Available online at www.sciencedirect.com
ScienceDirect
Progress in Natural Science 18 (2008) 495–499
Progress in Natural Science
www.elsevier.com/locate/pnsc

Special report

A new algorithm for degree-constrained minimum spanning tree based on the reduction technique

Aibing Ning [a,*], Liang Ma [a], Xiaohua Xiong [b]

[a] School of Management, University of Shanghai for Science and Technology, Shanghai 200093, China
[b] College of Computer and Information, Shanghai Second Polytechnic University, Shanghai 201209, China

Received 29 October 2007; received in revised form 12 November 2007; accepted 12 November 2007

Abstract

The degree-constrained minimum spanning tree (DCMST) is an *NP*-hard problem in graph theory. It consists of finding a spanning tree whose vertices should not exceed some given maximum degrees and whose total edge length is minimal. In this paper, novel mathematical properties for DCMST are indicated which lead to a new reduction algorithm that can significantly reduce the size of the problem. Also an algorithm for DCMST to solve the smaller problem is presented which has been preprocessed by reduction algorithm.

Keywords: Degree-constrained minimum spanning tree; Reduction algorithm; Edge exchange technique; Combinatorial optimization

1. Introduction

The minimum spanning tree (MST) problem is an important problem in the design of communication networks and it can be solved in polynomial time [1,2]. However, in real networks, vertices (or nodes) are usually subject to some degree constraints. For example, exchanges or switches can only be physically connected to a limited number of linking wires. Also when designing a network for maximum reliability, introducing a degree constraint limits the damage that may be caused by a single exchange failure. Unlike the MST, the general degree-constrained minimum spanning tree (DCMST) is *NP*-hard [3–7].

Generally speaking, finding a DCMST is a difficult task [5], and several, mainly heuristic, solution approaches exist in the literature. Ribeiro and Souza [3] implemented a variable neighborhood search for generating good heuristic solutions for the DCMST. Recently, Andrade et al. [7] present a fast and effective heuristic for its solution. Using Lagrangian dual information, optimality of a solution can be proven in several cases. Goemans [8] designed a polynomial-time approximation algorithm for the DCMST problem with degree bound b for all vertices that finds a spanning tree of maximum degree at most $b + 2$ whose cost is at most the cost of the optimum spanning tree of maximum degree b. Volgenant [9], Knowles and Corne [10], Krishnamoorthy et al. [11] and Raidl [12] presented and compared several heuristics like simulated annealing approaches, evolutionary algorithms, Lagrangian relaxation and branch-and-bound methods on different classes of instances. Caccetta and Hill [13] also implemented a standard branch-and-cut algorithm which solves the DCMST problem exactly. We also reported an evolutionary algorithm for DCMST [14].

Different from these measures, the algorithm presented in this paper fully employs the mathematical properties of the problem to get a reduction technique which reduces the size of the problem. Then, based on the classic Kruskal's algorithm, it employs the edge exchange technique to get an algorithm to solve the problem which has been preprocessed by the reduction algorithm.

* Corresponding author. Tel.: +86 21 50762101.
E-mail address: nabnab@163.com (A. Ning).

1002-0071/$ - see front matter © 2007 National Natural Science Foundation of China and Chinese Academy of Sciences. Published by Elsevier Limited and Science in China Press. All rights reserved.
doi:10.1016/j.pnsc.2007.11.014

2. Notations and problem definition

2.1. Notations

Let us introduce some of the basic vocabulary and notions of graph theory. We will use the vocabulary and notions in the following parts of the paper.

