




# On the Complexity of Finding 1-Center Spanning Trees

Pin-Hsian Lee  

National Taiwan Normal University, Taipei, Taiwan

Meng-Tsung Tsai  

Academia Sinica, Taipei, Taiwan

Hung-Lung Wang  

National Taiwan Normal University, Taipei, Taiwan

---

## Abstract

We consider the problem of finding a spanning tree  $T$  of a given undirected graph  $G$  such that any other spanning tree can be obtained from  $T$  by removing  $k$  edges and subsequently adding  $k$  edges, where  $k$  is minimized over all spanning trees of  $G$ . We refer to this minimum  $k$  as the *treeradius* of  $G$ .

Treeradius is an interesting graph parameter with natural interpretations: (1) It is the smallest radius of a Hamming ball centered at an extreme point of the spanning tree polytope that covers the entire polytope. (2) Any graph with bounded treeradius also has bounded treewidth. Consequently, if a problem admits a fixed-parameter algorithm parameterized by treewidth, it also admits a fixed-parameter algorithm parameterized by treeradius.

In this paper, we show that computing the exact treeradius for  $n$ -vertex graphs requires  $2^{\Omega(n)}$  time under the Exponential Time Hypothesis (ETH) and does not admit a PTAS, with an inapproximability bound of 1153/1152, unless  $P = NP$ . This hardness result is surprising, as treeradius has significantly higher ETH complexity compared to analogous problems on shortest path polytopes and star subgraph polytopes.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** Treeradius, Spanning tree polytope, Shortest  $s, t$ -path polytope

**Digital Object Identifier** 10.4230/LIPIcs.WADS.2025.43

**Funding** This research was supported in part by the National Science and Technology Council under contract NSTC 113-2628-E-003-001-MY3 and 113-2221-E-001-026.

**Acknowledgements** We want to thank the anonymous reviewers for their valuable comments.

## 1 Introduction

We consider the problem of finding a most representative spanning tree of a given undirected connected simple graph  $G = (V, E)$ . A spanning tree  $T$  of  $G$  is considered *most representative* if any other spanning tree of  $G$  can be obtained from  $T$  by removing  $k$  edges and subsequently adding  $k$  edges, where  $k$  is minimized over all spanning trees of  $G$ . We refer to this minimum  $k$  as *treeradius*. We show that if the treeradius of  $G$  is bounded, then the treewidth of  $G$  also is bounded. Consequently, if a problem admits a fixed-parameter algorithm parameterized by treewidth, it also admits a fixed-parameter algorithm parameterized by treeradius. Moreover, such a spanning tree has applications in devising algorithms for computing multi-objective spanning trees, which we discuss in Section 1.1.

The most representative spanning tree of  $G$  corresponds to a discrete 1-center among the extreme points in the spanning tree polytope of  $G$ , a concept that is well-established in the literature. Formally, for each spanning tree  $T$  of  $G$ , we define its indicator vector  $v(T) \in 2^E$ , where the  $i$ th coordinate of  $v(T)$  is 1 if the  $i$ th edge in  $E$  is included in  $T$ , and 0 otherwise.



© Pin-Hsian Lee, Meng-Tsung Tsai, and Hung-Lung Wang;  
licensed under Creative Commons License CC-BY 4.0

19th International Symposium on Algorithms and Data Structures (WADS 2025).

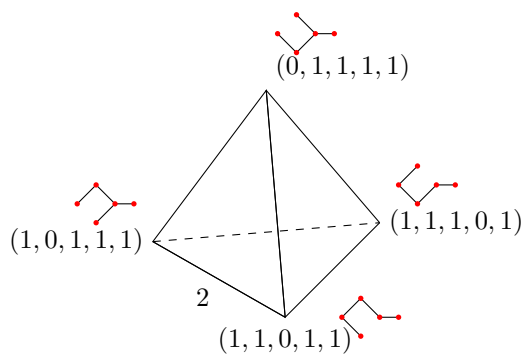
Editors: Pat Morin and Eunjin Oh; Article No. 43; pp. 43:1–43:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Placing the starting points of these indicator vectors at the origin in the space  $2^E$ , the convex hull of their endpoints is the *spanning tree polytope*. Since all indicator vectors have the same number of coordinates with value 1, their endpoints are precisely the extreme points of the polytope. A discrete (resp. continuous) *1-center* among these extreme points is an extreme point  $p$  (resp. any point  $p$  in the space) that minimizes the radius of the Hamming ball centered at  $p$  and covering the entire polytope. If the representing point of a spanning tree corresponds to a discrete 1-center of the polytope, we refer to it as a *1-center spanning tree* of  $G$ . Clearly, the notion of a 1-center spanning tree is equivalent to that of the most representative spanning tree. An illustration of the spanning tree polytope is depicted in Figure 1.



■ **Figure 1** The spanning tree polytope of the banner graph, a 5-vertex pseudotree with a 4-cycle, contains the maximum number of vertices among all spanning tree polytopes in three-dimensional space, as shown in Lemma 26.

Though a discrete 1-center in a set of  $n$  points can be found in  $O(n^2)$  time using a naive algorithm and cannot be computed in  $O(n^{2-\varepsilon})$  time for any constant  $\varepsilon > 0$  [1] unless the Hitting Set Conjecture [3] fails, finding a discrete 1-center among the extreme points in the spanning tree polytope is a fundamentally different and potentially more computationally intensive, as the number of spanning trees in  $G$  can be exponential in  $|V|$ . Assuming the Exponential Time Hypothesis (ETH) [16, 17], we have the following result:

► **Theorem 1.** *Given an  $n$ -vertex undirected connected simple graph  $G$ , finding a 1-center spanning tree of  $G$  requires  $2^{\Omega(n)}$  time unless ETH fails and can be solved in  $2^{\tilde{O}(n)}$  time. Moreover, computing the treeradius of  $G$  is APX-complete with inapproximability constant 1153/1152.*

We say that a problem has *ETH complexity*  $2^{\tilde{\Theta}(n^c)}$  if there exists an algorithm that solves it in  $2^{\tilde{O}(n^c)}$  time, and assuming ETH every algorithm solving this problem requires at least  $2^{\tilde{\Omega}(n^c)}$  time, where  $\tilde{\Omega}(f(n)) = \Omega(f(n)\text{polylog } n)$  and  $\tilde{O}(f(n)) = O(f(n)\text{polylog } n)$ . We find the ETH complexity  $2^{\tilde{\Theta}(n^1)}$  of 1-center spanning trees interesting when compared to the computation of discrete 1-centers among the extreme points of other polytopes associated with  $G$ , as summarized in Theorem 2. Finding a discrete 1-center among the extreme points naturally generalizes to any graph problem where the solution is chosen from a collection of edge sets of equal size, ensuring a one-to-one correspondence between solutions and the extreme points of the polytope in  $2^E$ . However, the polytopes we considered are not arbitrary and must satisfy the following conditions.

