An Effective Construction Algorithm for the Steiner Tree Problem Based on Edge Betweenness

Misa Fujita¹, Takayuki Kimura² and Kenya Jin’no²

¹ Graduate School of Electronics, Information and Media Engineering, Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-machi, Minami-Saitama, Saitama 345-8501, Japan
² Department of Electrical and Electronic Engineering, Nippon Institute of Technology
4-1 Gakuendai, Miyashiro-machi, Minami-Saitama, Saitama 345-8501, Japan
E-mail: ¹e1112407@estu.nit.ac.jp, ²tkimura, jinno)@nit.ac.jp

Abstract

Given an undirected weighted graph \( G = (V, E, c) \) and a set \( T \), where \( V \) is the set of nodes, \( E \) is the set of edges, \( c \) is a cost function, and \( T \) is a subset of nodes called terminals, the Steiner tree problem in graphs is that of finding the subgraph of the minimum weight that connects all of terminals.

The Steiner tree problem is an example of an \( \mathcal{NP} \)-complete combinatorial optimization problem [1]. Thus, approximate methods are usually employed for constructing the Steiner tree. In this study, the KMB algorithm [2], which is an efficient construction method for Steiner tree problems, is enhanced by considering edge betweenness [3]. The results of numerical simulations indicate that our improved KMB algorithm shows good performances for various types of benchmark Steiner tree problems.

1. Introduction

Given an undirected weighted graph \( G = (V, E, c) \) and a set \( T \), where \( V \) is the set of nodes, \( E \) is the set of edges, \( c \) is a non-negative cost function that measures the lengths of the edges, and \( T \) is a subset of nodes called terminals, the Steiner tree is a subgraph that connects all of the terminals.

The tree must be connected
The tree must contain all of the terminals

Various patterns of Steiner trees can be constructed from a network. Some examples of Steiner trees are presented in Fig. 1. In Fig. 1, colored nodes indicate terminals and white nodes indicate non-terminals. Figure 1(a) and (b) have the same costs. However, the topologies are different from each other. Figure 1(c) is the minimum cost Steiner tree in this network.

![Figure 1: Examples of the Steiner trees](image)

The Steiner trees are employed in various real-world applications, such as VLSI routing [4], wirelength estimation [5], and network routing [6]. In order to determine a Steiner tree using a numerical model, we first define the decision variable of an edge, \( x(e_i) \), as follows:

\[
x(e_i) = \begin{cases} 
1 & (e_i \in E_{ST}) \\
0 & (\text{otherwise}) 
\end{cases}
\]

In Eq. (1), \( e_i \) is the \( i \)th edge in \( E \), and \( E_{ST} \) is a subset of edges used in the Steiner tree. If \( e_i \) is included in the Steiner tree, then the decision variable \( x(e_i) \) takes a value of 1, otherwise, the decision variable \( x(e_i) \) takes a value of 0.

Next, we define the objective function of the Steiner tree as follows:

\[
\min_{i=1}^{\left| E \right|} c(e_i)x(e_i)
\]

where \( \left| E \right| \) is the number of edges, and \( c(e_i) \) is the cost of the \( i \)th edge in the network. If the edge \( e_i \) is included in the Steiner tree, the cost of the edge, \( c(e_i) \), is added to the Steiner tree. In the Steiner tree problem, the following constraints are imposed.

- The tree must be connected
- The tree must contain all of the terminals

The Steiner tree problem is an example of an \( \mathcal{NP} \)-complete combinatorial optimization problem [1]. The Dreyfus-Wagner algorithm [7] was first proposed to obtain an exact optimum solution. However, the cost of the calculation to obtain an optimum solution using this algorithm becomes \( O(|V|^3/2 + |V|^2(2^{|T|}-1 - |T|-1) + |V|(3^{|T|}-1 - 2^{|T|}+3)/2) \), where \( |V| \) is the number of nodes and \( |T| \) is the number of terminals. \( |\cdot| \) indicates the number of elements. Thus, a huge calculation time is necessary if \( n \) and \( t \) are large.

Alternatively, approximate methods can be applied to construct Steiner trees. Various types of the approximate algorithms for the Steiner tree problems have previously been proposed [8–10]. Among these, the KMB algorithm [2] is an effective method for obtaining shorter Steiner trees. However, the KMB algorithm obtains various patterns of Steiner trees, and sometimes a Steiner tree is obtained at a very high cost. Thus, the KMB algorithm should be modified to ensure that low cost Steiner trees are stably obtained, by using other information in a network.
With this motivation, the KMB algorithm is improved in this paper by using the edge betweenness. In the results of numerical simulations, our improved KMB algorithm achieves a stable performance for various types of benchmark Steiner tree problems.