A *path* $p(u,v)$ in a graph $G=(V,E)$ from vertex $u \in V$ to vertex $v \in V$ is a list $p(u,v)=(u,v_1,v_2,\ldots,v_k,v)$ of vertices of V such that (u,v_1), $(v_k,v) \in E$, and $(v_i,v_{i+1}) \in E$ for $i=1,\ldots,k-1$. A path $p(u,v)$ in G with $u=v$ is called a *cycle* in G. The *length* $|p(u,v)|$ of a *path* $p(u,v)=(u,v_1,v_2,\ldots,v_k,v)$ is defined by $k+1$, namely the number of edges between u and v in $p(u,v)$. If $(v,w) \in E$, then we call w and v *neighbors*. The *neighbors* of a vertex v is $N(v)=\{w|(v,w) \in E\}$. For $u,v \in V$, we abbreviate $N(u) \cup N(v)$ with $N(u,v)$. $N([v])=N(v) \cup \{v\}$ denotes the *closed neighborhood* of $v \in V$. The degree $d(v)$ of $v \in V$ is defined by $d(v)=|N(v)|$. Let $G=(V,E)$ be a graph and $V1 \subseteq V$. $G|_{V1}=(V1,E1), E1=\{(x,y)|x,y \in V1 \text{ and } (x,y) \in E\}$ is called the restriction from G to $V1$. A graph $G^1=(V^1,E^1)$ is called a *subgraph* of $G=(V,E)$ if $V^1 \subseteq V$ and $E^1 \subseteq E$. We denote the subgraph property with $G^1 \subseteq G$. For a graph $G=(V,E)$ and a vertex $u \in V$ we write $G-u$ to denote $(V\backslash\{u\}, E\backslash\{(x,y)|x=u \text{ or } y=u\})$, the graph G without the vertex u. For a set V^1 we write $G-V^1$ for $(V\backslash V^1, E\backslash\{(x,y)|x \subseteq V^1 \text{ or } y \subseteq V^1\})$.

Let $d^T(v_j)$ denote the degrees of the vertex v_j in the graph T, b_j denote the constraints on the degree $d^T(v_j)$ of each vertex v_j in the graph T, $w(v_a,v_b)$ the weight or cost of the edge (v_a,v_b), and T^* a degree-constrained minimum spanning tree of graph G.

2.2. Problem definition

The degree-constrained minimum spanning tree problem can be stated as follows:

Let $G=(V,E)$ be a connected weighted undirected graph, where $V=\{v_1,v_2,\ldots,v_n\}$ is the vertex set of G, and $E=\{e_1,e_2,\ldots,e_m\}$ is the edge set of G. Let $W=\{w_1,w_2,\ldots,w_m\}$ represent the weight or cost of each edge where the weight is restricted to be a non-negative real number.

Any subgraph of G can be described using a vector $\boldsymbol{X}=(x_1,x_2,\ldots,x_m)$ where each element x_i is defined by:

$$x_i=\begin{cases}1 & \text{if edge } e_i \text{ is part of the subgraph}\\ 0 & \text{otherwise}\end{cases} \tag{1}$$

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

(1) S contains all the vertices of G.
(2) S is connected and contains no cycles.

Now let T be the set of all spanning trees in G, a minimum spanning tree is defined as (2).

$$\min\left\{z(x)=\sum_{i=1}^{m} w_i x_i | x_i \in T\right\} \tag{2}$$

If b_j is the constraint on the degree $d^T(v_j)$ of each vertex v_j in the graph T, then a degree-constrained minimum spanning tree is defined as (3).

$$\min\left\{z(x)=\sum_{i=1}^{m} w_i x_i | d^T(v_j) \leqslant b_j, v_j \in V, x_i \in T\right\} \tag{3}$$

3. Algorithm

3.1. Mathematical properties

We introduce the properties of the problem that can be employed to reduce the size and the hardness of the problem.

Theorem 1. *All the incident edges of the pendant vertices (vertices of degree 1) should be included in the degree-constrained minimum spanning tree T^*.*

Proof. Since degree-constrained minimum spanning tree is a connected graph, so all the incident edges of the pendant vertices (vertices of degree 1) should be included in the T^* and should be deleted from graph G. Otherwise, the tree T^* must be a non-connected graph. All pendant vertices and their incident edges should be included in the degree-constrained minimum spanning tree. □

Theorem 2. *If $G=(V,E)$ is a connected undirected graph, $V_1=\{v_i|b_i=1 \text{ and } v_i \in V\}$, $E_1=\{(v_i,v_j) | v_i, v_j \in V_1, (v_i,v_j) \in E\}$, and $|V|>2$ then any edge in E_1 should be excluded from T^*.*

Proof. Suppose that an edge $(v_i,v_j) \in E_1$ and the edge (v_i,v_j) are in the T^*, since $b_i=1$ and $b_j=1$, then v_i, v_j cannot be connected with any other vertex in T^*, so any edge in E_1 should be excluded from T^*. □

Theorem 3. *Suppose v_k is a degree-2 vertex in the graph G with two neighbors v_i and v_j such that there do not exist any path $p(v_i,v_j)$ which do not visit the vertex v_k. That is, there do not exist any path $p(v_i,v_j)$ such that $v_k \notin p(u,v)$ in G. Then the edge (v_k,v_i) and the edge (v_k,v_j) must be included in T^*.*

Proof. Assume that edge (v_k,v_i) or edge (v_k,v_j) is excluded in T^*, since there do not exist any path $p(v_i,v_j)$ such that $v_k \notin p(v_i,v_j)$ in G, then there must not exist any path $p(v_i,v_j)$ in T^*, and then T^* is not a connected graph. So the edge (v_k,v_i) and the edge (v_k,v_j) must be included in T^*. □

We can use the dynamic transitive closure algorithm [15,16] to decide whether there exist any path $p(v_i,v_j)$ such that $v_k \notin p(v_i,v_j)$.