- We exclude the polytopes where outputting an extreme point is intractable. This is because outputting a discrete 1-center among the extreme points is no easier, so the hardness result for outputting an extreme point already implies the hardness of finding a discrete 1-center among the extreme points.

Typical examples include finding a discrete 1-center among the extreme points in the Hamiltonian path polytopes and the  $k$ -edge dominating set polytopes [13].

- We exclude the polytopes where enumerating all extreme points can be solved in polynomial time. This is because outputting a discrete 1-center among the extreme points is easy if all extreme points are given.

Typical examples include finding a discrete 1-center among the extreme points of the minimum cut polytope, which has  $O(n^2)$  minimum cuts [15], and the  $k$ -cycle polytope for any fixed  $k$ , which contains  $O(n^k)$   $k$ -cycles. In both cases, all extreme points can be identified efficiently.

► **Theorem 2.** *The ETH complexities of finding a discrete 1-center among the extreme points in the spanning tree polytope, the shortest  $s, t$ -path polytope, and the star subgraph  $S_r$  polytope of an  $n$ -vertex graph are*

$$2^{\tilde{\Theta}(n^1)}, 2^{\tilde{\Theta}(n^{1/2})}, 2^{\tilde{\Theta}(n^0)}, \text{ respectively.}$$

In addition, we note that finding a continuous 1-center in a set of  $n$  points is known to be NP-hard [12, 20] but admits a PTAS [21]. This complements the APX-hardness result in Theorem 1, as both problems involve finding a 1-center and have exponentially large solution spaces relative to their input sizes.

## 1.1 Applications

Finding a 1-center spanning tree has applications in devising algorithms for computing multi-objective spanning trees, such as max-color spanning trees [8, 23] and max-leaf spanning trees [7, 10, 19, 25].

For graph classes with treeradius at most  $k$ , a 1-center spanning tree  $T_{1\text{-center}}$  is a good approximation of a max-color spanning tree  $T_{\text{max-color}}$  because  $T_{\text{max-color}}$  can be obtained from  $T_{1\text{-center}}$  by removing  $k$  edges and subsequently adding  $k$  edges. So the number of colors in  $T_{1\text{-center}}$  cannot be less than that in  $T_{\text{max-color}}$  by  $k$ .

Similarly, for such graph classes, a 1-center spanning tree  $T_{1\text{-center}}$  is also a good approximation of a max-leaf spanning tree  $T_{\text{max-leaf}}$ . Because deleting  $k$  edges and subsequently adding  $k$  edges can change the degrees of at most  $4k$  nodes, the number of leaves in  $T_{1\text{-center}}$  cannot be less than that in  $T_{\text{max-leaf}}$  by  $4k$ .

## 1.2 Related Work

Finding a discrete (resp. continuous) 1-center has been widely studied when the input consists of a collection of strings. While related to our work, these studies differ in that our polytope is defined by specific edge subsets of an underlying graph, serving as a *succinct representation* of the polytope. In contrast, the strings in these related works are given explicitly and do not allow for such a compact representation. This distinction makes proving hardness in our setting more challenging.

The task in the previous work is defined as follows: given a set  $S$  of  $n$  length- $d$  strings over an alphabet  $\Sigma$ , find a string  $s^*$  that minimizes the radius of the Hamming ball centered at  $s^*$  enclosing all strings in  $S$ . There are two major variants:

- the continuous closest string problem [2, 5, 12, 14, 20, 21, 24], where the string  $s^*$  can be chosen from anywhere in  $\Sigma^d$ , and
- the discrete closest string problem [1, 2], where the string  $s^*$  can be chosen only from  $S$ .

## 1.3 Paper Organization

The paper is organized as follows. In Section 2, we establish our results on 1-center spanning trees, proving Theorem 1. Next, in Section 3 and Section 4, we analyze the problem of finding a discrete 1-center among the extreme points of the shortest  $s, t$ -path polytope and the star subgraph polytope, thereby proving Theorem 2. In Section 5, we present an extremal bound on the number of vertices in a spanning tree polytope in three-dimensional space using 2-distance sets. Finally, in Section 6, we explore the relationship between treewidth and treeradius.

## 2 1-Center Spanning Tree

The graph under consideration is assumed to be connected, simple, and undirected. We use  $G - F$  to denote the spanning subgraph of  $G$  with edge set  $E(G) \setminus F$ . For the difference  $A \setminus B$  of two sets  $A$  and  $B$ , when  $B$  is a singleton, say  $\{b\}$ , we write  $A - b$ . The number of components of a graph  $H$  is denoted by  $c(H)$ .

Let  $G$  be a graph, and let  $\Gamma(G)$  be the set of spanning trees of  $G$ . The 1-center spanning tree of  $G$  is a spanning tree with minimum *eccentricity*, where the eccentricity of a spanning tree  $T$  is defined as

$$\text{ecc}(T) = \max_{T' \in \Gamma(G)} (|E(T) \setminus E(T')| + |E(T') \setminus E(T)|).$$

Notice that the eccentricity of a 1-center spanning tree is exactly twice the treeradius of the graph. We start with a simple but essential observation regarding the eccentricity of a spanning tree.

► **Proposition 3.** *Let  $G$  be a graph of order  $n$ , and let  $T$  be a spanning tree of  $G$ . Then  $\text{ecc}(T) = 2(n - k)$ , where  $k = c(G - E(T))$ .*

**Proof.** Take  $k - 1$  edges from  $E(T)$  to connect the  $k$  components of  $G - E(T)$ . Along with a spanning forest of  $G - E(T)$ , we have a spanning tree that shares exactly  $k - 1$  edges with  $T$ , and thus  $\text{ecc}(T) \geq 2(n - k)$ . On the other hand, every spanning tree of  $G$  takes at least  $k - 1$  edges from  $E(T)$  to connect the components of  $G - E(T)$ , implying  $\text{ecc}(T) \leq 2(n - k)$ . ◀

► **Remark 4.** As an immediate result, computing the eccentricity of a spanning tree can be done in linear time.

