability prob(i, j) of each candidate edge (i, j) in the candidate edge set Ω, as shown in Eq. (7).

$$\text{prob}(i, j) = \begin{cases} \dfrac{\eta(i, j)^{\beta}}{\sum_{(i,j)\in\Omega} \eta(i, j)^{\beta}} & \text{, if } (i, j) \in \Omega \\ 0 & \text{, otherwise} \end{cases} \tag{7}$$

Step 2: generate a random number $q = rand(0, 1)$; if $q > q_0$; subsequently, a predefined threshold uses a roulette-wheel selection scheme (also known as the a fitness proportionate selection scheme in GAs) to select an edge (a, b) from the candidate edge set Ω according to the selection probability (Eq. (7)). Otherwise, $q \le q_0$, in which case the algorithm selects the edge (a, b) with the highest selection probability, which is determined based on $(a, b) = arg\ \max_{(i,j)\in\Omega}\{\text{prob}(i, j)\}$.

Both q_0 and β are constant numbers that should be specified a priori as system parameters. The heuristic function $\eta(i, j)$ was adopted from the heuristic function proposed in [39], although a different approach is used to set the parameter values (detailed in the following subsections). The two versions of the heuristic function $\eta(i, j)$ adopted from [39] are expressed in Eqs. (8a) and (8b).

$$\eta(i, j) = \begin{cases} \dfrac{1}{\alpha_{ij}|V|} & \text{, if } g_i + 1 > \varphi \\ \dfrac{1}{\alpha_{ij}} & \text{, otherwise} \end{cases} \tag{8a}$$

$$\eta(i, j) = \begin{cases} \dfrac{1}{\alpha_{ij}|V|} & \text{, if } g_i + 1 > \varphi \\ \dfrac{1}{\alpha_{ij}\dfrac{delay(i) + d_{ij}}{\lambda}} & \text{, otherwise} \end{cases} \tag{8b}$$

where g_i represents the out-degree of node v_i, and φ and λ denote the node-degree and path-length constraint values, respectively. In Eqs. (8a) and (8b), the value of α_{ij} changes dynamically during each iteration of spanning tree construction procedure (Line 18, Fig. 1). The update rules for parameter α_{ij} are discussed in following subsections. Any edge (i, j) with a high node degree at node v_i ($g_i + 1 > \varphi$) is penalized, and the corresponding value in the heuristic function $\eta(i, j)$ is decreased, thereby decreasing the probability of that edge being included in the spanning tree (Eq. (7)). Similarly, as shown in Eq. (8b), an edge is penalized if it violates the path-delay constraint ($\dfrac{delay(i) + d_{ij}}{\lambda} > 1$; i.e., $delay(j) > \lambda$). The difference between Eqs. (8a) and (8b) is whether the delay constraints are considered (the following subsections show that α_{ij} is related only to the edge cost). Ideally, Eq. (8a) assigns a higher selection probability to the edges with comparatively lower edge costs, whereas Eq. (8b) weights the score relative to the value of the path-delay constraint. Consequently, the rule in Eq. (8a) is more aggressive in order to increase the probability of including edges with low edge-cost values at the risks of violating the path constraints. By contrast, the condition in Eq. (8b) is comparatively more conservative by reducing the probability of including edges with high path-delay values. As recommended in [39], we balanced the two effects by alternating between the two rules in each iteration.

4. Fitness Evaluation

In the proposed algorithm (Fig. 1), the fitness of the spanning tree construction is evaluated during each iteration (Line 7, Fig. 1). Assume that the spanning tree $T = (V', E')$ is con-

structed according to the spanning tree construction algorithm (Fig. 2) based on the input graph $G = (V, E)$. As recommended in [39], we applied the fitness function $F(T)$ shown in Eq. (9).

$$F(T) = \gamma^{|V|-|V'|}\left(\frac{1}{\sum_{(i,j)\in E'} C_{ij}}\right)\prod_{V_i \in V'} \Phi(g_i - \varphi) \tag{9}$$

$$\Phi(g_i - \varphi) = \begin{cases} 1, if\ (g_i - \varphi) \le 0 \\ \gamma, if\ (g_i - \varphi) > 0 \end{cases} \tag{10}$$

where γ is a positive constant with value of less than 1 (in this study, $\gamma = 0.5$), g_i is the out-degree of node v_i, and φ denotes the upper bound of the out-degree of a node to be allowed. By definition, an MST should cover all nodes (i.e., $|V| = |V'|$). However, the spanning tree construction algorithm might construct an incomplete tree (i.e., $|V'| < |V|$) because of its random edge-selection behavior. The term $\gamma^{|V|-|V'|}$ (γ is a constant with a value of less than 1) in Eq. (9) is a penalty term that is applied when a constructed tree T is an incomplete MST. Furthermore, the step function $\Phi(\cdot)$ provides an additional penalty term for any node violating the node-degree constraints.