3.2. Reduction algorithm

Based on the above analysis, the reduction algorithm for DCMST may be sketched as:

Algorithm 1. Reduction_DCMST

Input: Graph $G=(V,E)$, weight function w and degree constraint b.
Output: $G=(V,E)$, T^*.
begin
1. Using Theorem 2 to remove all edges that satisfy the conditions.
2. Using Theorem 1 to remove all pendant vertices (vertices of degree 1) and the corresponding edges from G and add the corresponding edges to T^*.
3. Using Theorem 3 to remove all the edges that satisfy the conditions from $\boldsymbol{G}$ and add all the edges to T^*.

end

The worst case time complexity of step 1 is $O(n^2)$ where $n=|V|$, step 2 is $O(n)$, and step 3 is $O(n^3)$. So the worst case time complexity of the entire algorithm is $O(n^3)$.

3.3. Finding a degree-constrained tree

A tree is a subgraph of G that does not contain any circuits. As a result, there is exactly one path from each vertex in the tree to each other vertex in the tree. A spanning tree of a graph G is a tree containing all vertices of G. A minimum spanning tree (MST) of an undirected, weighted graph G is a spanning tree of which the sum of the edge weights (costs) is minimal.

There are several greedy algorithms for finding a minimal spanning tree T^* of a graph. The algorithms of Kruskal and Prim are well known of them.

Kruskal's algorithm. Repeat the following step until the set T^* has $n-1$ edges (initially T^* is empty). Add to T^* the shortest edge that does not form a circuit with edges already in T^*.

Prim's algorithm. Repeat the following step until the set T^* has $n-1$ edges (initially T^* is empty): Add to T^* the shortest edge between a vertex in T^* and a vertex not in T^* (initially pick any edge of shortest length).

Although both are greedy algorithms, they are different in the sense that Prim's algorithm grows a tree until it becomes the MST, whereas Kruskal's algorithm grows a forest of trees until the forest reduces to a single tree, the MST.

In this paper, we find a degree-constrained tree using an algorithm based on the classic Kruskal's algorithm to build a minimum spanning tree where it always keeps the degrees of all vertices satisfying all the degree constraints.

3.4. Edge exchange technique

Edge exchange technique was employed to improve a spanning tree of graph. Firstly, edge exchange deletes some edges from a tree, thus breaking the tree into two or more than two subtrees, and then reconnects those subtrees into one tree in a possible way [14,17]. Based on this idea, we use two edge exchange methods to improve the existing degree-constrained tree.

(1) 1-opt edge exchange operator

Definition 1. Given a spanning tree T of graph G and two edges $e\in T$ and $e^1\notin T$, the pair (e,e^1) gives an admissible exchange in T if $T-e+e^1$ is a spanning tree.
It can be seen that 1-opt edge exchange operation changes the degree of vertex in spanning tree.

(2) 2-opt edge exchange operator
Since 1-opt edge exchange operator changes the degree of vertex in spanning tree, the improved spanning tree by 1-opt exchange operator might not satisfy the degree constraint. The 2-opt edge exchange operation does not change degree of any vertex in spanning tree, so the improved spanning tree by 2-opt edge exchange operator will always satisfy the degree constraint.
$T=(V,E^T)$ is a spanning tree of graph $G=(V,E)$. Suppose that edge $(i,i+1)\in E^T$, edge $(j,j+1)\in E^T$, edge $(i,j)\in G$ and edge $(i,j)\notin E^T$, edge $(i+1,j+1)\in G$ and edge $(i+1,j+1)\notin E^T$, and $w(i,j)+w(i+1,j+1)<w(i,i+1)+\mathrm{w}(j,j+1)$. Then the edges $(i,i+1)$ and $(j,j+1)$ can be replaced by edges (i,j) and $(i+1,j+1)$.