A cut is a set of edges whose removal disconnects the graph<sup>1</sup>, and a  $k$ -cut for any integer  $k \geq 2$  is a cut whose removal results in at least  $k$  components. From Proposition 3 we know that the edge set of a 1-center spanning tree  $T$  is a cut that induces no cycle with  $c(G - E(T))$  as large as possible. It is worth noting that every connected graph  $G$  contains a spanning tree  $T$  with  $c(G - E(T)) \geq 2$ . A natural relaxation is to ask how a cut that induces no cycle can be computed, with the resulting components as many as possible. A cut that induces no cycle is called *cycle-free*.

► **Observation 5.** *For  $n \geq 2$ , complete graph  $K_n$  has a cycle-free  $k$ -cut if and only if  $k = 2$ . Moreover, any subgraph of  $K_n$  induced by a cycle-free 2-cut is isomorphic to  $K_{1,n-1}$ .*

<sup>1</sup> In the literature, a cut typically refers to a partition of the vertices into two subsets. Here, we slightly override this definition to better align it with the notion of a  $k$ -cut. Specifically, a cut in our context refers to a superset of the standard cut-set, the set of edges with end-vertices in different subsets of the partition.

We relate the eccentricity of a spanning tree with a cycle-free cut as follows.

► **Proposition 6.** *Let  $G$  be a graph of order  $n$ . There is a spanning tree  $T$  with  $\text{ecc}(T) \leq 2(n - k)$  if and only if  $G$  has a cycle-free  $k$ -cut.*

**Proof.** By Proposition 3, a spanning tree  $T$  with  $\text{ecc}(T) \leq 2(n - k)$  is a  $k$ -cut. Thus the necessity follows. For sufficiency, let  $F$  be a cycle-free  $k$ -cut. Since  $F$  induces no cycle, there is a spanning tree  $T$  of  $G$  such that  $F \subseteq E(T)$ . It follows immediately that  $k = c(G - F) \leq c(G - E(T))$ . By Proposition 3,  $\text{ecc}(T) = 2(n - c(G - E(T))) \leq 2(n - k)$ . ◀

By Propositions 3 and 6, we have the following equivalent definitions for a 1-center spanning tree.

► **Definition 7** (Restatements of 1-Center Spanning Trees). *Let  $T$  be a spanning tree of a graph  $G$ . The following statements are equivalent:*

- (i)  $T$  is a spanning tree with minimum eccentricity.
- (ii)  $T$  is a spanning tree that maximizes the number of components in  $G - E(T)$ .
- (iii)  $T$  is a spanning tree consisting of the edges in a cycle-free  $k$ -cut  $C$  with the largest possible  $k$  and the edges in a spanning tree of each component in  $G - E(C)$ .

We adopt these alternative definitions in the following sections. Our first main result, Theorem 1, is developed based on

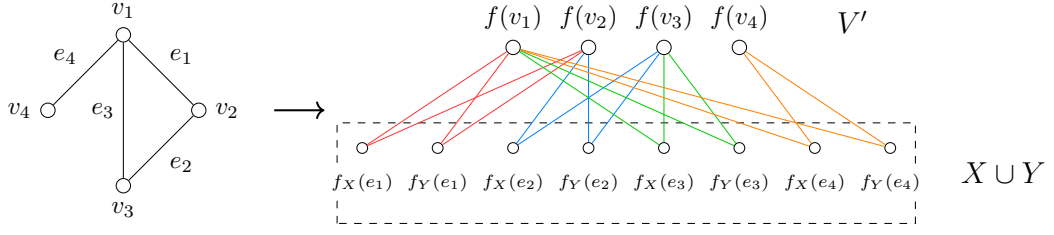
- Lemma 14, in Section 2.2, for the lower bound of  $2^{\Omega(n)}$  on the time complexity, assuming ETH;
- Corollary 19, in Section 2.3, for the APX-completeness and the inapproximability factor.

The problems involved in the subsequent discussion, including the sources of hardness, are named below.

- **CENTERSPANNINGTREE:** Given a graph  $G$ , find a spanning tree of  $G$  with minimum eccentricity. The optimal eccentricity is denoted by  $\text{OPT}_{\text{ecc}}$ .
- **CYCLEFREECUT:** Given a graph  $G$ , find a cycle-free cut  $F$  that maximizes  $c(G - F)$ .
- **CYCLEFREECUT-D:** This is the decision variant of **CYCLEFREECUT**. Namely, for a given graph  $G$  and an integer  $k$ , determine whether there is a cycle-free  $k$ -cut of  $G$ .
- **B-MAXINDEPSET:** Given a graph with maximum degree at most  $B$ , find an independent set of maximum size. The optimal size is denoted by  $\text{OPT}_{\text{ind}}$ .
- **B-INDEPENDENTSET:** Given a graph  $G$  with maximum degree at most  $B$  and an integer  $k$ , determine whether  $G$  contains an independent set of size  $k$ . We write **INDEPENDENTSET** if the graph under consideration has no degree constraint.

The following hardness results are known:

- **Theorem 8** (Johnson and Szegedy [18]). *There is no algorithm with running time  $2^{o(n)}$  to solve 3-MAXINDEPSET unless the Exponential Time Hypothesis (ETH) fails.*
- **Theorem 9** (Berman and Fujito [6]). *It is APX-hard to approximate 3-MAXINDEPSET.*
- **Theorem 10** (Chlebík and Chlebíková [11]). *It is NP-hard to approximate 4-MAXINDEPSET within a factor of  $\frac{48}{47}$ .*



■ **Figure 2** The reduction given in the proof of Theorem 11. The left is a graph  $G$ , which is the graph in an INDEPENDENTSET instance, while the right is the graph  $G'$  in the reduced instance of CYCLEFREECUT-D. The gadget for each edge of  $G$  is a 4-cycle in  $G'$ . The vertex set  $X \cup Y$  forms a clique.

## 2.1 NP-Hardness

A *split graph* is a graph whose vertex set can be partitioned into two sets  $I$  and  $K$ , where  $I$  is an independent set and  $K$  is a clique.