5. Evolutionary Strategy

The rules for setting the heuristic function $\eta(i, j)$ are one of the key design features of the proposed algorithm. Eq. (11) shows the setting of the key parameter α_{ij} involved in the heuristic function $\eta(i, j)$ (Eqs. (8a) and (8b)). Initially, the value of α_{ij} for edge (i, j) is set at c_{ij} (i.e., the edge cost). Subsequently, after each iteration, α_{ij} is recalculated and the value of $\eta(i, j)$ is updated (Line 18, Fig. 1) to include the edges (i, j) in the spanning tree with the lowest cost in the current iteration (i.e., $(i, j) \in E_{iBest}$) or those among all previous iterations (i.e., $(i, j) \in E_{gBest}$). Consequently, for each edge (i, j), we set the value $\alpha_{ij} = c_{ij}/4$ for any edge included in the edges of minimal-cost spanning tree in both current iteration (i.e., E_{iBest}) and in those among all previous iterations (i.e., E_{gBest}). If the edge (i, j) is included in either E_{iBest} or in E_{gBest} (but not in both), then the weighting is doubled (i.e., $\alpha_{ij} = c_{ij}/2$); conversely, if the edge (i, j) is included in neither E_{iBest} nor E_{gBest}, the weighting is doubled (i.e., $\alpha_{ij} = c_{ij}$).

$$\alpha_{ij} = \begin{cases} \frac{c_{ij}}{4}, & \text{if } (i,j) \in A = \{E_{iBest} \cap E_{gBest}\} \\ \frac{c_{ij}}{2}, & \text{if } (i,j) \in \{E_{iBest} \cup E_{gBest}\}/A \\ c_{ij}, & \text{otherwise} \end{cases} \tag{11}$$

V. RESULTS

The performance of the proposed PLCMST algorithm was assessed by measuring the solution quality and computation time. We compared its performance with that of two other algorithms—the NGA [39] and ADDCMST [40]. The NGA is a GA-based algorithm, whereas the ADDCMST is an ACO-based algorithm. All three algorithms tested in this study are soft-computing approaches, which trade solution quality with computation time. The results were compared using LINGO (version 11.0.0.29) [7], a widely used software application for solving linear and nonlinear optimization problems. The solutions generated by LINGO are guaranteed to be optimal, which provided a baseline for comparing the solutions from the NGA, ADDCMST, and PLCMS.

1. Simulation Setup

The NGA, ADDCMST, and PLCMST were implemented using Microsoft Visual C++ 2008. All tests were performed on a personal computer with an Intel processor (Core2 Duo 2.2 GHz) and 3 GB RAM (DDR2, 333MHz).

No public benchmark data exist to test the performance of the DDC-MST problems; therefore, we adopted the criteria for generating the test data from [39], which is the source of one of the algorithms examined in this study. Because it would be impossible to generate graphs identical to those used in [39], we generated test graphs with identical or similar characteristics. In addition, the constraint rules were adopted from [39]. Each test graph is associated with two out-degree upper bounds ($\varphi = 3$ or 5) and two path-delay upper bounds ($\lambda = 2\Psi$ or 4Ψ), where Ψ is the maximal shortest path delay from root node v_s in the test graph, which can be expressed as $\Psi = \max\{shortest\_path\_delay(P_t) |\ t \text{ is a node in } T\}$. The value of Ψ for a given test graph $G = (V, E)$ can be obtained using a conventional shortest-path algorithm, such as the one proposed by Dijkstra [9], in which the edge-cost value is replaced with the edge-delay value. Combining the two constraints yields four possible combinations of path delay and out-degree upper bounds for each test graph.

Similar to the procedures presented in [39], the test graphs $G = (V, E, C, D)$ were generated using a random graph generator, which was also used in setting the node degree, constructing the edges, and assigning edge-cost and edge-delay values. Each group of test graphs is denoted as *Nx-y*, where *x* is node size and *y* is edge size. To produce a data set that was similar to the one used in [39], we compared the graph characteristics (e.g., maximal, minimal, and mean node degree, MST cost, maximal degree of the MST, and maximal shortest path delay) and excluded any graph that was dissimilar.

