Finding a Low-Diameter and Low-Weight k-Connected Subgraph

Weifa Liang[†] George Havas[‡] Anne Street[§]

† Department of Computer Science
The Australian National University
Canberra, ACT 0200, Australia
‡ Centre for Discrete Mathematics and Computing
Department of Computer Science and Electrical Engineering
The University of Queensland, QLD 4072, Australia
§ Centre for Discrete Mathematics and Computing
Department of Mathematics
The University of Queensland, Queensland 4072, Australia

Abstract

Low-diameter, low-weight subgraphs have useful applications in networks. The problem is how to find them. Given a weighted undirected k-connected graph $G = (V, E)$, let $\mathrm{diam}(G)$ *denote the diameter of G and let $w(G)$ be the sum of the weights of the edges in G. In this context, the problem is to find a k-connected spanning subgraph $G' = (V, E')$ of G such that the diameter and the weight of G' are minimized simultaneously. Since this problem is NP-hard, it is unlikely that there is a polynomial algorithm for it. We present an approximate solution with a performance guarantee. That is, we find a k-connected spanning subgraph G' in polynomial time such that* $\mathrm{diam}(G') \leq \alpha\mathrm{diam}(G)$ *and $w(G') \leq \beta w(G^*)$ for any fixed $\alpha > 1$, where G^* is a minimum k-connected spanning subgraph of G and β is a constant depending on α and k.*

Keywords: approximation algorithm, combinatorial optimization, k-connectivity, NP-hard problem.

1 Introduction

Let $G(V, E)$ be a weighted undirected graph with $|V| = n$ and $|E| = m$. A graph is *k-edge connected* (*k-vertex connected*) if there are k edge-disjoint (vertex-disjoint) paths between each pair of vertices or, equivalently, no two vertices can be separated by removing fewer than k edges (vertices). We use "G is k-connected" to mean G is either k-edge connected or k-vertex connected, depending on the context. In this paper we assume that G is k-connected, and the problem is to find a k-connected spanning subgraph G^* of G that simultaneously minimizes the diameter and the weight of G^*. This problem is NP-hard in a well-defined sense [14, §4.2].

Therefore, we focus on finding an approximation solution instead. (For brevity we may say 'minimize' for 'approximately minimize' when the meaning is clear.) That is, for any given $\alpha > 1$, we will find a k-connected spanning subgraph G' such that $\text{diam}(G') \leq \alpha \text{diam}(G)$ and $w(G') \leq \beta w(G^*)$ where $\text{diam}(G)$ is the diameter of G, $w(G)$ is the sum of the weights of the edges of G, G^* is a k-connected spanning subgraph of G with minimum $w(G^*)$, and $\beta > 1$ is a constant depending on α. Our general algorithm has many applications in communication networks, VLSI design, and distributed computing, for example.

The special case $k = 1$ has been extensively studied. Awerbuch et al [2] apply it to global routing design in distributed networks, giving an approximation solution. For any fixed $\alpha > 1$, their algorithm finds a spanning tree whose diameter is no greater than $\alpha \text{diam}(G)$ and whose weight is no greater than $\beta w(T_{mst})$, where $w(T_{mst})$ is the weight of a minimum spanning tree (MST) of G. Cong et al [4] study a variant of this problem (instead of minimizing diameter, they minimize radius) and they provide a better trade-off between α and β. Recently Khuller et al [14] presented the best possible trade-off between α and β so far, where $\beta \geq 1 + \frac{2}{\alpha - 1}$. Their sequential algorithm runs in $O(n)$ time given the shortest tree rooted at r, T_r, and an MST of G, T_{mst}, and in $O(m + n \log n)$ time otherwise, where $\text{diam}(T_r) = \text{diam}(G)$. The parallel implementation of their algorithm requires $O(\log n)$ time and $O(n)$ processors if both T_r and T_{mst} are given.

A closely related problem is to find a *minimum k-connected spanning subgraph* G^* of G where G^* has minimum weight. This problem is NP-hard for $k \geq 2$ [10, GT31]. There are a number of recent studies of this problem [1, 5, 13, 17, 22, 23]. For our problem with $k \geq 2$, the only paper which deals with simultaneously minimizing the diameter and the weight of a k-connected spanning subgraph is by Obraczka and Danzig [21]. Motivated by data replica inconsistency on the Internet, they formalized their problem as finding 2- and 3-vertex connected spanning subgraphs which simultaneously minimize the diameter and the weight of the subgraph. Using simulated annealing, they carried out a series of experiments to demonstrate that the performance of their algorithm is good in general. Unfortunately, their algorithm basically is ad hoc, and they do not provide any performance guarantees, compared with an optimal solution.

