ANZIAM J. 58(2017), 333–341
doi:10.1017/S1446181117000074

COMPUTATIONAL METHODS FOR LOGISTICS PROBLEMS RELATED TO OPTIMAL TREES

LONGSHU WU[1], QIN WANG✉[1] and XIAOBING YANG[1]

(Received 14 July, 2016; accepted 15 October, 2016; first published online 7 March 2017)

Abstract

In recent years, balanced network optimization problems play an important role in practice, especially in information transmission, industry production and logistics management. In this paper, we consider some logistics optimization problems related to the optimal tree structures in a network. We show that the most optimal subtree problem is NP-hard by transforming the connected dominating set problem into this model. By constructing the network models of the most balanced spanning tree problem with edge set restrictions, and by finding the optimal subtrees in special networks, we present efficient computational methods for solving some logistics problems.

2010 *Mathematics subject classification*: 90C27.

Keywords and phrases: computational method, spanning tree, polynomial algorithm, nondeterministic polynomial-time (NP)-hard.

1. Introduction

Network optimization has attracted an increasing interest, and people have found many applications in the real world in recent years [1, 5]. Many logistics management problems are related to the optimal tree structures in networks [3, 6, 7, 9, 11]. Given a graph G, we call it a *forest* if it is acyclic, and we call it a *tree* if it is connected and acyclic. A component of G is a maximal connected subgraph of G. The spanning subgraph of G is a subgraph which contains all the vertices of G. If the spanning subgraph of G is a tree, we call it a *spanning tree* of G. We say that T is a *minimum spanning tree* of G if $W(T) = \sum_{e\in T} w(e)$ is minimum among all the spanning trees of G, where w is a weight vector defined on the edge set of G. The balance degree of a tree T is the difference of the maximum weight and the minimum weight on T, which can be expressed as $b(T) = \max\{w(e) - w(e') \mid e, e' \in T\}$.

We consider the problem of finding the most balanced spanning tree T of G such that the balance degree of T is minimum. Efficient polynomial-time algorithms for

[1]College of Sciences, China Jiliang University, Hangzhou, China; e-mail: wls@cjlu.edu.cn, wq@cjlu.edu.cn, yangxiaobing467977@163.com.

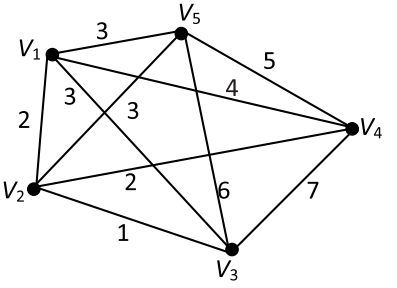

FIGURE 1. Example of the logistics problem.

solving the most balanced spanning tree problem in a network have been proposed [2, 4, 8, 10]; Wang et al. [10] proved the following lemma.

LEMMA 1.1 [10]. *Let $G = (V, E, w)$ be an acyclic connected undirected network with $|V| = n$ and $|E| = m$. Let w be a finite nonnegative vector such that $w(e_1) \le w(e_2) \le \cdots \le w(e_m)$. Denote $T' = \{T \mid T$ is the spanning tree of G which contains edge $e_1\}$ and suppose that $w(T_1) = \min\{w(T) \in T'\}$. Then we have $b(T) \ge b(T_1)$ for any $T \in T'$.*

This paper is organized as follows. In Section 2, we present efficient methods to solve the problem of choosing machine parts in a factory. We show that the most optimal subtree problem is NP-hard, and obtain the methods for finding the most optimal subtrees in some special networks in Section 3. Some extension problems and concluding remarks are given in Section 4.

2. Algorithms for logistics problems

In a factory, suppose that in order to satisfy the assembling requirement, we need to choose a batch of machine parts without exactly having the same properties. We write the detailed network optimization model by constructing a network as follows. Let each vertex of the network represent a property of the machine parts. Suppose that every machine part possesses two properties. Then we add an edge between the two vertices of the properties, and let the weight of this edge be the life span of the machine part. The factory would like to choose the machine parts with life spans as close as possible to satisfy the requirement for updating and replacing the product, such that the waste is small. Obviously, the machine parts should contain all the properties, and the amount of the parts should be as small as possible. Taking Figure 1 as an example, we present the detailed solution procedure and results of this problem.

However, in practice, the factory may have nepotism with some machine parts suppliers. As a result, the factory is supposed to choose some kinds of machine parts. For example, if the factory must choose the machine part e_{24} in the logistics problem of Figure 1, then we have Algorithm 2.