2. Simulation Results

(A) Performance comparison with LINGO:

Table 1 shows the results of five test graphs $G = (V, E, C, D)$ of identical node size (50) and edge size (100) but of varying edge-connection topology (denoted as N50-100). The computation time was limited to 4 hours (14400 s). Given the nature of soft-computing approaches, a random procedure could be embedded in the algorithms. Consequently, a single

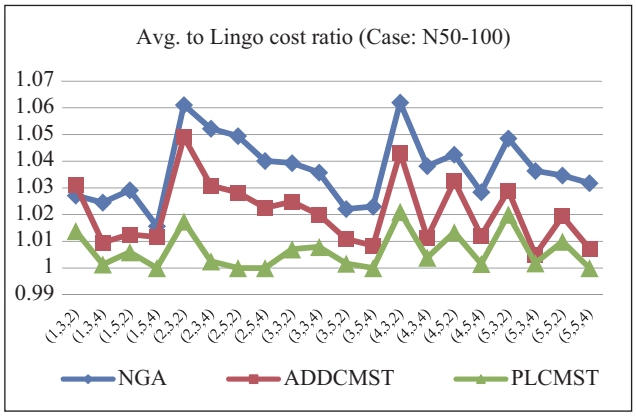

Fig. 3. Average spanning tree cost for graphs N50-100 (normalized with Lingo results).

instance of an input problem $I = (G(V, E, C, D), \varphi, \lambda)$ could yield varying results. Therefore, each instance of I was tested 30 times for each algorithm, and the mean of the results was calculated.

The results in Fig. 3 emphasize the difference in the results. Because the LINGO solution is an optimal one, we normalized the NGA, ADDCMST, and PLCMST results with those obtained using LINGO. In the figure, the y-axis represents the ratio of values (normalized with the LINGO results) for the corresponding input instance of $I = (G(V, E, C, D), \varphi, \lambda)$, and the x-axis represents various input configurations, denoted as (a, b, c), where a is the index of input graph G, b is the value of the node-degree constraint φ, and c is the value of the path-delay constraint λ expressed in units of Ψ (i.e., the maximal shortest path delay from the root node of the G). In Fig. 3, the first entry (1, 3, 2) along the x-axis corresponds to the first entry in Table 1 ($G = 1$, $\varphi = 3$, $\lambda = 2\Psi$). As shown in the first entry in Table 1, the mean costs of LINGO, NGA, ADDCMST, and PLCMST are 1510, 1551, 1557, and 1531, respectively. Accordingly, the normalized ratios for the NGA (1551/1510), ADDCMST (1557/1510), and PLCMST (1531/1510) are 1.0271, 1.0311, and 1.0139, respectively, which correspond to the first entry in Fig. 3.

The N50-100 results (Table 1) show that the proposed PLCMST algorithm outperformed the NGA and ADDCMST in all 20 instances. Moreover, the PLCMST yielded lower costs and shorter computation times. Fig. 3 shows that the N50-100 results obtained using the proposed method are similar to the optimal solutions obtained using LINGO. The results obtained using the proposed method are within 2% larger than the optimal solutions; moreover, most of those cases are less 1% larger. When using LINGO, the computation time ranged from 1 to 928 seconds, whereas that of the proposed PLCMST algorithm is less than 4 seconds for all test cases.

Subsequently, we increased the edge size to 250. Table 2 shows the results of five test graphs with 50 nodes and 250 edges (denoted as N50-250). Because the graphs were large, LINGO was unable to obtain a solution within the allocated

Table 1. Results for test graphs N50-100 (node size = 50, edge size = 100).

Graph (N50-100)			LINGO		NGA		ADDCMST		PLCMST	
G	φ	λ	cost	time	cost	time	cost	time	cost	time
1	3	2Ψ	1510	447	1551	2.963	1557	2.206	1531	1.901
	3	4Ψ	1467	114	1503	3.212	1481	2.606	1469	2.055
	5	2Ψ	1510	928	1554	2.960	1529	2.218	1519	1.872
	5	4Ψ	1467	25	1490	3.206	1484	2.606	1467	2.055
2	3	2Ψ	2407	8	2554	3.499	2525	2.642	2449	2.459
	3	4Ψ	2393	4	2518	3.632	2467	2.887	2399	2.488
	5	2Ψ	2407	3	2526	3.505	2475	2.642	2407	2.446
	5	4Ψ	2393	3	2489	3.639	2447	2.907	2393	2.493
3	3	2Ψ	2416	3	2511	3.569	2476	2.741	2433	2.429
	3	4Ψ	2404	12	2490	3.608	2452	2.953	2423	2.489
	5	2Ψ	2397	1	2450	3.586	2423	2.793	2401	2.496
	5	4Ψ	2389	5	2444	3.628	2409	2.980	2389	2.503
4	3	2Ψ	2613	252	2775	2.746	2726	2.315	2668	1.161
	3	4Ψ	2567	57	2665	3.454	2596	2.794	2577	2.382
	5	2Ψ	2613	134	2724	2.762	2698	2.344	2648	1.198
	5	4Ψ	2567	30	2640	3.469	2598	2.810	2571	2.427
5	3	2Ψ	2245	8	2354	3.509	2310	2.751	2290	2.432
	3	4Ψ	2226	3	2307	3.633	2237	3.020	2230	2.560
	5	2Ψ	2220	6	2297	3.517	2263	2.740	2242	2.425
	5	4Ψ	2201	3	2271	3.600	2217	2.945	2201	2.469

time limit (i.e., 4 h) for all of the N50-250 tests (five test graphs, each with four unique constraint setting on (φ, λ)), indicating that conventional optimization packages, such as LINGO, are unfeasible for solving complex DDC-MST problems, which justifies the need to use soft-computing approaches to obtain a solution, particularly for a large problems.