Thus motivated, we consider a general k-connected spanning subgraph problem which simultaneously minimizes the diameter and the weight of the subgraph. We present approximation algorithms for it which have performance guarantees. For a weighted undirected k-connected graph G, our general algorithm delivers an approximation solution whose diameter is no greater than $\alpha \mathrm{diam}(G)$ and whose weight is no greater than $(1+\frac{2}{\alpha-1})c(k)G^*$ for any given $\alpha > 1$. Here $c(k)$ is the factor to find a k-connected spanning subgraph G'' in polynomial time such that $w(G'')$ is $c(k)$ times the optimum. In particular, for the k-edge connected case, the best result for $c(k)$ so far is 2 [17]; and for the k-vertex connected case, the best result so far for $c(k)$ is $2\sum_{i=1}^{k} 1/i = 2H(k)$ [23] if $k > 5$, $c(2) = c(3) = 2$, and $c(4) = c(5) = 3$ otherwise [1, 5].

2 Notations and Basic Definitions

Let $G = (V, E)$ be a weighted undirected graph. A *minimum spanning tree* of G is a spanning tree of G of minimum total edge weight. The *distance* $D_G(u, v)$ between vertices u and v in G is the minimum weight of any path in G between them. Let $D_G(u) = \max\{D_G(u, v) \mid v \in V\}$. Then the *diameter*, denoted by $\mathrm{diam}(G)$, of G is $\max\{D_G(u) \mid u \in V\}$. A *shortest-path tree* of G, rooted at a vertex r, is a spanning tree such that, for any vertex v, the distance between r and v is the same as in G. A vertex $v \in V$ in G is *an articulation point* if deletion of v leaves G disconnected. Similarly, an edge $e \in E$ in G is a *bridge* if deletion of e leaves G disconnected.

A *branching* of a weighted directed graph rooted at r is a directed spanning tree of the graph such that each vertex except r has in-degree exactly one and r has in-degree zero. The *weight* of a branching is the sum of the weights of the edges in the branching. An *optimal branching* is a branching with minimum weight.

We deal with the problem for finding a spanning tree which minimizes both the distance between the other vertices and the root and the weight of it. A minimum spanning tree is a connected subgraph with minimum weight, whereas a shortest-path tree minimizes distances from the root. Since finding a spanning tree that simultaneously minimizes both the weight and the distance is NP-hard, there are some studies on balancing these two parameters [4, 14]. In particular, Khuller et al [14] gave the following *Light Approximate Shortest-path Tree* (LAST) definition.

Definition 1 *For $\alpha \geq 1$ and $\beta \geq 1$, a spanning tree T of G meeting the following two requirements is called an (α, β)-LAST rooted at r.*
1. *(Distance) The distance between r and v in T is at most α times the distance from r to v in G.*

2. *(Weight) The weight of T is at most β times the weight of a minimum spanning tree of G.*

We extend the LAST definition to the *Low Diameter, Low Weight k-connected Spanning Subgraph* (LDLWSS$_k$) definition.

Definition 2 *Let G be a weighted undirected k-connected graph. For $\alpha \geq 1$ and $\beta \geq 1$, a k-connected spanning subgraph G' of G meeting the following two requirements is called an (α, β)-LDLWSS$_k$.*
1. *(Diameter) The diameter of G' is at most α times the diameter of G.*
2. *(Weight) The weight of G' is at most β times the weight of a minimum k-connected spanning subgraph of G.*

The (α, β)-LAST rooted at r is a special case of an (α, β)-LDLWSS$_k$ when $k = 1$ and $\mathrm{diam}(T_r) = \mathrm{diam}(G)$. For this special case, Khuller et al [14] prove the following;

Theorem 1 *Let G be a graph with nonnegative edge weights; let r be a vertex of G; let $\alpha > 1$ and $\beta \geq 1 + \frac{2}{\alpha-1}$. Then G contains an (α, β)-LAST rooted at r. The LAST can be computed in linear time given a minimum spanning tree T_{mst} of G and a shortest-path tree T_r, and in $O(m + n\log n)$ time otherwise. Such an (α, β)-LAST rooted at r can also be found in $O(\log n)$ time using $O(n)$ processors on a CREW PRAM if T_{mst} and T_r are given.*

The basic idea of their algorithm is to traverse the minimum spanning tree, maintaining a current tree, and check each vertex when it is encountered to ensure that the distance requirement for that vertex is met in the current tree. If it is not met, the edges of the shortest path between the vertex and the root are added into the current tree. Other edges are discarded so that a tree structure is maintained.

The same method as in Theorem 1 gives the following corollary.