► **Lemma 11.** *Given a graph  $G$  and a positive integer  $k$ , it is NP-hard to determine whether  $G$  has a cycle-free  $k$ -cut, even when  $G$  is a split graph.*

**Proof.** We reduce INDEPENDENTSET to CYCLEFREECUT-D. Precisely, for an instance  $(G, k)$  of INDEPENDENTSET, we construct an instance  $(G', k + 1)$  of CYCLEFREECUT-D as follows (see Fig. 2). The vertex set of  $G'$  is  $V' \cup X \cup Y$ , where  $|V'| = |V(G)|$  and  $|X| = |Y| = |E(G)|$ . Each member of  $V'$  corresponds to a vertex of  $G$ , and each member of  $X$  or  $Y$  corresponds to an edge of  $G$ . We use the bijections  $f: V(G) \rightarrow V'$ ,  $f_X: E(G) \rightarrow X$ , and  $f_Y: E(G) \rightarrow Y$  to describe the correspondences, and for convenience let  $g = f^{-1}$ ,  $g_X = f_X^{-1}$ , and  $g_Y = f_Y^{-1}$ . Two vertices  $a$  and  $b$  of  $G'$  are adjacent if

- $a$  and  $b$  are incident in  $G$ ; namely, one of them corresponds to an edge of  $G$  and the other corresponds to an endpoint of the edge, or
- $\{a, b\} \subseteq X \cup Y$ .

We claim the following.

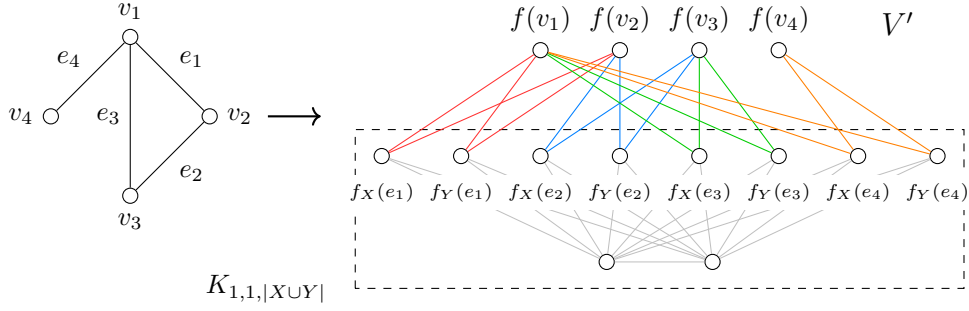
▷ **Claim 12.**  $G$  has an independent set of size at least  $k$  if and only if  $G'$  has a cycle-free  $(k + 1)$ -cut.

For  $k = 1$ , the claim holds due to Observation 5 so we assume that  $k \geq 2$ . For necessity, let  $I$  be an independent set of size  $k$  and let  $I' = f(I)$ . Consider the set of edges  $F = \{e \in E(G') : e \text{ has an endpoint in } I'\}$ . Clearly,  $G' - F$  contains at least  $k + 1$  components. Moreover, every member  $x$  in  $X \cup Y$  has at most one neighbor in  $I'$  since otherwise the two neighbors  $a$  and  $b$  of  $x$  have the corresponding vertices  $g(a)$  and  $g(b)$  adjacent in  $G$ . Thus, there is no cycle between  $I'$  and  $(V' \setminus I') \cup X \cup Y$ , showing that  $F$  is cycle-free.

For sufficiency, let  $F$  be a cycle-free  $(k + 1)$ -cut of  $G'$ . We show that

- $X \cup Y$  is contained in a component of  $G' - F$ , and
- the other components, each consisting of a single vertex, together correspond to an independent set of  $G$ .

First,  $G' - F$  contains at least three components. If the clique  $X \cup Y$  is not contained in a component, then by Observation 5 there exists  $x \in X \cup Y$  with  $x$  in a component and  $(X \cup Y) - x$  contained in another. Without loss of generality, assume that  $x \in X$ . The third



■ **Figure 3** The reduction from INDEPENDENTSET to CYCLEFREECUT-D on sparse graphs.

component contains a vertex, say  $v'$ , in  $V'$ . Let  $v = g(v')$ ,  $e = g_X(x)$ , and  $y = f_Y(e)$ . If  $v$  is an endpoint of  $e$ , then  $x$ ,  $y$ , and  $v'$  form a triangle contained in  $F$ . Otherwise,  $v$  is an endpoint of an edge  $e'$  of  $G$ . Let  $x' = f_X(e')$  and  $y' = f_Y(e')$ . Then  $v'$ ,  $x'$ ,  $y'$ , and  $x$  form a 4-cycle contained in  $F$ . Therefore,  $X \cup Y$  is contained in a component of  $G' - F$ .

For the second, except the component containing  $X \cup Y$ , all components consist of vertices from  $V'$ . Since  $V'$  induces an independent set of  $G'$ , each of these components consists of a single vertex. In addition, let  $u$  and  $v$  be the endpoints of an edge  $e$  in  $G$ , and let  $u' = f(u)$ ,  $v' = f(v)$ ,  $x = f_X(e)$  and  $y = f_Y(e)$  be the corresponding vertices in  $G'$ . If both  $u'$  and  $v'$  are in components not containing  $X \cup Y$ , then  $u'$ ,  $v'$ ,  $x$  and  $y$  form a 4-cycle, violating that  $F$  is cycle-free. Thus, the components not containing  $X \cup Y$  correspond to an independent set of  $G$  of size at least  $k$ .

Since  $V'$  induces an independent set and  $X \cup Y$  induces a clique, the graph  $G'$  is a split graph. Thus, the theorem follows. ◀

## 2.2 Exact 1-Center Spanning Trees in Sparse Graphs

Assuming ETH, solving 3-MAXINDEPSET takes  $2^{\Omega(n)}$  time, where  $n$  is the number of vertices of the input graph [18], as stated in Theorem 8. The same result can be obtained for CENTERSPANNINGTREE; that is, finding a 1-center spanning tree in a graph takes  $2^{\Omega(N)}$  time, assuming ETH, where  $N$  is the number of vertices of the graph. This follows from the reduction given in Lemma 11 with the hardness source replaced with a 3-MAXINDEPSET instance, making  $N = O(n)$ . Below, we strengthen the result, showing that the ETH lower bound of  $2^{\Omega(n)}$  ( $2^{\Omega(N)}$ ) still holds for finding a 1-center spanning tree in a sparse graph.

We modify the reduction given in the proof of Lemma 11. Recall that we reduce an independent set instance  $(G, k)$  to a cycle-free cut instance  $(G', k+1)$ , with  $V(G') = V' \cup X \cup Y$  (see Fig. 2). Instead of making  $X \cup Y$  a clique, we substitute it with a sparse structure, a complete 3-partite graph  $K_{1,1,|X \cup Y|}$  (see Fig. 3), to ensure the sparsity of the whole graph. Like a complete graph, a complete 3-partite graph admits cycle-free  $k$ -cuts only if  $k = 2$ , as shown in Lemma 13. With this property, an argument similar to that given in the proof of Lemma 11 can be obtained. Details are given in the proof of Lemma 14.