For those complex test cases, because the optimal values could not be obtained using LINGO, we used a low bound instead. The low bound values shown in Tables 2-4 denote the cost of the unconstrained MSTs. The value can be obtained easily by using conventional MST algorithms, such as Prim's algorithm [29]. Moreover, the true optimal value is equal to or more than the lower bound. Because the DDC-MST problem is NP-hard, identifying the true optimal value of the problem is complex, particularly for large problems. Therefore, we used the low bound value as a reference to assess the solution quality. Fig. 4, we plot the N50-250 MST cost results. Similar to Fig. 3, the value of each data point in the plot is normalized by the lower-bound value. The results show that the proposed PLCMST algorithm outperforms the NGA and ADDCMST, and the solution quality of the proposed method is comparable (i.e., the values are near the lower bound).

(B) Performance under general cases:

In addition to the cases for node size of 50, we evaluated numerous randomly generated graphs of varying size (N = 50, 100, 200, and 300) edge density. The 19 types of test graph are shown in Tables 3 and 4. The edge size was carefully selected

Table 2. Results for test graphs N50-250 (node size = 50, edge size = 250).

Graph (N50-250)			LINGO		NGA		ADDCMST		PLCMST		Low bound
G	φ	λ	cost	time	cost	time	cost	Time	cost	time	cost
1	3	2Ψ	---	14400	2308	4.533	2279	3.355	1916	3.333	1520
	3	4Ψ	---	14400	1936	5.577	1901	4.354	1585	3.923	1520
	5	2Ψ	---	14400	2198	4.572	2162	3.426	1783	3.467	1520
	5	4Ψ	---	14400	1890	5.564	1892	4.380	1560	3.969	1520
2	3	2Ψ	---	14400	2185	5.022	2176	3.829	1747	3.631	1494
	3	4Ψ	---	14400	1838	5.716	1822	4.719	1538	4.030	1494
	5	2Ψ	---	14400	2103	5.085	2107	3.860	1718	3.713	1494
	5	4Ψ	---	14400	1831	5.740	1801	4.696	1527	3.989	1494
3	3	2Ψ	---	14400	2384	4.674	2335	3.508	1959	3.260	1623
	3	4Ψ	---	14400	2051	5.619	1988	4.454	1695	3.904	1623
	5	2Ψ	---	14400	2216	4.712	2206	3.629	1845	3.243	1623
	5	4Ψ	---	14400	1959	5.563	1995	4.500	1690	3.888	1623
4	3	2Ψ	---	14400	2347	4.720	2289	3.436	1983	3.611	1558
	3	4Ψ	---	14400	1865	5.668	1896	4.415	1589	4.119	1558
	5	2Ψ	---	14400	2253	4.761	2251	3.428	1939	3.688	1558
	5	4Ψ	---	14400	1848	5.683	1870	4.447	1580	4.146	1558
5	3	2Ψ	---	14400	2377	4.996	2264	3.863	1924	3.670	1655
	3	4Ψ	---	14400	1970	5.755	1920	4.691	1695	4.143	1655
	5	2Ψ	---	14400	2231	5.089	2180	3.962	1844	3.748	1655
	5	4Ψ	---	14400	1922	5.796	1887	4.725	1660	4.138	1655

Table 3. Results for all test graphs with low edge density (≤ 0.3).

		NGA			ADDCMST			PLCMST			Low bound
Edge Desity	graph	Avg cost	Avg exec time (s)	Suc rate	Avg cost	Avg exec time (s)	Suc rate	Avg cost	Avg exec time (s)	Suc rate	Avg cost
≤ 0.1	N50-100	2306	3.39	100	2268	2.70	100	2235	2.24	100	2201
	N100-200	3798	9.17	100	3689	8.00	100	3550	7.14	100	3470
	N200-400	5281	31.88	96	5105	25.08	100	4862	18.99	96	4636
	N300-600	13744	45.27	96	13229	39.80	100	12515	39.18	96	11857
0.1-0.3	N50-250	2086	5.24	100	2061	4.08	100	1739	3.78	100	1570
	N100-1000	4301	17.31	90	4458	14.21	97	2894	14.08	98	1996
	N200-2000	8046	63.40	81	11684	48.38	82	5186	44.44	96	3636
	N300-3000	10026	78.52	74	10738	65.06	59	6312	66.34	79	4527
average		6198	31.77	92	6654	25.91	92	4911.6	24.52	95.6	4237

Table 4. Results for all test graphs with high edge density(≧ 0.4).