Corollary 1 *Let G be a graph with nonnegative edge weights; let T_r be a shortest-path tree with $\mathrm{diam}(T_r) = \mathrm{diam}(G)$, and let T be a spanning tree of G; let $\alpha > 1$ and $\beta \geq 1 + \frac{2}{\alpha-1}$. Then a spanning tree T_A with $\mathrm{diam}(T_A) \leq \alpha \mathrm{diam}(G)$ and $w(T_A) \leq (1 + \frac{2}{\alpha-1})w(T)$ can be computed in linear time. Such a spanning tree can also be found in $O(\log n)$ time using $O(n)$ processors on a CREW PRAM.*

3 Finding a k-Connected Spanning Subgraph

In this section we present a simple algorithm to find an (α, β)-LDLWSS$_k$ in a weighted undirected k-connected graph $G(V, E)$. Suppose $\alpha > 1$ is given. The algorithm proceeds as follows. First, determine the diameter of

G, $\text{diam}(G)$. This can be done by generating n shortest-path trees rooted at each vertex $v \in V$ and finding the maximum one from all $D_G(u)$ for $u \in V$. Let r be such a vertex and T_r be the shortest-path tree rooted at r, so $\text{diam}(T_r) = \text{diam}(G)$. Then, find an approximate minimum k-connected spanning subgraph G'' for G in polynomial time. Third, find an MST of G'', denoted by T. Fourth, construct an (α, β)-LAST, T_A, of G'' rooted at r, by applying the algorithm of Khuller et al [14]. That is, modify T to make it become T_A by depth-first search on T, using the edges in T_r and T. Finally the graph $G' = T_A \cup G''$ is obtained. Obviously G' is k-connected because G'' is k-connected and G'' is a subgraph of G'.

Theorem 2 *Given a weighted undirected k-connected graph $G(V, E)$ with diameter $\text{diam}(G)$, for any fixed $\alpha > 1$, an (α, β)-LDLWSS$_k$ of G can be found in polynomial time, where $\beta = (1 + \frac{2}{(\alpha-1)\lfloor k/2 \rfloor})c(k))$ and $c(k)$ is defined as follows. Let G'' and G^* be an approximate minimum k-connected spanning subgraph and a minimum k-connected spanning subgraph of G respectively, $w(G'') \le c(k)w(G^*)$ and $c(k) > 1$.*

Proof Since finding G^* is NP-hard for any $k > 1$, we find an approximate solution G'' for it instead, which is $c(k)$ times the optimum, so that

$$w(G'') \quad \le \quad c(k)w(G^*). \tag{1}$$

Then, find a minimum spanning tree T of G'' whose weight satisfies

$$\begin{aligned} w(T) &\le w(G'')/\lfloor k/2 \rfloor \\ &\le c(k)w(G^*)/\lfloor k/2 \rfloor. \end{aligned} \tag{2}$$

This can be done because there always exist at least $\lfloor k/2 \rfloor$ edge-disjoint spanning trees in a k-connected undirected graph [25, p. 353], and the weight of T is the minimum among those spanning trees by its definition. By Corollary 1,

$$w(T_A) - w(T_A \cap T) \quad \le \quad \frac{2w(T)}{\alpha - 1}. \tag{3}$$

Having inequalities 1, 2 and 3, we have

$$\begin{aligned} w(G') &= w(T_A \cup G'') \\ &\le w(T_A) - w(T_A \cap T) + w(G'') \\ &\le \frac{2}{(\alpha-1)} w(T) + w(G'') \\ &\le (1 + \frac{2}{(\alpha-1)\lfloor k/2 \rfloor}) c(k) w(G^*), \end{aligned} \tag{4}$$

and

$$\begin{aligned}\text{diam}(G') &\le \text{diam}(T_A) \\ &\le \alpha\text{diam}(G) \qquad (5)\end{aligned}$$

because T_A is a subgraph of G', and $\text{diam}(T_A) \le \alpha\text{diam}(G)$ by Corollary 1. $\square$

From Theorem 2, the following corollaries can be derived easily.