3.5. The algorithm and the time complexity

From the above analysis, the main algorithm for DCMST may be sketched as:

Algorithm 2. Main_DCMST

Input: Graph $G=(V,E)$, weight function w and degree constraint b.
Output: $G^*=(V^*,E^*)$, G^* denote one degree-constrained minimum spanning tree of G.
begin
1. $V^*=V$; $E^*=\Phi$; Start with a graph $T^*=(V^*,E^*=\Phi)$ consisting of only the vertices of G and no edges; This can be viewed as n connected components, each vertex being one connected component.
2. Executing the sub-algorithm reduction_DCMST {Step 3 to step 6 find a solution of G.}
3. $E_1=E$;
4. Select the smallest cost edge $e_{\min}=(v_k,v_h)$ from E_1.
5. If the edge $e_{\min}$ connects two different connected components of T^*, $d^{T*}(v_k)<b_k$ and also $d^{T*}(v_h)<b_h$, then
   {$E^*=E^*+\mathrm{e}_{\min}$; $E_1=E_1-e_{\min}$;}
   else
   {$E_1=E_1-e_{\min}$; goto 4;}
6. if $|E^*|<n-1$ then
   goto 4;

7. Using edge exchange techniques to improve the solution.

end

The worst case time complexity of step 1 is $O(n)$ where $n = |V|$, step 2 is $O(n^3)$. The worst case time complexity of steps 3–6 of the algorithm is also $O(n)$. The worst case time complexity of 2-opt is $O(n^2)$ and complexity of 1-opt is $O(n)$. So the worst case time complexity of the entire algorithm is $O(n^3)$.

4. Experimental results and analysis

Example. Graph G and the weight of the edge are presented in Fig. 1. The degree constraint b is $b_e = b_f = 1$, $b_a = b_b = b_d = b_g = b_h = b_i = b_j = 2$, $b_c = 4$.

Analysis:

(1) $V = V^* = \{a,b,c,d,e,f,g,h,i,j\}$, $E^* = \Phi$, $E = \{(a,b), (b,c), (c,d), (c,e), (d,f), (d,g), (d,h), (e,f), (g,h), (g,i), (g,j), (i,j)\}$, $T^* = (V^*, E^*)$, $G = (V, E)$

(2) According to Theorem 2, we remove edge (e,f) from G, and then the graph G is shown in Fig. 2.

(3) According to Theorem 1, we remove the pendant vertices (vertices of degree 1) and corresponding edges from the graph G and add the edges to tree T^*, then the graph G and tree T^* are shown in Fig. 3.

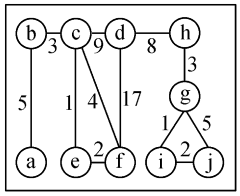

Fig. 1. Graph G.

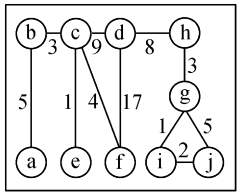

Fig. 2. Graph G after removing edge (e,f).

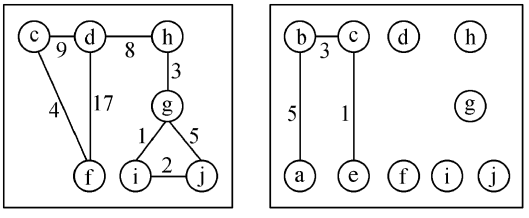

Fig. 3. Graph G (left) and tree T^* (right).

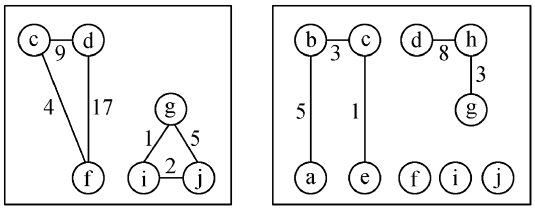

Fig. 4. Graph G (left) and tree T^* (right).

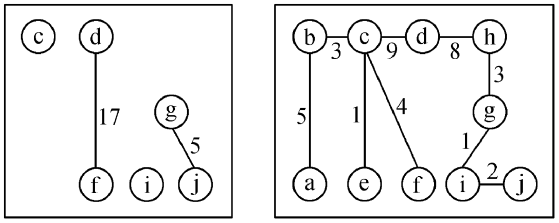

Fig. 5. Graph G (left) and tree T^* (right).

(4) According to Theorem 3, we remove edge (d,h) and edge (h,g) from the graph G and add them to tree T^*, then the graph G and tree T^* are shown in Fig. 4.