		NGA			ADDCMST			PLCMST			Low bound
Edge Desity	graph	Avg cost	Avg exec time(s)	Suc rate	Avg cost	Avg exec time(s)	Suc rate	Avg cost	Avg exec time(s)	Suc rate	Avg cost
0.4-0.6	N50-500	2545	11.06	81	2360	4.98	78	1865	4.53	88	1282
	N100-2000	4857	26.38	42	6962	17.93	34	2736	18.40	77	1768
	N200-8000	0	0	0	0	0	0	4136	68.52	43	2579
	N300-17000	12750	121.84	71	14253	90.31	50	4826	115.60	99	2482
0.6-0.8	N50-750	2193	12.04	99	2168	5.54	92	1376	5.53	100	1052
	N100-3000	5370	27.78	16	7881	18.22	11	2896	19.53	79	1518
	N200-12000	10709	83.84	35	20763	62.01	17	3878	78.01	80	2187
	N300-27000	14721	145.24	47	16893	107.24	49	3691	151.23	100	2442
≧0.8	N50-1000	2600	12.31	96	2554	5.93	77	1537	5.84	100	1017
	N100-4000	5605	28.57	78	6249	19.49	72	2925	21.09	93	1417
	N200-16000	15479	93.79	58	38248	71.74	62	2803	93.56	100	1978
Average		7682.9	56.28	62.3	11833	40.34	53.4	2969.9	52.89	87.2	1793.4

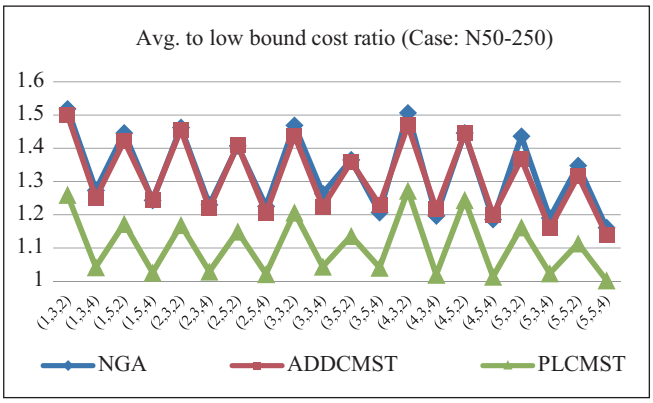

Fig. 4. Average spanning tree cost for graphs N50-250 (normalized with low bound results).

to consider both sparse and dense edge graphs, in which we used edge density to indicate the degree of sparseness (or denseness) of the graph. Edge density is defined as the number of edges in a graph divided by the maximal number of possible edges in a graph with an identical number of nodes, which is equal to $n(n-1)/2$ for a graph with n nodes. For each graph group *Nx-y*, we randomly selected five graphs $G(V, E, C, D)$ of identical node size and edge size but of varying edge-connection topology, thereby obtaining graphs with unique edge-cost and delay values. Consequently, we obtained $19 \times 5 = 95$ test graphs. Similar to the N50-100 (Table 1) and N50-250 (Table 2) cases, each test graph $G(V, E, C, D)$ is associated with four configurations of constraints (i.e., $\varphi = 3$ or 5; $\lambda = 2\Psi$ or 4Ψ); consequently, we obtained $95 \times 4 = 380$ test input instances.

Tables 3 and 4 show the results of the 380 test input instances. For clarity, we summarized the results based on the graph group *Nx-y*, each comprising five graphs of identical node size and edge size, as well as $5 \times 4 = 20$ test input instances. To smooth the stochastic effects resulting from the nature of soft-computing algorithms, each input instance $I = (G(V, E, C, D), \varphi, \lambda)$ was tested 30 times for the NGA, ADDCMST, and PLCMST. Accordingly, each graph group was subject to 600 test iterations. For each iteration, we noted the executed algorithm, cost of the identified optimal spanning tree, and execution time required to obtained a solution. All test input instances (particularly the large problems) where no feasible results were obtained were noted as failed tests. Assume that algorithm *A* performed 600 test iterations on graph group *Nx-y*. Subsequently, the number of failed tests f can be used to calculate the success rate of algorithm *A* for that graph group as $(600-f)/600$. The success rate represents the probability of an algorithm generating a feasible solution identical termination criteria. Subsequently, for the successful tests, the mean cost value and execution time was calculated for each graph group and algorithm. To facilitate a simple comparison, we provide the lower-bound value of each case.

The data in Tables 3 and 4 were reorganized to facilitate a comparison from various perspectives. To obtain a performance ratio for comparing the efficiency of the PLCMST with that of the NGA and ADDCMST, the data generated from the NGA and ADDCMST were normalized with the corresponding data obtained from the PLCMST. Among the results shown in Table 5, no values are listed for the N200-8000 case because both the NGA and ADDCMST failed to generate results (as shown in Table 4).