Corollary 2 *Let a weighted undirected graph $G(V, E)$ be k-edge connected. Then, for any fixed $\alpha > 1$, a k-edge connected spanning subgraph G' of G with $\text{diam}(G') \le \alpha\text{diam}(G)$ and $w(G') \le (2 + \frac{4}{(\alpha-1)\lfloor k/2 \rfloor})w(G^*)$ can be found in $O(kn(m + n\log n)\log n)$ time, where G^* is a minimum k-edge connected spanning subgraph of G.*

Proof By the algorithm of Khuller and Vishkin [17], an approximate minimum k-edge connected spanning subgraph G'' of G can be found in $O(kn(m+n\log n)\log n)$ time and $c(k) = 2$. All other steps can be finished in $O(mn + n^2 \log n)$ time. By Theorem 2, the corollary then follows. $\square$

Corollary 3 *Let a weighted undirected graph $G(V, E)$ be k-vertex connected. Then, for any fixed $\alpha > 1$, a k-vertex connected spanning subgraph G' of G with $\text{diam}(G') \le \alpha\text{diam}(G)$ and $w(G') \le (2+\frac{4}{(\alpha-1)\lfloor k/2 \rfloor})H(k)w(G^*)$ can be found in $O(k^3m'n^2)$ time, where $m' = min\{kn, m\}$, G^* is a minimum k-vertex connected spanning subgraph of G, and $H(k) = \sum_{i=1}^{k} 1/i$.*

Proof By the algorithm of Ravi and Williamson [23], an approximate minimum k-vertex connected spanning subgraph G'' of G can be found in $O(k^3m'n^2)$ time and $c(k) = 2\sum_{i=1}^{k} 1/i$ where $m' = min\{kn, m\}$. All other steps can be finished in $O(mn+n^2\log n)$ time. By Theorem 2, the corollary then follows. $\square$

When $k = 2, 3, 4$ or 5, there is a better algorithm for finding a k-vertex connected spanning subgraph, simultaneously minimizing its diameter and weight.

Corollary 4 *Let a weighted undirected graph $G(V, E)$ be k-vertex connected, with $k = 2, 3, 4$ or 5. Then, for any fixed $\alpha > 1$, a 2-vertex connected spanning subgraph G' of G whose weight is twice the optimum can be found in $O(n^3m)$ time; a 3-vertex connected spanning subgraph whose weight is twice the optimum can be found in $O(n^2m\log n)$ time; and a 4- or 5-vertex connected spanning subgraph whose weight is 3 times the optimum can be found in $O(n^3m)$ time. Note that all such subgraphs have diameters which are α times the diameter of G.*

Proof By the algorithm of Penn and Shasha-Krupnik [22], an approximate minimum 2-vertex connected spanning subgraph G'' of G can be found in $O(n^3m)$ time, and $c(2) = 2$. By the algorithm of Dinitz and Nutov [5], an approximate minimum 3-vertex connected spanning subgraph G'' of G can be found in $O(n^2m \log n)$ time, and $c(3) = 2$. Also a k-vertex connected spanning subgraph such that $\mathrm{diam}(G') \leq \alpha\mathrm{diam}(G)$ and $w(G') \leq 3w(G^*)$ can be found in $O(n^3m)$ time for $k = 4$ or 5. By Theorem 2, the corollary then follows. □

Sometimes, geometry may help. For example, if the edge weights of G satisfy the triangle inequality, we have

Corollary 5 *Let a weighted undirected graph $G(V, E)$ be k-vertex connected. If the edge weights satisfy the triangle inequality, then for any fixed $\alpha > 1$, a k-vertex connected spanning subgraph G' of G with* $\mathrm{diam}(G') \leq \alpha\mathrm{diam}(G)$ *and* $w(G') \leq (2 + \frac{4}{(\alpha-1)\lfloor k/2 \rfloor})(1 + \frac{k-1}{n})w(G^*)$ *can be found in $O(k^2n^2m)$ time, where G^* is a minimum k-vertex connected spanning subgraph of G.*

Proof By the algorithm of Khuller and Raghavachari [13], if the edge weights of G satisfy the triangle inequality, an approximate minimum k-vertex connected spanning subgraph G'' of G can be found in $O(k^2n^2m)$ time, and $c(k) = 2(1 + (k-1)/n)$. All other steps can be finished in $O(mn + n^2 \log n)$ time. By Theorem 2, the corollary then follows. □

Corollary 6 *Given a weighted undirected biconnected graph $G(V, E)$ with edge weights satisfying the triangle inequality, for any fixed $\alpha > 1$, a biconnected spanning subgraph G' of G with* $\mathrm{diam}(G') \leq \alpha\mathrm{diam}(G)$ *and* $w(G') \leq (1.5 + \frac{3}{\alpha-1})w(G^*)$ *can be found in time $O(n^{5/2})$, where G^* is a minimum biconnected spanning subgraph of G.*

Proof Frederickson and JáJá [7] present a better approximation solution for graphs with edge weights satisfying the triangle inequality. Their algorithm finds a biconnected spanning subgraph within 1.5 times the optimum. The dominant time complexity of their algorithm is in finding a minimum-cost maximum matching in an n-vertex graph, which requires $O(n^{5/2})$ time. □