THEOREM 2.1. *The logistics problem with nepotism can be solved in polynomial time with computational complexity $O(|E|^2 \log |E|)$.*

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

Algorithm 1:

STEP 1. Sort the edges of G according to the weights such that $w(e_{23}) \le w(e_{12}) \le w(e_{24}) \le w(e_{13}) \le w(e_{15}) \le w(e_{25}) \le w(e_{14}) \le w(e_{45}) \le w(e_{35}) \le w(e_{34})$.

STEP 2. Use Kruskal's algorithm [5] to find the minimum spanning tree containing edge e_{23}. The tree is $\{e_{23}, e_{12}, e_{24}, e_{15}\}$, and the balance degree is 2. Continue to find the minimum spanning tree containing e_{12} in $G - e_{23}$. The tree is $\{e_{12}, e_{24}, e_{13}, e_{15}\}$, and the balance degree is 1. The minimum spanning tree containing e_{24} in $G - \{e_{23}, e_{12}\}$ is $\{e_{24}, e_{13}, e_{15}, e_{25}\}$, and the balance degree is 1. The minimum spanning tree containing e_{13} in $G - \{e_{23}, e_{12}, e_{24}\}$ is $\{e_{13}, e_{15}, e_{25}, e_{14}\}$, and the balance degree is 1. The minimum spanning tree containing e_{15} in $G - \{e_{23}, e_{12}, e_{24}, e_{13}\}$ is $\{e_{15}, e_{25}, e_{14}, e_{35}\}$, and the balance degree is 3. The minimum spanning tree containing e_{25} in $G - \{e_{23}, e_{12}, e_{24}, e_{13}, e_{15}\}$ is $\{e_{25}, e_{14}, e_{45}, e_{35}\}$, and the balance degree is 3. There is no spanning tree in $G - \{e_{23}, e_{12}, e_{24}, e_{13}, e_{15}, e_{25}\}$. Stop, go to STEP 3.

STEP 3. The trees $\{e_{12}, e_{24}, e_{13}, e_{15}\}$,$\{e_{24}, e_{13}, e_{15}, e_{25}\}$ and $\{e_{13}, e_{15}, e_{25}, e_{14}\}$ are the optimal spanning trees with the minimum balance degree.

Algorithm 2:

STEP 1. Sort the edges of $G - e_{24}$ according to the weights such that $w(e_{23}) \le w(e_{12}) \le w(e_{13}) \le w(e_{15}) \le w(e_{25}) \le w(e_{14}) \le w(e_{45}) \le w(e_{35}) \le w(e_{34})$.

STEP 2. Use Kruskal's algorithm [5] to find the minimum spanning tree containing edges e_{24} and e_{23}. The tree is $\{e_{24}, e_{23}, e_{12}, e_{15}\}$, and the balance degree is 2. Continue to find the minimum spanning tree containing e_{24} and e_{12} in $G - e_{23}$. The tree is $\{e_{24}, e_{12}, e_{13}, e_{15}\}$, and the balance degree is 1. The minimum spanning tree containing e_{24} and e_{13} in $G - \{e_{23}, e_{12}\}$ is $\{e_{24}, e_{13}, e_{15}, e_{25}\}$, and the balance degree is 1. The minimum spanning tree containing e_{24} and e_{15} in $G - \{e_{23}, e_{12}, e_{13}\}$ is $\{e_{24}, e_{15}, e_{25}, e_{35}\}$, and the balance degree is 4. The minimum spanning tree containing e_{24} and e_{25} in $G - \{e_{23}, e_{12}, e_{13}, e_{15}\}$ is $\{e_{24}, e_{25}, e_{14}, e_{35}\}$, and the balance degree is 4. The minimum spanning tree containing e_{24} and e_{14} in $G - \{e_{23}, e_{12}, e_{13}, e_{15}, e_{25}\}$ is $\{e_{24}, e_{14}, e_{45}, e_{35}\}$, and the balance degree is 4. There is no spanning tree in $G - \{e_{23}, e_{12}, e_{13}, e_{15}, e_{25}, e_{14}\}$. Stop, go to STEP 3.

STEP 3. The trees $\{e_{24}, e_{12}, e_{13}, e_{15}\}$ and $\{e_{24}, e_{13}, e_{15}, e_{25}\}$ are the optimal spanning trees containing edge e_{24} with minimum balance degree.