Based on the 380 test input instances, Table 5 shows that the ratio of the solution quality (the columns entitled "Avg. Cost") is higher than 1 for both the NGA and ADDCMST, implying that the proposed PLCMST algorithm generated superior spanning trees (i.e., lower computational cost). In addition, the proposed method exhibited superior performance for the high edge-density cases. Regarding the success rate, Table 5 shows that the ratio is more than 1 in most cases, implying that the PLCMST is more likely to generate a feasible solution than the NGA and ADDCMST are. However, two exceptions occurred in the N200-400 and N300-600 cases, where the success rate of the ADDCMST is marginally higher than that of the PLCMST.

Regarding the execution time, the PLCMST generated superior solutions (i.e., lower computational cost) in less time than the NGA and ADDCMST required in most cases. For the cases where the PLCMST was slower than the other two algorithms (i.e., execution time ratio < 1), the quality of the solution was superior (cost ratio > 2). The only exception occurred in the N300-3000 case, where the ADDCMST yielded a cost ratio of 1.7. This discussion shows that the proposed PLCMST algorithm outperforms the NGA and ADDCMST; in most cases, it generated better solutions in less time.

VI. CONCLUSION

This paper presents a novel PSO-based metaheuristic algorithm for solving DDC-MST problems, in which we are asked to identify an MST under both degree and delay constraints. The performance of the proposed PLCMST algorithm was compared with that of LINGO, a widely used software package for linear and nonlinear optimization problems. Our experimental results show that the proposed algorithm generated high-quality solutions for the N50-100 test graphs (Table 1). For all 20 test graphs, the performance ratio (of minimal cost obtained by the proposed algorithm to that obtained by optimal solutions) is within 1.02, and the computation time was considerably less than that required by LINGO. Subsequently, we extended the problem size by increasing the size of the graphs (N50-250). We observed that LIGNO was unable to solve these test graphs within 4 hours. Under identical test conditions, the PLCMST obtained suitable solutions close to the low bound within 5 seconds (Table 2). The results indicate that conventional linear optimization packages, such as LINGO, may be unfeasible for solving relatively large DDC-MST problems.

In addition, we compared the performance of the PLCMST with that of the NGA, which is based on GA metaheuristics,

Table 5. Performance comparison between NGA/ADDCMST/PLCMST.

		NGA /PLCMST			ADDCMST/PLCMST		
Edge desity	graph	Avg cost	Avg exec time (s)	Succ rate	Avg cost	Avg exec Time (s)	Succ rate
≤ 0.1	N50-100	1.03	1.51	1	1.01	1.2	1
	N100-200	1.07	1.28	1	1.04	1.12	1
	N200-400	1.09	1.68	1	1.05	1.32	1.04
	N300-600	1.1	1.16	1	1.06	1.02	1.04
0.1-0.3	N50-250	1.2	1.39	1	1.19	1.08	1
	N100-1000	1.49	1.23	0.92	1.54	1.01	0.99
	N200-2000	1.55	1.43	0.84	2.25	1.09	0.85
	N300-3000	1.59	1.18	0.94	1.7	0.98	0.75
Average (For ≤ 0.3)		1.27	1.36	0.96	1.36	1.10	0.96
0.4-0.6	N50-500	1.36	2.44	0.92	1.27	1.1	0.89
	N100-2000	1.78	1.43	0.6	2.54	0.97	0.51
	N200-8000	x	x	0	x	x	0
	N300-17000	2.64	1.05	0.72	2.95	0.78	0.51
0.4-0.8	N50-750	1.59	2.18	0.99	1.58	1	0.92
	N100-3000	1.85	1.42	0.7	2.72	0.93	0.49
	N200-12000	2.76	1.07	0.44	5.35	0.79	0.21
	N300-27000	3.99	0.96	0.47	4.58	0.71	0.49
≧0.8	N50-1000	1.69	2.11	0.96	1.66	1.02	0.85
	N100-4000	1.92	1.35	0.84	2.14	0.92	0.77
	N200-16000	5.52	1	0.58	13.65	0.77	0.6
Average (For ≧0.4)		2.51	1.5	0.66	3.847	0.9	0.57

and the ADDCMST, which is an ACO algorithm. Previous studies have shown that the NGA and ADDCMST can solve DDC-MST problems efficiently. To demonstrate the efficiency and effectiveness of the PLCMST, we compared it with those two algorithms. Intensive simulations were performed to evaluate the performance of the NGA, ADDCMST, and PLCMST. We generated 380 instances $I = (G(V, E, C, D), \varphi, \lambda)$ of graphs with varying node size (N = 50, 100, 200, and 300), edge size (ranging from 100 to 2700), and edge density (from 0.013 to 0.816). As shown in Table 5, the PLCMST generated superior spanning trees (i.e., lower computational cost). Moreover, the performance of the PLCMST was even higher for the high edge-density cases. Regarding the success rate, the PLCMST exhibited a higher probability of generating a feasible solution in comparison with the NGA and ADDCMST, with two exceptions among the 380 tests. Regarding the execution time, the proposed algorithm generated superior results (i.e., lower computational cost) with less time. For the cases where the PLCMST took longer, we observed that it generated solutions that were markedly superior (cost ratio > 2) with only one exception. Compared with the NGA and ADDCMST, the proposed PLCMST algorithm generally obtained superior solutions at less computational cost in almost all of the cases examined in this study.