4 Finding a 2-Connected Spanning Subgraph

In this section we introduce another algorithm for the special case $k = 2$. The algorithm is basically adapted from the algorithm for 2-connectivity augmentation by Khuller and Ragahavahavi [16]. Suppose $G(V, E)$ is a weighted undirected 2-connected graph. The algorithm runs as follows. First find an MST, T_{mst}, of G, and find a shortest path tree T_r of G such that $\text{diam}(T_r) = \text{diam}(G)$. Then for any given $\alpha > 1$, apply the algorithm due to Khuller et al [14] to find an $(\alpha, \frac{2}{\alpha-1})$-LAST rooted at r, denoted by T_A, using edges from T_r and T_{mst}. Third, augment T_A to make it 2-connected, using edges in $E - T_A$. The augmentation procedure is as follows. Construct an auxiliary directed graph $G^D(V, E^D)$ such that G^D is strongly connected, based on T_A. Find an optimal branching rooted at r in G^D. Let E_D be the edge set of the branching, and let E_B be the corresponding edge subset of E, which produces E_B in the construction of G^D. Finally, produce a graph $G' = T_A \cup E_B$, and G' is 2-connected as proved in [16].

If we are dealing with 2-edge connectivity augmentation, $G^D = (V, E^D)$ is constructed as follows [16]. For every tree edge (u, v) in T_A, assume that u is the parent of v in T_A, then add a directed edge $\langle v, u\rangle$ to E^D, and assign weight 0 to this edge. For every non-tree edge $(u, v) \in E - T_A$, if u is the parent of v in T_A, then add a directed edge $\langle u, v\rangle$ to E^D, and assign the weight of its undirected edge to this edge. If neither one is an ancestor of the other, let $t = LCA(u, v)$ be the lowest common ancestor of u and v in T_A, then add two directed edges $\langle t, u\rangle$ and $\langle t, v\rangle$ to E^D, and assign the weight of (u, v) to each of them. Obviously G^D is strongly connected.

If we deal with 2-vertex connectivity (biconnectivity) augmentation, $G^D = (V, E^D)$ is constructed in a similar way [16]. After the same start, in this case if neither one is the ancestor of the other, then add *four* directed edges $\langle t, u\rangle$, $\langle t, v\rangle$, $\langle u, v\rangle$ and $\langle v, u\rangle$ to E^D. As a result, G^D is strongly connected.

Now we analyze the performance guarantee of the above algorithms. Recall E_B is the corresponding edge subset of E in $E_D - T_A$, which produces E_B by the construction of G^D. We have

Lemma 1 *$w(E_B) \leq 2w(G^*)$.*

Proof First we construct another directed graph G'^D with the above approach, using the edges of T_A and the edges of G^*. Then we find an optimal branching rooted at r in G'^D. Obviously G'^D is also strongly connected because G^* is 2-connected. Let T_B and T'_B be optimal branchings of G^D and G'^D respectively. Then

$$\begin{aligned} w(T_B) &\leq w(T'_B) \\ &\leq 2w(G^*) \end{aligned} \tag{6}$$

because G'^D is a subgraph of G^D, and every edge of G^* appeared in T'_B at most twice. Note that some of the edges in T_B or T'_B coming from T_A have weights 0. Let E_B be the corresponding undirected edge set of T_B which does not contain any edge in T_A. Then by inequality (6)

$$\begin{aligned} w(E_B) &\leq w(T_B) \\ &\leq 2w(G^*). \end{aligned} \tag{7}$$

□

Theorem 3 *Given a weighted undirected 2-connected graph $G(V, E)$, for any fixed $\alpha > 1$, an $(\alpha, 3+\frac{2}{\alpha-1})$-LDLWSS$_2$ can be found in $O(mn+n^2 \log n)$ time.*

Proof We first analyze the performance guarantee for G' obtained by the proposed algorithm. We then give the computational complexity for finding G'.

It is obvious that a minimum spanning tree of G is the minimum connected spanning subgraph of G. So, $w(T_{mst}) \leq w(G^*)$. Then, we have the following inequality:

$$\begin{aligned} w(G') &= w(T_A \cup E_B) \leq w(T_A) + w(E_B) \\ &\leq (1 + \frac{2}{\alpha - 1})w(T_{mst}) + w(E_B) \quad \text{(by Corollary 1)} \\ &\leq (3 + \frac{2}{\alpha - 1})w(G^*) \quad \text{(by Lemma 1).} \end{aligned}$$

It is easy to show that $\text{diam}(G') \leq \alpha\text{diam}(G)$ because T_A is a subgraph of G' and $\text{diam}(T_A) \leq \alpha\text{diam}(G)$.