(5) Find a solution of G by adding edges in the following order:

a. Since edge (g,i) is the smallest edge in graph G and satisfies all the other conditions, so add the edge (g,i) to T^*. Then change E^* and E by executing the following operator:

$$E^* = E^* + (g,i), \quad E = E - (g,i)$$

b. Since edge (i,j) is the smallest edge in graph G and satisfies all the other conditions, so add the edge (i,j) to T^*. Then change E^* and E by executing the following operator:

$$E^* = E^* + (i,j), \quad E = E - (i,j)$$

c. Since edge (c,f) is the smallest edge in graph G and satisfies all the other conditions, so add the edge (c,f) to T^*. Then change E^* and E by executing the following operator:

$$E^* = E^* + (c,f), \quad E = E - (c,f)$$

d. Since edge (c,d) is the smallest edge in graph G and satisfies all the other conditions, so add the edge (c,d) to T^*. Then change E^* and E by executing the following operator:

$$E^* = E^* + (c,d), \quad E = E - (c,d)$$

Now, the graph G and tree T^* are shown in Fig. 5.

(6) Since no better solution can be achieved by edge exchange, so Fig. 5 is the final solution.

5. Conclusion

In this paper, we have proposed a new algorithm for DCMST. The algorithm employs the properties of the

problem to reduce it's size and hardness and the edge exchange technique to improve the existing solution.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 70471065), Shanghai Leading Academic Discipline Project (Grant No. T0502) and Shanghai Scientific Special Funds for Cultivation and Selection of Excellent Young Teaching Staffs of Higher Education (Grant No. 21012).

References

[1] Syslo MM, Deo N, Kowalik JS. Discrete optimization algorithms. Englewood Cliffs: Prentice-Hall; 1983.
[2] Graham R, Hell P. On the history of the minimum spanning tree problem. Ann Hist Comput 1985;7:43–57.
[3] Ribeiro CC, Souza MC. Variable neighborhood search for the degree-constrained minimum spanning tree problem. Discrete Appl Math 2002;118(1–2):43–54.
[4] Markus B, Michael J, Frauke L. A primal branch-and-cut algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of 6th international workshop on efficient and experimental algorithms, Rome, Italy; 2007.
[5] Garey MR, Johnson DS. Computers and intractability, a guide to the theory of NP-completeness. New York: W.H. Freeman and Company; 1979.
[6] Narula SC, Ho CA. Degree-constrained minimum spanning tree. Comput Oper Res 1980;7(4):239–49.
[7] Andrade R, Lucena A, Maculan N. Using Lagrangian dual information to generate degree constrained spanning trees. Discrete Appl Math 2006;154(5):703–17.
[8] Goemans MX. Minimum bounded-degree spanning trees. In: Proceedings of the 47th annual IEEE symposium on foundations of computer science; 2006. p. 273–82.
[9] Volgenant A. A Lagrangian approach to the degree-constrained minimum spanning tree problem. Eur J Oper Res 1989;39:325–31.
[10] Knowles JD, Corne DW. A new evolutionary approach to the degree-constrained minimum spanning tree problem. IEEE Trans Evol Comput 2000;4(2):125–34.
[11] Krishnamoorthy M, Ernst AT, Sharaiha YM. Comparison of algorithms for the degree constrained minimum spanning tree. J Heuristics 2001;7:587–611.
[12] Raidl GR. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of the 2000 congress on evolutionary computation; 2000. p. 104–11.
[13] Caccetta L, Hill SP. A branch and cut method for the degree-constrained minimum spanning tree problem. Networks 2001;37(2):74–83.
[14] Ning AB, Ma L, Xiong XH. Solving degree-constrained minimum spanning tree with a new algorithm. In: Proceedings of 2007 international conferences on managements science & engineering. Harbin, China, August 20–22; 2007. p. 381–6.
[15] Alberts D, Cattaneo G, Italiano G. An empirical study of dynamic graph algorithms. In: Proceedings of the 7th annual ACM-SIAM symposium on discrete algorithms. Atlanta, United States; 1992. p. 192–201.
[16] King V, Sagert G. A fully dynamic algorithm for maintaining the transitive closure. In: Proceedings of the 31st ACM symposium on theory of computing; 1999. p. 492–8.
[17] Lin S. Computer solutions of the traveling salesman problem. Bell Syst Tech J 1965;44:2245–69.