REFERENCES

1. Aissa, M. and Ben Mnaouer, A., "A new delay-constrained algorithm for multicast routing tree construction," *International Journal of Communication Systems*, Vol. 17, No. 10, pp. 985-1000 (2004).
2. Almeida, A. M., Martins, P., and De Souza, M. C., "md-MST is NP-hard for d ≥ 3," *Electronic Notes in Discrete Mathematics*, Vol. 36, pp. 9-15 (2010).
3. Bau, Y. T., Ho, C. K., and Ewe, H. T., "An ant colony optimization approach to the degree-constrained minimum spanning tree problem," *Proceeding of 2005 International Conference on Computational Intelligence and Security (CIS 2005)*, Part I, LNAI 3801, pp. 657-662 (2005).
4. Behle, M., Juenger, M., and Liers, F., "A primal branch-and-cut algorithm for the degree-constrained minimum spanning tree problem," *Proceeding of 6th Workshop on Experimental Algorithms (WEA 2007)*, LNCS 4525, pp. 379-392 (2007).
5. Benyahia, I., "A survey of ant colony optimization algorithms for telecommunication networks," *International Journal of Applied Metaheuristic Computing (IJAMC)*, Vol. 3, No. 2, pp. 18-32 (2012).
6. Binh, H. T. T. and Nguyen, T. B., "New particle swarm optimization algorithm for solving degree constrained minimum spanning tree problem," *Proceeding of 10th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2008)*, LNAI 5351, pp. 1077-1085 (2008).
7. Bui, T. N. and Zrncic, C. M., "An ant-based algorithm for finding degree-constrained minimum spanning tree," *Proceeding of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, Seattle, WA, USA, pp. 11-18 (2006).
8. Chen, L., Yang, Z., and Xu, Z., "A degree-delay-constrained genetic algorithm for multicast routing tree," *Proceeding of the Fourth International Conference on Computer and Information Technology*, Wuhan, China, pp. 1033-1038 (2004).
9. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms, 3rd Ed.*, The MIT Press, Cambridge, MA (2009).
10. Craveirinha, J., Climaco, J., Martins, L., Da Silva, C. G., and Ferreira, N., "A bi-criteria minimum spanning tree routing model for MPLS/overlay networks," *Telecommunication Systems*, Vol. 52, No. 1, pp. 203-215