The time complexity of the proposed algorithms is as follows. Finding a shortest-path tree T_r such that $\text{diam}(T_r) = \text{diam}(G)$ takes $O(mn+n^2 \log n)$ time, by checking the shortest-path tree rooted at every vertex. (Each tree takes $O(m + n \log n)$ time [8].) Then finding an MST of G, T_{mst}, takes $O(m+n \log n)$ time [9]. The construction of G^D needs $O(n^2)$ time. Finding an optimal branching of G^D takes $O(m + n \log n)$ time [9]. Therefore G' can be found in $O(mn + n^2 \log n)$ time. □

Theorem 4 *Given a weighted 2-connected graph $G(V, E)$, for any fixed $\alpha > 1$, an $(\alpha, 3 + \frac{2}{\alpha-1})$-LDLWSS$_2$ can be found in $O(\log^2 n)$ time using $O(n^3)$ processors on a* CRCW PRAM.

Proof The parallel implementation of the above algorithm can be done as follows. T_r can be found in $O(\log^2 n)$ time using $O(n^3)$ processors on a CRCW PRAM by matrix multiplication. T_{mst} can be found in $O(\log^2 n)$ time using $O(n^2/\log^2 n)$ processors on a CREW PRAM [3]. T_A can be found in $O(\log n)$ time using $O(n)$ processors on a CREW PRAM by Corollary 1. G^D can be constructed in $O(1)$ time using $O(n^2)$ processors on a CRCW PRAM provided that the lowest common ancestor of each pair of vertices in T_A can be found in $O(1)$ time. Given two vertices in T_A, their lowest common ancestor can be found in $O(1)$ time using an auxiliary tree T', where the construction of T' needs $O(\log n)$ time and $O(n)$ processors on an EREW PRAM [24]. Finding an optimal branching in G^D requires $O(\log^2 n)$ time and $O(n^3)$ processors [20]. Thus, E_B can be found in $O(1)$ time using $O(n)$ processors on a CREW PRAM because $|T_B| \le n-1$. As a result, the graph $G' = T_A \cup E_B$ can be found in $O(\log^2 n)$ time using $O(n^3)$ processors on a CRCW PRAM. $\Box$

5 Associated Steiner Problems

In this section we generalize k-connected spanning subgraphs to Steiner k-connected subgraphs in the following way. First, the *minimum Steiner k-connected subgraph problem* is defined as follows. Given a weighted undirected k-connected graph $G(V,E)$ and a set $S \subset V$, to find a subgraph $G^*[S]$ of G spanning all vertices of S such that there are k edge-disjoint or vertex-disjoint paths in $G^*[S]$ between any pair of vertices in S and $w(G^*[S])$ is minimized. The problem is NP-hard because it becomes the famous Steiner tree problem when $k = 1$, which is NP-complete [10].

Lemma 2 *Let $G(V,E)$ be a weighted undirected 2-connected graph, and $S \subset V$. Then the minimum Steiner 2-connected subgraph $G^*[S]$ of G is 2-connected.*

Proof By the definition of the minimum Steiner 2-connected subgraph, for every two vertices $u, v \in S$, there are 2-edge (vertex) disjoint paths in $G^*[S]$ between u and v. In the following we prove the 2-edge connectivity case. The approach to 2-vertex connectivity case is similar and is omitted.

Let $G^*[S]$ be the minimum Steiner 2-edge subgraph spanning all vertices in S. Obviously $G^*[S]$ is connected. Assume that $G^*[S]$ is not 2-edge connected. Then, there must be a bridge (u,v) in $G^*[S]$. At least one of u and v is not in S (otherwise there are 2 edge-disjoint paths between u and v by the definition of $G^*[S]$). Now we construct a spanning tree T of $G^*[S]$. Then (u,v) is in T. Let T_u and T_v be the subtrees containing u and v respectively, after deleting (u,v) from T. Then a vertex $u' \in S$ must be in T_u, and a vertex $v' \in S$ must be in T_v. If this claim is true, then there is only one path in $G^*[S]$ between u' and v' because (u, v) is a bridge, which contradicts the definition of $G^*[S]$. Otherwise, assume that T_v does not contain any vertex of S. Then the deletion of all vertices from T_v and their incident edges from $G^*[S]$ will result in a remaining graph $G'[S]$, which is still a Steiner 2-edge connected graph, and $G'[S]$ has a smaller weight than that of $G^*[S]$. This contradicts the initial assumption that $G^*[S]$ is the minimum Steiner 2-edge connected subgraph. So, there is no bridge in $G^*[S]$, and $G^*[S]$ is 2-edge connected. □

Lemma 2 does not extend to arbitrary Steiner k-connected subgraphs. For $k > 2$ it is possible for each pair of vertices in S to be k-edge (vertex) connected in $G^*[S]$ while $G^*[S]$ is not k-edge (vertex) connected.