► **Lemma 13.** *For  $s \geq 3$  and  $k \geq 3$ , any complete  $s$ -partite graph has no cycle-free  $k$ -cut.*

**Proof.** Suppose to the contrary that there is a complete  $s$ -partite graph  $G$  that has a cycle-free  $k$ -cut  $F$ , for  $k \geq 3$ . Observe that there is a component of  $G - F$  that intersects at least two partite sets since otherwise  $F$  induces a  $C_3$ . Let  $Q$  be such a component. We claim that there is a  $C_3$  or a  $C_4$  connecting  $Q$  and two other components  $Q'$  and  $Q''$ .



If  $Q$  intersects only two partite sets, we take  $Q'$  as a component that intersects another partite set, say  $U$ , and  $Q''$  an arbitrary component other than  $Q$  and  $Q'$ . Let  $q'$  and  $q''$  be two vertices from  $Q'$  and  $Q''$ , respectively, with  $q' \in U$ . If  $q'$  and  $q''$  are adjacent, then there is a  $C_3$  induced by  $q'$ ,  $q''$ , and a vertex in  $Q$ ; otherwise, both  $q'$  and  $q''$  belong to  $U$ , and there is a  $C_4$  formed by  $q'$ ,  $q''$ , and two vertices from different partite sets in  $Q$ .

If  $Q$  intersects three or more partite sets,  $Q'$  and  $Q''$  can be arbitrary. Then an argument similar to the above applies. Consequently,  $F$  is not cycle-free in either case, which is a contradiction.  $\blacktriangleleft$

► **Lemma 14.** *Assuming ETH, there is no  $2^{o(N)}$ -time algorithm to compute a 1-center spanning tree of an  $N$ -vertex sparse graph.*

**Proof.** We reduce 3-INDEPENDENTSET to CYCLEFREECUT-D. Let  $(G, k)$  be an instance of 3-INDEPENDENTSET. The reduced instance,  $(G', k+2)$ , is constructed like the reduction given in the proof of Lemma 11. The vertex set  $V(G')$  is  $V' \cup X \cup Y \cup \{a, b\}$ , where  $|V'| = |V(G)|$ ,  $|X| = |Y| = |E(G)|$ . The correspondences are the following bijections:

- $f: V(G) \rightarrow V'$ ,
- $f_X: E(G) \rightarrow X$ , and
- $f_Y: E(G) \rightarrow Y$ .

Also, for convenience, let  $g = f^{-1}$ ,  $g_X = f_X^{-1}$ , and  $g_Y = f_Y^{-1}$ . The adjacency in  $G'$  is defined as follows. The subgraph induced by  $X \cup Y \cup \{a, b\}$  is a complete 3-partite graph, with partite sets  $\{a\}$ ,  $\{b\}$ , and  $X \cup Y$ . In addition, for every edge  $e$  of  $G$  with endpoints  $u$  and  $v$ , we make a 4-cycle  $f(u) \rightarrow f_X(e) \rightarrow f(v) \rightarrow f_Y(e) \rightarrow f(u)$ . See Fig. 3 for an illustration.

We claim that  $G$  has an independent set of size  $k$  if and only if  $G'$  has a cycle-free  $(k+2)$ -cut. For sufficiency, let  $F$  be a cycle-free  $(k+2)$ -cut. Then  $G' - F$  has at least  $k+2$  components. By Lemma 13,  $X \cup Y \cup \{a, b\}$  intersects at most two components so at least  $k$  components consist of vertices from  $V'$ . Since  $V'$  induces an independent set of  $G'$ , each of these components consists of a single vertex. Let  $u$  and  $v$  be two such components. There is no edge  $e$  of  $G$  with endpoints  $g(u)$  and  $g(v)$  since otherwise  $u$ ,  $f_X(e)$ ,  $v$ , and  $f_Y(e)$  form a 4-cycle. Thus, the components consisting of vertices in  $V'$  correspond to an independent set of  $G$ , with size at least  $k$ .

For necessity, let  $I$  be an independent set of  $G$  with  $|I| \geq k$ . Let  $F$  be the cut of  $G'$  such that the components  $G' - F$  are the subgraphs induced by the following vertex subsets:

- $\{f(u)\}$ , for each  $u \in I$ ,
- $(X \cup Y \cup \{a, b\} \cup (V' \setminus f(I))) - f_X(e)$ , where  $e$  is an arbitrary edge of  $G$ , and
- $\{f_X(e)\}$ .

Clearly,  $F$  is a  $(k+2)$ -cut. We show that  $F$  is cycle-free below. Let  $C$  be the component induced by  $(X \cup Y \cup \{a, b\} \cup (V' \setminus f(I))) - f_X(e)$ . Each vertex of  $C$  has at most one neighbor outside  $C$ , which implies that  $F$  induces no cycle passing through  $C$ . The remaining subgraph, induced by  $f_X(e) \cup f(I)$ , contains at most two edges so there is no cycle, either. Thus,  $F$  is a cycle-free  $(k+2)$ -cut.

The graph  $G'$  in the reduced instance is sparse. The reason is that it has  $n + 2m + 2$  vertices and  $8m + 1$  edges, where  $n$  and  $m$  are the numbers of vertices and edges of  $G$ , respectively. In addition, when the maximum degree of  $G$  is no more than three, we have  $N = O(n)$ , and by Theorem 8 the theorem follows.  $\blacktriangleleft$



Since the number of spanning trees in an  $N$ -vertex sparse graph is of  $2^{O(N)}$ , the naive method of enumerating all spanning trees and keeping the one with minimum eccentricity is optimal for sparse graphs.

► **Corollary 15.** *To compute a 1-center spanning tree of a sparse graph  $G$ , the naive method of checking all spanning trees of  $G$  is optimal, assuming ETH.*

## 2.3 Approximation

We show that there is a PTAS reduction from  $B$ -MAXINDEPSET to CENTERSPANNINGTREE, with the error parameter linearly preserved.