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

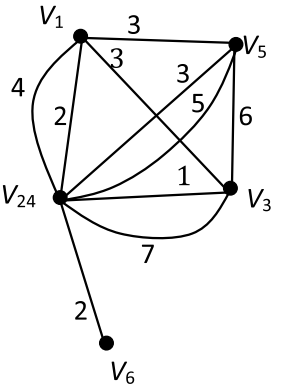

FIGURE 2. The logistics problem with nepotism.

PROOF. The correctness of Algorithm 1 can be directly deduced from Lemma 1.1, because the balance degree of the spanning tree obtained by Algorithm 1 is minimum among all possible candidate spanning trees.

In fact, Algorithm 2 is equivalent to Algorithm 1, when we transfer the logistics problem of Figure 1 to Figure 2. In Figure 2, we contract the edge e_{24} and its two end vertices v_2 and v_4 in Figure 1 to one vertex v_{24}. Now the edges e_{12} and e_{14}, the edges e_{25} and e_{45}, and the edges e_{23} and e_{34} turn into multiple edges between vertices v_1 and v_{24}, v_5 and v_{24}, and v_3 and v_{24}, respectively. Furthermore, we add a new vertex v_6 and a new edge between v_6 and v_{24} in Figure 2, and we set the weight of this edge to 2, which equals $w(e_{24})$ in Figure 1. After this transformation, we know that the edge between v_6 and v_{24} is contained in every spanning tree of Figure 2. Hence, we can use Algorithm 1 to find the optimal spanning trees with minimum balance degree in Figure 2.

In the case when the factory has an interest relationship with more than one machine parts supplier, the factory must purchase many kinds of machine parts. We can use the above method to contract the given machine parts and obtain a new figure; then we can use Algorithm 1 to solve the logistics problem with nepotism.

We see that both Algorithm 1 and Algorithm 2 have the same computational complexities. Here, we only analyse the time complexity of Algorithm 1. The running time used in STEP 1 for sorting the weights of the edges of G is $O(|E|\log|E|)$. Finding the minimum spanning tree in STEP 2 takes $O(|E|\log|E|)$ time. Kruskal's algorithm is used at most $O(|E|)$ times, so the time complexity is given by $O(|E|^2\log|E|)$. □

On the other hand, we can also consider the most unbalanced spanning tree problems. The problem is to find a spanning tree T in a graph G such that the balance degree of T is as large as possible. Since any two edges which are not multiple edges of G will not form a cycle, the problem can be solved by finding two edges e and f with maximum difference, which are not multiple edges of G. Note that any spanning tree containing e and f will be a most unbalanced spanning tree of G.

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

Now we consider the most unbalanced spanning tree problem with edge restrictions. Let F be a forest of G. This problem is to find a spanning tree T^* of G such that (a) T^* contains F and (b) the balance degree of T^* is as large as possible. We now present an efficient algorithm to solve this problem.

Algorithm 3:

Step 1. Contract each component of the given forest F into a single vertex and denote the resulting graph by G_1.

Step 2. Find the most unbalanced spanning tree T_1 in G_1.

Step 3. Denote $U(F) = \max_{e\in F} w(e)$ and $L(F) = \min_{e\in F} w(e)$. Denote $U(T_1) = \max_{e\in T_1} w(e)$ and $L(T_1) = \min_{e\in T_1} w(e)$. The optimal solution is $T^* = T_1 \cup F$, and the optimal balance degree of T^* is $\max\{U(F), U(T_1)\} - \min\{L(F), L(T_1)\}$.

Theorem 2.2. *The most unbalanced spanning tree problem with edge restrictions can be solved in polynomial time with computational complexity $O(|E|)$.*

Proof. First, since T_1 is a spanning tree of G_1 by contracting F of G, and $T^* = T_1 \cup F$, we know that T^* is connected and acyclic. So T^* is a spanning tree. Because the problem is to find a most unbalanced spanning tree containing F, and T_1 is a most unbalanced spanning tree of G_1, we have T^* as the optimal solution with the maximum balance degree.

As for the computational complexity, the running time used in Step 1 for contracting F from G is $O(|E|)$. Searching the maximum or minimum element in a number set takes $O(|E|)$ time. Therefore, the running times used in Step 2 and Step 3 are also $O(|E|)$. So the time complexity of Algorithm 3 is given by $O(|E|)$. □

3. The most optimal subtrees

In real life, sometimes we need to set up an operation centre in a logistics system in order to control other areas. We use the vertices in a graph to represent each area, and we use the edges in the graph to represent the roads among the areas. An operation centre must be simple and connected, and other areas should be able to have direct connections with this operation centre.