2. KMB Algorithm

To obtain a Steiner tree using the KMB algorithm, we first construct the complete graph \( G_1 = (T, E_1, c_1) \) from \( G \) and \( T \) (Fig. 2(b)). In Fig. 2, the colored nodes express terminals, and the white nodes express non-terminals. In the complete graph \( G_1 \), all of the nodes are terminals. In addition, the cost of the edge between the \( i \) th node and the \( j \) th node, \( c(v_i, v_j) \), \((i,j = 1, \ldots, |T|, i \neq j)\), is set to the cost of the shortest path between these nodes. Next, we construct the minimum spanning tree \( T_{\text{Tree1}} \) from the complete graph \( G_1 \) (Fig. 2(c)). Finally, the paths of \( T_{\text{Tree1}} \) are replaced by the shortest paths on the original network, generating the Steiner tree (Fig. 2(d)). If the Steiner tree has branches that contain non-terminals, then these nodes are removed from the Steiner tree.

The KMB algorithm uses the shortest path algorithm and the minimum spanning tree algorithm to construct the Steiner tree. In this paper, we used Dijkstra’s algorithm [11] as the shortest path construction method, and Prim’s algorithm [12] as the minimum spanning tree construction method. The calculation cost of the shortest path algorithm is \( O(|V|^2) \). In addition, the calculation cost of the minimum spanning tree algorithm is \( O(|V|^2) \). In total, the KMB algorithm requires a running time of \( O(|T||V|^2) \) to construct the Steiner tree.

3. The KMB Algorithm Using Edge Betweenness

In the KMB algorithm, if an edge is shared by multiple paths in the Steiner tree, then the Steiner tree takes small cost. Thus, edge should be shared by multiple paths as often as possible in order to construct a shorter Steiner tree. To achieve this, we introduce edge betweenness to the original KMB algorithm. Edge betweenness represents the centrality of the edges in the network. If an edge is used by many paths between nodes, then that edge has high betweenness.

The edge betweenness of the edge \( e \) is defined as follows:

\[
 eb(e) = \frac{\sum_{s=1}^{|V|} \sum_{g=1}^{|V|-1} I(s,g)}{(|V|-1)(|V|-2)/2} \tag{3}
\]

where \(|V|\) is the number of nodes, \( s \) is a starting node of the shortest paths, \( g \) is a terminating node of the shortest paths, \( I(s,g) \) is the number of shortest paths between \( s \) and \( g \) through the edge \( e \), and \( n_i(s,g) \) is the number of shortest paths between \( s \) and \( g \).

In Fig. 3(a), the colored nodes represent terminals, and the white nodes are non-terminals. The edge betweenness of each edge is indicated in Fig. 3(b).

By using the edge betweenness, we define a new cost for the edge as follows:

\[
c_{\text{new}}(e) = c(e) - \alpha \times eb(e) \tag{4}
\]

In Eq. (4), \( c(e) \) is the given cost of the edge \( e \), \( eb(e) \) is the edge betweenness of the edge \( e \), and \( \alpha \) is a controlling parameter that determines the priority between cost and betweenness of the given edge. Using the newly defined edge cost, an edge is likely to be included in the Steiner tree if it has a low cost and high edge betweenness.

To obtain a Steiner tree using the propose method, we first construct a complete graph \( G'_1 \) from a given network using Eq. (4), as the same manner as in the original KMB algorithm (Fig. 4(c)). Next, we construct the minimum spanning tree \( T_{\text{Tree1}}' \) from \( G'_1 \) (Fig. 4(d)), and the paths in \( T_{\text{Tree1}}' \) are replaced by those from original network to construct the Steiner tree (Fig. 4(e)). If the Steiner tree has branches that contain only non-terminals, then these nodes are removed from the Steiner tree. Finally, the costs of the edges in the Steiner tree are replaced by the original costs (Fig. 4(f)).
An example of a Steiner tree constructed using the proposed method is presented in Fig. 4.

Figure 4: Construction of a Steiner tree using our proposed method

4. Numerical Experiments

To compare the performances of the conventional KMB algorithm and our proposed method, we used the benchmark problems provided in SteinLib [13].

In our proposed method, the controlling parameter $\alpha$, the optimum value of which depends on the topological features of the considered network, determines the priority between the edge cost and edge betweenness. Furthermore, we consider the maximum edge cost to be an important factor in determining the optimum value of $\alpha$. From this perspective, we set $\alpha$ to 30% of the maximum edge cost for each network in these simulations.

Tables 1, 2, and 3 present the error rates obtained by the conventional KMB algorithm and the proposed method for the test sets B, C, and D in SteinLib [13]. It is seen from these tables that the proposed method achieved a better performance than the conventional KMB algorithm in the maximum and median values. These results indicate that multiple edges are shared by the paths on the Steiner tree obtained using the newly defined cost that is calculated from cost and betweenness of a given edge.
5. Conclusions

In this study, we have enhanced the KMB algorithm for the Steiner tree problem by considering edge betweenness. We evaluated the performance of our proposed method using the test sets provided in SteinLib [13]. The results of the numerical simulations indicate that the proposed method achieves a stable performance compared with the conventional KMB algorithm. In particular, the proposed method exhibited a good performance in the maximum and the middle values in almost all the instances. The conventional KMB algorithm often obtains longer Steiner trees, because the algorithm does not consider the overlaps of edges on the Steiner tree. On the other hand, our proposed method employs a new edge cost based on edge betweenness. In this case, the proposed method is able to obtain different and shorter trees compared with those obtained using the conventional KMB algorithm.

In future work, we will consider proposing additional parameter setting methods.

References