► **Lemma 16.** *Let  $G$  and  $G'$  be instances of  $B$ -MAXINDEPSET and CENTERSPANNINGTREE, respectively, with  $G'$  reduced from  $G$  by the reduction given in the proof of Lemma 11. For  $0 < \delta \leq (B^2 + 2B)^{-1}$  and  $\varepsilon > 0$ , given a spanning tree  $T$  of  $G'$  satisfying  $\text{ecc}(T) \leq (1 + \delta\varepsilon)\text{OPT}_{\text{ecc}}$ , an independent set  $I$  of  $G$  satisfying  $|I| \geq (1 - \varepsilon)\text{OPT}_{\text{ind}}$  can be computed in polynomial time.*

**Proof.** Recall that for a given graph  $G$ , the requested graph  $G'$  has  $V(G') = V' \cup X \cup Y$ , where there are one-to-one correspondences between  $V(G)$  and  $V'$ ,  $E(G)$  and  $X$ , and  $E(G)$  and  $Y$ , respectively. Each edge of  $G$  corresponds to a 4-cycle in  $G'$ , and  $X \cup Y$  forms a clique (see Fig. 2). We show

- (i) how a spanning tree  $T$  of  $G'$  corresponds to an independent set  $I$  of  $G$ , and
- (ii) the error parameter is linearly preserved; i.e., if  $\text{ecc}(T) \leq (1 + \delta\varepsilon)\text{OPT}_{\text{ecc}}$ , then  $|I| \geq (1 - \varepsilon)\text{OPT}_{\text{ind}}$ .

Let  $N = |V(G')|$ . For (i), we choose  $I$  to be the set of vertices that correspond to the single-vertex components of  $G' - T$ . Moreover, we claim the following.

► **Claim 17.**  $G$  has an independent set of size  $k$  if and only if  $G'$  has a spanning tree of eccentricity  $2(N - (k + 1))$ .

Since every spanning tree is a cycle-free cut, the sufficiency follows immediately from Claim 12. For necessity, by Claim 12 there is a cycle-free  $(k + 1)$ -cut, say  $F$ , with  $c(G' - F) = k + 1$ . Specifically, there is a subset  $U'$  of  $V'$  forming  $k$  of the components, where every single vertex forms one. The remaining component, say  $C$ , is the subgraph induced by  $X \cup Y \cup (V' \setminus U')$ . Since  $X \cup Y$  is a clique and every vertex of  $V' \setminus U'$  has degree at least two in  $C$ , there is a spanning tree  $T_F$  of  $G'$  containing all edges in  $F$  without disconnecting  $C$ . By Proposition 3,  $\text{ecc}(T_F) = 2(N - (k + 1))$ . According to the analysis above, it can be further derived that

$$\text{ecc}(T) = 2(N - (|I| + 1)), \quad (1)$$

and

$$\text{OPT}_{\text{ecc}} = 2(N - (\text{OPT}_{\text{ind}} + 1)). \quad (2)$$

For (ii), assume that  $\text{ecc}(T) \leq (1 + \delta\varepsilon)\text{OPT}_{\text{ecc}}$ . We claim that  $|I| \geq (1 - \varepsilon)\text{OPT}_{\text{ind}}$ . Let  $n = |V(G)|$ . Since the maximum degree of  $G$  is upper bounded by  $B$ , we have

$$N = |V(G')| + 2|E(G)| \leq (B + 1)n. \quad (3)$$

The chromatic number of a graph is upper bounded by the maximum degree plus one [9]. It follows that

$$\text{OPT}_{\text{ind}} \geq \frac{n}{B + 1}. \quad (4)$$

### 43:10 On the Complexity of Finding 1-Center Spanning Trees

Along with Eq. (2),

$$\frac{\text{OPT}_{\text{ind}}}{\text{OPT}_{\text{ecc}}/2} \underset{(4)(2)}{\geq} \frac{n/(B+1)}{N - \text{OPT}_{\text{ind}} - 1} \underset{(3)}{\geq} \frac{n/(B+1)}{(B+1)n - n/(B+1) - 1} = \frac{1}{B^2 + 2B - o(1)}. \quad (5)$$

Thus, we have

$$\begin{aligned} |I| &\underset{(1)}{=} N - \frac{\text{ecc}(T)}{2} - 1 \geq N - \frac{1}{2}(1 + \delta\varepsilon)\text{OPT}_{\text{ecc}} - 1 \underset{(2)}{\geq} \text{OPT}_{\text{ind}} - \frac{\delta\varepsilon}{2}\text{OPT}_{\text{ecc}} \\ &\underset{(5)}{\geq} \text{OPT}_{\text{ind}} - (B^2 + 2B)\delta\varepsilon\text{OPT}_{\text{ind}} \geq (1 - \varepsilon)\text{OPT}_{\text{ind}}. \end{aligned} \quad (6)$$

◀

► **Corollary 18.** *CENTERSPANNINGTREE is APX-complete even when the input is restricted to split graphs. Moreover, it is NP-hard to approximate within a factor of 1153/1152.*

**Proof.** The APX-hardness follows immediately from Lemma 16. For the inapproximability factor, by Theorem 10 we have that 4-MAXINDEPSET is NP-hard to approximate within a factor of 48/47. Taking  $B = 4$ ,  $\delta = (B^2 + 2B)^{-1}$ , and  $\varepsilon = 1/48$  in Lemma 16, the corollary is established. ◀

Analogous to how we modify the proof of Lemma 11 to obtain Lemma 14, by replacing  $G'$  in Lemma 16 with the sparse graph in Lemma 14, we have the following results.

► **Corollary 19.** *CENTERSPANNINGTREE is APX-complete even when the input is restricted to sparse graphs. Moreover, it is NP-hard to approximate within a factor of 1153/1152.*

### 3 1-Center Shortest Path

Let  $G$  be an undirected simple graph, and let  $s$  and  $t$  be two distinct vertices of  $G$ . The set of shortest  $s, t$ -paths is denoted by  $\mathcal{P}_{s,t}$ . For a path  $P$  and two vertices  $u$  and  $v$  on  $P$ , the subpath running between  $u$  and  $v$  is denoted by  $u \overset{P}{\rightsquigarrow} v$ ; if  $u$  and  $v$  are adjacent on  $P$ , we write  $u \rightarrow v$  when depicting the subpath. When referring to a path  $P$ , we use the same symbol  $P$  to denote the set of edges on the path. Analogous to the eccentricity of a spanning tree, the eccentricity of a shortest  $s, t$ -path  $P$  is defined to be

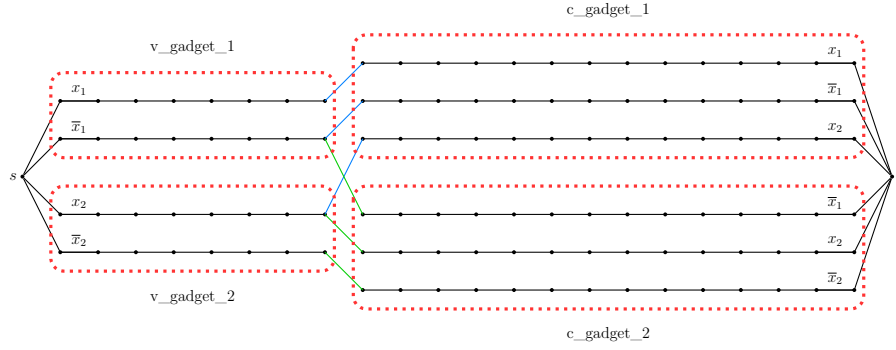
$$\text{ecc}(P) = \max_{Q \in \mathcal{P}_{s,t}} (|P \setminus Q| + |Q \setminus P|).$$