We transfer this problem to a mathematical model, which is to find a minimum-weight subtree in a network such that the vertices not belonging to the subtree can be adjacent to some vertex of the subtree. We will show that the most optimal subtree problem is NP-hard, even when the weight of each edge is one unit. We will use the NP-hard connected dominating set problem [5] for the reduction.

For a simple graph G, a vertex subset $X \subseteq V(G)$ is called a connected dominating set of G if the induced subgraph $G[X]$ is connected and every vertex in G either belongs

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

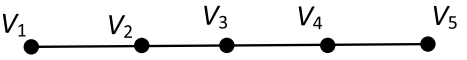

FIGURE 3. Chain network.

to X or is adjacent to a vertex in X. The decision version of the connected dominating set problem can be described as follows: for a given simple graph G and a positive integer r with $r \leq |V(G)|$, is there a connected dominating set X of G such that $|X| = r$?

THEOREM 3.1. *The most optimal subtree problem is NP-hard.*

PROOF. Let (G, r) be an instance of the connected dominating set problem, where G is a simple graph and r is a positive integer with $r \leq |V(G)|$. We construct an instance of the decision version of the most optimal subtree problem as follows. The graph in consideration is still G. Define the weight $w(e) = 1$ for each edge $e \in E(G)$. The threshold is defined by $Y = r - 1$. The decision version of the most optimal subtree problem asks whether there is a subtree F of G such that the vertices not belonging to F can be adjacent to some vertex of F and $w(F) \leq Y$. Note that this is equivalent to asking whether there is a subtree F of G such that the vertices not belonging to F can be adjacent to some vertex of F and $|E(F)| \leq Y$.

First, it is clear that this decision problem is in NP. Also, observe that the above contribution can be done in polynomial time. In the following, we show that the instance of the connected dominating set problem has a solution if and only if there is a subtree F of G such that the vertices not belonging to F can be adjacent to some vertex of F and $|E(F)| \leq Y$.

Suppose first that the instance of the connected dominating set problem has a solution. Then there is a connected dominating set X of G such that $|X| = r$. Define F to be a spanning tree of $G[X]$. Then $|E(F)| = r - 1 = Y$, and the vertices not belonging to F can be adjacent to some vertex of F, as required.

Conversely, suppose that there is a subtree F of G such that the vertices not belonging to F can be adjacent to some vertex of F and $|E(F)| \leq Y$. Then $|V(F)| = |E(F)| + 1 \leq r$ is a connected dominating set of G. We conclude that any vertex set X of G with $|X| = r$ and $V(F) \subseteq X$ is a solution of the instance of the connected dominating set problem; hence, the result follows. □

Since this problem is NP-hard, we analyse the following four kinds of special graphs, and we present the methods to find the minimum-weight subtrees in these networks. First, we consider the chain network as in Figure 3.

In this network, there is only one spanning tree. By deleting the two end points and the two edges incident to them in this path, we obtain the minimum-weight subtree of this chain network. Next, we consider the ladder graph as in Figure 4.

In this network, there are five possible subtrees, which are the path 2-5, the path 2-1-4-5, the path 1-2-3, the path 4-5-6 and the path 2-3-6-5. By comparing with the weights of these five subtrees, we obtain the minimum-weight subtree of this ladder network. When the ladder network is enlarged, we can deal with it by a similar method,

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

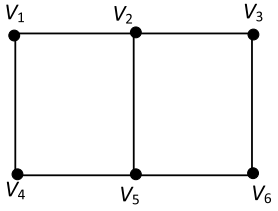

FIGURE 4. Ladder network.

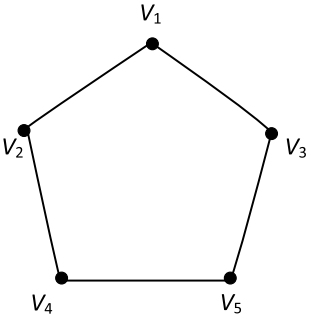

FIGURE 5. Cyclic network.

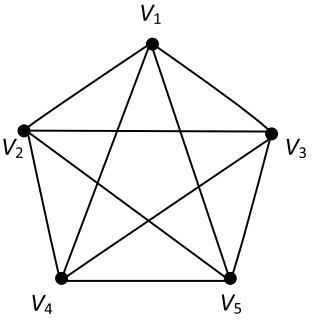

FIGURE 6. Complete graph.

while the situation of the possible subtrees may become more complicated. We discuss the cyclic graph in Figure 5.