Next, we generalize the minimum Steiner k-connected subgraph problem further. Let $S \subset V$ be a given set in a weighted undirected k-connected graph $G(V, E)$. The problem is to find a subgraph $G[S]$ spanning all vertices of S, such that there are k edge-disjoint (vertex-disjoint) paths joining each pair of vertices in S, and both the diameter and the weight of $G[S]$ are minimized. The following theorem shows that this problem is NP-hard too.

Theorem 5 *Given a weighted undirected k-connected graph $G(V, E)$ and a vertex subset $S \subset V$, the problem is to find a subgraph $G[S]$ of G spanning all vertices in S such that, for each pair of vertices in S, there are k edge-disjoint (vertex-disjoint) paths in $G[S]$, and the diameter and the weight of $G[S]$ are minimized simultaneously. This problem is NP-hard.*

Proof When $k = 1$, and we allow the diameter of $G[S]$ to be ∞, the problem becomes the Steiner tree problem. □

Theorem 6 *Let $G(V, E)$ be a weighted undirected graph with diameter* $\mathrm{diam}(G)$ *and $S \subset V$ be a vertex set. Then, for any fixed $\alpha > 1$, a tree $T[S]$ spanning all vertices of S such that* $\mathrm{diam}(T[S]) \leq \alpha \mathrm{diam}(G)$ *and* $w(T[S]) \leq (2 + \frac{4}{\alpha-1})w(T^*[S])$ *can be found in $O(mn + n^2 \log n)$ time. $T[S]$ can also be found in $O(\log^2 n)$ time using $O(n^3)$ processors on a* CRCW PRAM, *where $T^*[S]$ is a Steiner tree of G.*

Proof Let T_r be a shortest-path tree of G rooted at r such that $\mathrm{diam}(T_r) = \mathrm{diam}(G)$. Let $T[S]$ be an approximate Steiner tree, produced by the algorithm of Kou et al [19]. If r is also in $T[S]$, then select r as the root of $T[S]$, apply the algorithm of Khuller et al [14] to produce a tree T_A such that $\mathrm{diam}(T_A) \leq \alpha \mathrm{diam}(T_r) = \alpha \mathrm{diam}(G)$ using the edges in T_r, by traversing

$T[S]$ using depth-first search. Otherwise, select a vertex $w \in S$ from T_r as its root, and recompute the distances from all other vertices to w in T_r. Also, select w as the root of $T[S]$, and the other operations are the same as above. Obviously T_A spans all vertices in S, $\mathrm{diam}(T_A) \leq \alpha \mathrm{diam}(G)$, and $w(T_A) \leq (1 + \frac{2}{\alpha-1})w(T[S]) \leq (2 + \frac{4}{\alpha-1})w(T^*[S])$ because $w(T[S]) \leq 2w(T^*[S])$ by the algorithm of Kou et al [19].

Now we analyze the time complexity. Finding T_r needs $O(mn+n^2 \log n)$ time. Finding the approximate Steiner tree T needs $O(m+n \log n)$ time [18], and can also be done in $O(\log^2 n)$ time using $O(n^3)$ processors on a CRCW PRAM by a straightforward parallel implementation of the original algorithm [19]. The tree T_A can be obtained in $O(n)$ sequential time, or in $O(\log n)$ time using $O(n)$ processors on a CRCW PRAM by Corollary 1. Therefore, the algorithm requires $O(mn + n^2 \log n)$ sequential time, or $O(\log^2 n)$ parallel time using $O(n^3)$ processors on a CRCW PRAM. $\Box$

If the algorithm of Kou et al [19] is replaced by the algorithm of Zelikovsky [27], the approximate Steiner tree $T[S]$ obtained is 11/6 of optimal, i.e, $w(T[S]) \leq \frac{11}{6}w(T^*[S])$. Thus, we get a better approximation algorithm which produces a tree T_A such that $\mathrm{diam}(T_A) \leq \alpha \mathrm{diam}(G)$ and $w(T_A) \leq (\frac{11}{6} + \frac{22}{6(\alpha-1)})w(T^*[S])$, whereas the time for $T[S]$ needs $O(mn+|S|^4)$ [27]. So, the time for the algorithm is $O(mn+|S|^4+n^2 \log n)$.