A shortest  $s, t$ -path  $P^*$  is said to be *1-center* if

$$\text{ecc}(P^*) = \min_{P \in \mathcal{P}_{s,t}} \text{ecc}(P).$$

The *distance* between two vertices  $u$  and  $v$  is denoted by  $d(u, v)$ , which is the number of edges on a shortest  $u, v$ -path. For an edge with endpoints  $u$  and  $v$  on a shortest  $s, t$ -path, we denote it by  $uv$  if  $d(s, u) < d(s, v)$ .

Notice that the two conditions for the polytopes we consider are satisfied since a shortest  $s, t$ -path can be computed in polynomial time, whereas enumerating all shortest  $s, t$ -paths requires exponential time in the worst case.



■ **Figure 4** The variable gadgets and clause gadgets. The instance of the 3SAT has two variables  $x_1$  and  $x_2$ , and two clauses  $c_1$  and  $c_2$ . In this example,  $c_1 = \{x_1, \bar{x}_1, x_2\}$  and  $c_2 = \{\bar{x}_1, x_2, \bar{x}_2\}$ . The upper and the lower paths in a variable gadget correspond to the positive and the negative literals of the variable, respectively.

### 3.1 NP-Hardness

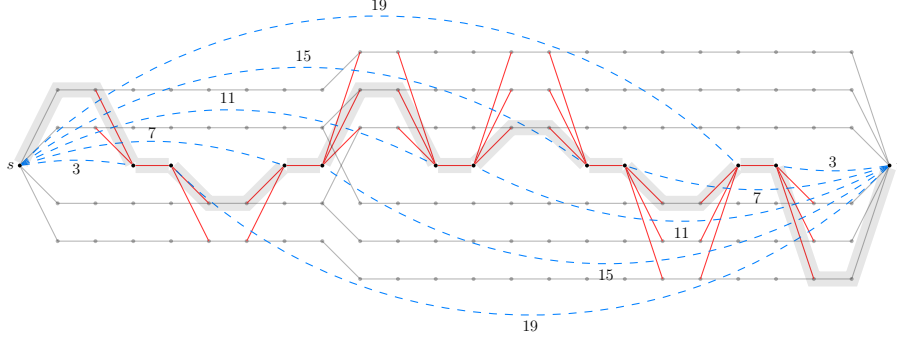
A *double star*  $S_{a,b}$ , with  $a \geq 2$  and  $b \geq 2$ , consists of two stars,  $K_{1,a}$  and  $K_{1,b}$ , along with an additional edge, called the critical edge, connecting the two internal nodes. We distinguish the two stars, the  $K_{1,a}$  and the  $K_{1,b}$ , of a double star as the left star and the right star, respectively. For two sets  $\mathcal{P} = \{P_1, \dots, P_p\}$  and  $\mathcal{Q} = \{Q_1, \dots, Q_q\}$  of disjoint paths, *bridging*  $\mathcal{P}$  and  $\mathcal{Q}$  using a double star  $S_{p,q}$  is to identify the  $p$  leaves of the left star of  $S_{p,q}$  with designated vertices on  $P_1, \dots, P_p$ , respectively, and to identify the  $q$  leaves of the right star with designated vertices on  $Q_1, \dots, Q_q$ , respectively. For two paths  $P$  and  $Q$ , we say that  $P$  *hits*  $Q$  if  $P \cap Q \neq \emptyset$ , and  $P$  *touches*  $Q$  if  $|P \cap Q| = 1$ .

► **Lemma 20.** *Given a graph  $G$  and two distinct vertices  $s$  and  $t$ , it is NP-hard to determine whether  $G$  contains a shortest  $s, t$ -path that hits all shortest  $s, t$ -paths in  $G$ ; i.e., to determine if there is a path  $P$  such that  $\text{ecc}(P) < 2d(s, t)$ .*

**Proof.** We give a reduction from 3SAT. We assume that each clause contains exactly three literals and each variable appears in some clauses. Consider a 3SAT instance  $\Phi$  with  $n$  variables  $x_1, \dots, x_n$  and  $m$  clauses  $c_1, \dots, c_m$ . In the reduced instance  $(G, s, t)$ , there are

- two distinguished vertices, one for  $s$  and the other for  $t$ ;
- $n$  variable gadgets,  $v\_gadget_1, \dots, v\_gadget_n$ , each consisting of two  $P_{4n5}$ ;
- $m$  clause gadgets,  $c\_gadget_1, \dots, c\_gadget_m$ , each consisting of three  $P_{8m-25}$ ;
- an assignment gadget, consisting of  $n + 2m - 1$  double stars; for each double star, the internal nodes  $u$  and  $v$  of the left and the right stars are connected to  $s$  and  $t$ , respectively, each with a path of specific length. We denote the  $s, u$ -path by  $R_s(u)$  and the  $v, t$ -path by  $R_t(v)$ .

Each path in the variable gadgets and the clause gadgets is called a *literal path*. For the two literal paths in a variable gadget, one corresponds to the positive literal and the other the negative one. For the three literal paths in a clause gadget, there is a one-to-one correspondence to the literals in the clause. To ease the presentation, we embed each literal path in the plane horizontally, with a left end and a right end. Vertices on a literal path are numbered from left to right. Let  $D(X, i) = \{v : v \text{ is the } i\text{th vertex on a path in gadget } X\}$ , where  $X$  is a variable gadget or a clause gadget. The gadgets are interrelated as follows.



■ **Figure 5** The assignment gadget. In this example,  $n = 2$  and  $m = 2$ . There are five double stars, one  $S_{2,2}$ , one  $S_{2,3}$ , and three  $S_{3,3}$ s, induced by the red edges. The dotted arcs are the paths  $R_s(u)$  and  $R_t(v)$ , for  $uv$  the critical edges. The number beside each arc indicates the length, i.e. the number of edges, of the path. The shaded path corresponds to the truth assignment  $(x_1, x_2) = (\text{true}, \text{true})$ . For each clause, the shaded path passes through two literal paths of the three, excluding one that corresponds to a true literal.