In this network, there is only one cycle. We can find the minimum-weight path with $n-3$ edges. We can also find the maximum-weight path with three edges, and delete the three edges and the two intermediate vertices in this path.

Finally, we consider the complete graph as in Figure 6. In the complete graph, because each vertex is adjacent to every other vertex, any vertex of this graph can be a minimum-weight subtree. Because there is no edge in this case, the weight of the subtree is zero, which is also the weight of any vertex of this complete graph.

https://doi.org/10.1017/S1446181117000074 Published online by Cambridge University Press

4. Concluding remarks

In this paper, we considered some logistics optimization problems related to optimal tree structures in a network. We designed computational methods for solving these logistics problems. By constructing the network models of the most balanced spanning tree problem with edge set restrictions, we presented efficient polynomial-time methods to solve the problem of choosing machine parts in a factory. The computational complexities of these methods are $O(|E|^2 \log |E|)$. We also presented an efficient algorithm with time complexity $O(|E|)$ to solve the most unbalanced spanning tree problem with edge restrictions.

We proved that the most optimal subtree problem is NP-hard by transforming the connected dominating set problem into this model. And, by finding the optimal subtrees in four kinds of special networks, we presented methods for solving the most optimal subtrees in these special networks.

Due to the practical potential of the most balanced optimization, it may be interesting to explore whether other relevant most balanced problems are solvable in polynomial time. It is also meaningful to consider the inverse problem of the most balanced optimization.

For the most optimal subtree problems, except for the chain network, the ladder network, the cyclic network and the complete graph, there exist many other kinds of special networks in practice. Because of the special structures of different networks, we can find their respective minimum-weight subtrees. For example, in an enlarged ladder network, the situation of the possible subtrees may become more complicated. In this case, it is challenging to find all the possible subtrees of a network. From this point of view, it is also meaningful to analyse the properties of special networks to obtain the optimal conditions of the most optimal subtrees.

Acknowledgement

This research is supported by the National Natural Science Foundation of China under Grant No. 11171316.

References

[1] C. Buchheim and L. Klein, "Combinatorial optimization with one quadratic term: spanning trees and forests", *Discrete Appl. Math.* **177** (2014) 34–52; doi:10.1016/j.dam.2014.05.031.

[2] P. M. Camerini, F. Maffioli, S. Martello and P. Toth, "Most and least uniform spanning trees", *Discrete Appl. Math.* **15** (1986) 181–197; doi:10.1016/0166-218X(86)90041-7.

[3] H. Charkhgard and M. Savelsbergh, "Efficient algorithms for traveling salesman problems arising in warehouse order picking", *ANZIAM J.* **57** (2015) 166–174; doi:10.1017/S1446181115000140.

[4] Z. Galil and B. Schieber, "On finding most uniform spanning trees", *Discrete Appl. Math.* **20** (1988) 173–175; doi:10.1016/0166-218X(88)90062-5.

[5] B. Korte and J. Vygen, *Combinatorial optimization, theory and algorithms*, 4th edn (Springer, Heidelberg, 2007).

[6] S. Kundu and S. Majumder, "A linear time algorithm for optimal k-hop dominating set of a tree", *Inform. Process. Lett.* **116** (2016) 197–202; doi:10.1016/j.ipl.2015.07.014.

[7] M. K. Murmu, "A distributed approach to construct minimum spanning tree in cognitive radio networks", *Procedia Comput. Sci.* **70** (2015) 166–173; doi:/10.1016/j.procs.2015.10.066.
[8] A. P. Punnen and K. P. K. Nair, "Constrained balanced optimization problems", *Comput. Math. Appl.* **37** (1999) 157–163; doi:10.1016/S0898-1221(99)00119-4.
[9] N. Shiono and H. Suzuki, "Optimal pipe-sizing problem of tree-shaped gas distribution networks", *European J. Oper. Res.* **252** (2016) 550–560; doi:10.1016/j.ejor.2016.01.008.
[10] F. Wang, Z. Xie and Z. J. Liang, "Minimal balanced degree spanning tree with edge set restricted problem", *Math. Theory Appl.* **24** (2004) 100–103; http://en.cnki.com.cn/Article_en/CJFDTotal-LLYY200402022.htm.
[11] Q. Wang, J. J. Yuan and J. Z. Zhang, "An inverse model for the most uniform problem", *Oper. Res. Lett.* **36** (2008) 26–30; doi:10.1016/j.or1.2007.03.006.