(2013).
11. Deo, N. and Micikevicius, P., "A heuristic for a leaf constrained minimum spanning tree problem," *Congressus Numerantium*, Vol. 141, pp. 61-72 (1999).
12. Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP Completeness*, W.H. Freeman, New York (1979).
13. Ghaboosi, N. and Haghighat, A. T., "Tabu search based algorithms for bandwidth-delay-constrained least-cost multicast routing," *Telecommunication Systems*, Vol. 34, Nos. 3-4, pp. 147-166 (2007).
14. Gouveia, L., Moura, P., and DeSousa, A., "Spanning trees with generalized degree constraints arising in the design of wireless networks," *Proceeding of 2011 International Network Optimization Conference (INOC 2011)*, LNCS 6701, pp. 77-82 (2011).
15. Gruber, M., van Hemert, J., and Raidl, G. R., "Neighborhood searches for the bounded diameter minimum spanning tree problem embedded in a vns, ea, and aco," *Proceeding of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, Seattle, WA, USA, pp. 1187-1194 (2006).
16. Jovanovic, R. and Tuba, M., "Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem," *Computer Science and Information Systems (ComSIS)*, Vol. 10, No. 1, pp. 133-149 (2013).
17. Julstrom, B. A., "Codings and operators in two genetic algorithms for the leaf-constrained minimum spanning tree problem," *International Journal of Applied Mathematics and Computer Science*, Vol. 14, No. 3, pp. 385-396 (2004).
18. Krishnamoorthy, M. and Ernst, A. T., "Comparison of algorithms for the degree constrained minimum spanning tree," *Journal of Heuristics*, Vol. 7, No. 6, pp. 587-611 (2001).
19. Kruskal, J., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, Vol. 7, No. 1, pp. 48-50 (1956).
20. Kuo, H.-C., Chang, J.-R., and Liu, C.-H., "Particle swarm optimization for global optimization problems," *Journal of Marine Science and Technology*, Vol. 14, No. 3, pp. 170-181 (2006).
21. Leggieri, V., Haouari, M., and Triki, C., "A branch-and-cut algorithm for the Steiner tree problem with delays," *Optimization Letters*, Vol. 6, No. 8, pp. 1753-1771 (2011).
22. Liang, Y.-H., Chang, B.-J., and Lin, Y.-M., "Solve the tree setup problem and minimize control overhead for high-density members in delay-bounded distributed multicast networks," *Wireless Personal Communications*, Vol. 65, No. 4, pp. 875-894 (2012).
23. Martinez, L. C. and Da Cunha, A. S., "A parallel Lagrangian relaxation algorithm for the min-degree constrained minimum spanning tree problem," *Proceeding of 2nd International Symposium on Combinatorial Optimization (ISCO 2012)*, LNCS 7422, pp. 237-248 (2012).
24. Mauthe, A., Hutchison, D., Coulson, G., and Namuye, S., "Multimedia group communications: towards new services," *Distributed Systems Engineering*, Vol. 8, No. 3, pp. 197-210 (1996).
25. Moy, J., "Multicast routing extensions for OSPF," *Communications of the ACM*, Vol. 37, No. 8, pp. 61-67 (1994).
26. Narula, S. C. and Ho, C. A., "Degree-constrained minimum spanning trees," *Computers and Operations Research*, Vol. 7, No. 4, pp. 239-249 (1980).
27. Oencan, T., Cordeau, J.-F., and Laporte, G., "A tabu search heuristic for the generalized minimum spanning tree problem," *European Journal of Operational Research*, Vol. 191, No. 2, pp. 306-319 (2008).
28. Pham, Q. D., Deville, Y., and Van Hentenryck, P., "LS(Graph): a constraint-based local search for constraint optimization on trees and paths," *Constraints*, Vol. 17, No. 4, pp. 357-408 (2012).
29. Prim, R. C., "Shortest connection networks and some generalizations," *Bell System Technical Journal*, Vol. 36, pp. 1389-1401 (1957).
30. Qu, R., Xu, Y., Castro, J. P., and Landa-Silva, D., "Particle swarm optimization for the Steiner tree in graph and delay-constrained multicast routing problems," *Journal of Heuristics*, Doi: 10.1007/s10732-012 -9198 -2 (2012).
31. Ravi, R., Marathe, M. V., Ravi, S. S., Rosenkrantz, D.-J., and Hunt III, H. B., "Approximation algorithms for degree-constrained minimum-cost network-design problems," *Algorithmica*, Vol. 31, No. 1, pp. 58-78 (2001).
32. Reimann, M. and Laumanns, M., "A hybrid aco algorithm for the capacitated minimum spanning tree problem," *Proceeding of First International Workshop on Hybrid Metahuristics (HM2004)*, Valencia, Spain, pp. 1-10 (2004).
33. Salama, H. F., Reeves, D. S., and Viniotis, Y., "The delay constrained minimum spanning tree problem," *Proceeding of the Second IEEE Symposium on Computers and Communications*, Alexandria, Egypt, pp. 699-703 (1997).
34. Singh, A., "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Applied Soft Computing*, Vol. 9, No. 2, pp. 625-631 (2009).
35. Skorin-Kapov, N. and Kos, M., "A grasp heuristic for the delay-constrained multicast routing problem," *Telecommunication Systems*, Vol. 32, No. 1, pp. 55-69 (2006).
36. Soak, S.-M., Corne, D., and Ahn, B.-H., "A new encoding for the degree constrained minimum spanning tree problem," *Proceeding of 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2004)*, LNAI 3213, pp. 952-958 (2004).
37. Soak, S.-M. and Jeon, M., "The property analysis of evolutionary algorithms applied to spanning tree problems," *Applied Intelligence*, Vol. 32, No. 1, pp. 96-121 (2010).
38. Torkestani, J. A., "An adaptive heuristic to the bounded-diameter minimum spanning tree problem," *Soft Computing*, Vol. 16, No. 11, pp. 1977-1988 (2012).
39. Tseng, S.-Y., Huang, Y.-M., and Lin, C.-C., "Genetic algorithm for delay and degree-constrained multimedia broadcasting on overlay networks," *Computer Communications*, Vol. 29, No. 17, pp. 3625-3632 (2006).
40. Tseng, S.-Y., Lin, C.-C., and Huang, Y.-M., "Ant colony-based algorithm for constructing broadcasting tree with degree and delay constraints," *Expert Systems with Applications*, Vol. 35, No. 3, pp. 1473-1481 (2008).
41. Waitzman, D., Partridge, C., and Deering, S. E., "Distance vector multicast routing protocol," *IETF RFC 1075* (1988).
42. Xu, Y. and Qu, R., "A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems," *Applied Intelligence*, Vol. 36, pp. 229-241 (2012).
43. Xue, G. L., "Minimum-cost QoS multicast and unicast routing in communication networks," *IEEE Transactions on Communications*, Vol. 51, No. 5, pp. 817-824 (2003).