Remarks: In [26], Williamson et al first considered a *generalized Steiner network problem* which is to find a minimum cost subgraph that has r_{ij} edge-disjoint paths between i and j, given requirements r_{ij} (≥ 0) for each pair i, j of vertices, where both $i \in S$ and $j \in S$. For this problem, they presented an approximation algorithm which delivers a solution of $2k$ times optimum where $k = \max_{i,j \in S}\{r_{ij}\}$. Later Goemans et al [11] improved the approximation factor to $2H(k)$. Recently Jain [12] presented an algorithm which delivers an approximation solution which is two times optimum. This is the first algorithm which delivers a solution which is of constant degree approximation. Clearly, the minimum Steiner k edge-connected subgraph problem is a special case in which $r_{ij} = k$. Thus, all algorithms for the generalized Steiner network problem can be used as a subroutine for our problem.

6 Conclusion

We have presented approximation algorithms with performance guarantees for constructing low-diameter and low-weight k-connected spanning subgraphs in weighted undirected k-connected graphs. The algorithms exhibit a trade-off between the diameter and the weight of the subgraph obtained.

Acknowledgments

The authors were partially supported by Australian Research Council Large Grant A49702337. The work of the first author was done in part while he was affiliated with the Centre for Discrete Mathematics and Computing at The University of Queensland.

References

[1] V. Auletta and M. Parente. Better algorithms for minimum weight vertex-connectivity problems, *STACS'97*, Lecture Notes in Computer Science 1200 (1997) 547–558.

[2] B. Awerbuch, A. Baratz and D. Peleg. Cost-sensitive analysis of communication protocols. *Proc. 9th Symp. on Principles of Distributed Computing*, ACM Press (1989) 177–187.

[3] F.Y. Chin, J. Lam and I-N. Chen. Efficient parallel algorithms for some graph problems. *Commun. ACM* 25 (1982) 659–665.

[4] J. Cong, A.B. Kahng, G. Robins, M. Sarrafzadeh and C.K. Wong. Provably good performance-driven global routing. *IEEE Trans. Computer-Aided Design* 11 (1992) 739–752.

[5] Y. Dinitz and Z. Nutov. Finding optimum k-vertex connected spanning subgraphs: improved approximation algorithms for $k = 3, 4, 5$. *Algorithms and Complexity–Third Italian Conf.*, Lecture Notes in Computer Science 1203 (1997) 13–24.

[6] K.P. Eswaran and R.E. Tarjan. Augmentation problems. *SIAM J. Comput.* 5 (1976) 653–665.

[7] G.N. Frederickson and J. JáJá. On the relationship between the biconnectivity augmentation and traveling salesman problems. *Theoretical Computer Sci.* 19 (1982) 189–201.

[8] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34 (1987) 596–615.

[9] H.N. Gabow, Z. Galil, T. Spencer and R.E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* 6 (1986) 109–122.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness.* W.H. Freeman (1979).

[11] M.X. Goemans, A. Goldberg. S. Plotkin, D. Shmoys, E. Tardos and D.P. Williamson. Approximation algorithms for network design problems. *ACM–SIAM SODA* (1994) 223-232.

[12] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *IEEE FOCS* (1998).

[13] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *J. Algorithms* 21 (1996) 434–450.

[14] S. Khuller, B. Raghavachari and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica* 14 (1995) 305–321.

[15] S. Khuller and B. Schieber. Efficient parallel algorithms for testing k-connectivity and finding disjoint s-t paths in graphs. *SIAM J. Comput.* 20 (1991) 352–375.

[16] S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. Algorithms* 14 (1993) 214–225.

[17] S. Khuller and U. Vishkin. Biconnectivity approximations and graph carvings. *J. ACM* 41 (1994) 214–235.

[18] L. Kou. On efficient implementation of an approximation algorithm for the Steiner tree problem. *Acta Informatica* 27 (1990) 369–380.

[19] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica* 15 (1981) 141–145.

[20] L. Lovász. Computing ears and branchings in parallel. *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science* (1985) 464–467.

[21] K. Obraczka and P. Danzig. Finding low-diameter, low edge-cost, networks. *Technical Report*, TR 97-652, Dept. of Computer Sci., University of Southern California, LA (1997).

[22] M. Penn and H. Shasha-Krupnik. Improved approximation algorithms for weighted 2- and 3-vertex connectivity augmentation problems. *J. Algorithms* 22 (1997) 187–196.

[23] R. Ravi and D.P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica* 18 (1997) 21–43.

[24] B. Schieber and U. Vishkin. On finding lowest common ancestors: simplification and parallelization. *SIAM J. Comput.* 17 (1988) 1253–1262.

[25] D.B. West. *Introduction to Graph Theory.* Prentice Hall (1996).

[26] D.P. Williamson, M.X. Goemans, M. Mihail and V. V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problem. *Algorithmica* 15 (1995) 435–454.

[27] A.Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica* 9 (1993) 463–470.