The left end of each literal path in a variable gadget is made adjacent to  $s$ ; the right end of each literal path in a clause gadget is made adjacent to  $t$ ; for each pair of literal paths  $(P, Q)$ , with  $P$  in a variable gadget and  $Q$  in a clause gadget, if both  $P$  and  $Q$  correspond to an identical literal, concatenate these two paths by making the right end of  $P$  and the left end of  $Q$  adjacent. We call the resulting graph  $H$ . See Fig. 4 for an example.

Consecutive variable gadgets, consecutive clause gadgets, and the literal paths in a clause gadget are bridged using the double stars in the assignment gadget. Details are given below.

- Each of the first  $n - 1$  double stars is an  $S_{2,2}$ , bridging a pair of consecutive variable gadgets. To bridge  $v\_gadget_i$  and  $v\_gadget_{i+1}$ , for  $i \in [n - 1]$ , the designated sets of vertices are  $D(v\_gadget_i, 4i - 2)$  and  $D(v\_gadget_{i+1}, 4i + 1)$ .
- The  $n$ th double star is an  $S_{2,3}$ , bridging the  $n$ th variable gadget with the first clause gadget. The designated sets of vertices are  $D(v\_gadget_n, 4n - 2)$  and  $D(c\_gadget_1, 1)$ .
- The following  $2m - 1$  double stars are  $S_{3,3}$ s. Except for the last one, every two of them bridge a clause gadget with itself, and the clause gadget with the next one; the last double star bridges the last clause gadget with itself only. For  $i \in [m]$ , to bridge  $c\_gadget_i$  with itself, the designated sets of vertices are  $D(c\_gadget_i, 8i - 6)$  and  $D(c\_gadget_i, 8i - 3)$ ; to bridge  $c\_gadget_i$  with  $c\_gadget_{i+1}$ , the designated sets of vertices are  $D(c\_gadget_i, 8i - 2)$  and  $D(c\_gadget_{i+1}, 8i + 1)$ .

For a critical edge  $uv$  of a double star in an assignment gadget, the numbers of vertices on  $R_s(u)$  and  $R_t(v)$  are chosen so that the path  $s \xrightarrow{R_s(u)} u \rightarrow v \xrightarrow{R_t(v)} t$  has length equal to a shortest  $s, t$ -path in  $H$  and remains a shortest  $s, t$ -path in  $G$ . This completes the construction of  $(G, s, t)$ . See Fig. 5 for an example.

► **Observation 21.** *If a shortest  $s, t$ -path passes through no critical edges, then the subpath excluding  $s$  and  $t$  consists of two literal paths, one from a variable gadget and the other from a clause gadget. Moreover, the two literal paths correspond to the same literal.*

We now show that  $\Phi$  is satisfiable if and only if there is a shortest  $s, t$ -path that hits all shortest  $s, t$ -paths in  $G$ . For necessity, given a satisfiable assignment, we consider a shortest  $s, t$ -path  $P$  such that

- $P$  passes through all critical edges in the assignment gadget;
- $P$  touches the path corresponding to a true literal of every variable gadget;
- $P$  touches two of the three paths of every clause gadget, excluding an arbitrary one corresponding to a true literal in the clause.

If there is a path  $Q$  not hit by  $P$ , then  $Q$  contains no critical edges. By Observation 21,  $Q$  contains two disjoint subpaths corresponding to the same literal. However, the one from the variable gadget corresponds to a false literal, and the other, which is from a clause gadget, corresponds to a true literal. This leads to a contradiction, so  $P$  is the requested path. For sufficiency, observe that the path  $P$  contains all critical edges. In addition,  $P$  touches exactly one literal path in each variable gadget. Let  $A$  be the truth assignment such that true literals have their corresponding literal paths in the variable gadgets touched by  $P$ . We claim that  $A$  satisfies  $\Phi$ . Suppose to the contrary that there is a clause that contains no true literal. Then, there is a literal whose corresponding literal paths in the variable gadget and the clause gadget are not hit by  $P$ , and so is the shortest  $s, t$ -path that passes through these two literal paths, a contradiction. ◀

► **Remark 22.** For a 3SAT instance of  $n$  variables and  $m$  clauses, the reduced instance in the proof above has  $\Theta((n + m)^2)$  vertices. It follows that computing a 1-center shortest path cannot be done in  $2^{o(\sqrt{n})}$  time, assuming ETH.

► **Corollary 23.** Let  $G$  be an  $n$ -vertex graph, and let  $s$  and  $t$  be two vertices of  $G$ . Assuming ETH, there is no  $2^{o(\sqrt{n})}$ -time algorithm to compute a 1-center shortest  $s, t$ -path of  $G$ .

### 3.2 An Exact Algorithm to Compute a 1-Center Shortest $s, t$ -Path

A naive way to compute a 1-center shortest  $s, t$ -path is to enumerate all shortest  $s, t$ -paths, and keep the one with minimum eccentricity. The eccentricity of a given shortest  $s, t$ -path  $P$  can be computed in polynomial time by finding a shortest  $s, t$ -path that uses the edges on  $P$  as few as possible. Thus, this naive method takes  $2^{O(n)}$  time. In the following, we aim for developing a  $2^{\tilde{O}(\sqrt{n})}$ -time algorithm. To achieve the reduced running time, we consider another way, a dynamic programming algorithm, to compute the eccentricity of a shortest  $s, t$ -path, say  $P$ . The vertex set of the graph is partitioned into layers, depending on the distances from  $s$ . For each vertex  $u$  in layer  $i$ , we record the least number of edges that an  $s, u$ -path shares with  $P$ . This value, which we call the intersection index of  $u$  with respect to  $P$ , can be computed recursively by finding an optimal path of the form  $s \rightsquigarrow u' \rightarrow u$ , where  $u'$  is some vertex in layer  $i - 1$ .

To compute the eccentricity of a 1-center shortest  $s, t$ -path, it suffices to compute for every shortest  $s, t$ -path  $P$  the intersection index of  $t$  with respect to  $P$  and maintain the minimum. We treat the intersection indices of the vertices in a layer with respect to  $P$  as a function, called the *index function* of  $P$  to the layer. A key observation is that for any layer, the number of different index functions is upper bounded by  $(d(s, t) + 1)^{|V_i|}$ . The idea of our algorithm is to distinguish the layers by their “thickness”, the numbers of vertices in the layers. For thin layers, we maintain all possible index functions. On the other hand, we enumerate all subpaths passing through the thick layers and compute its eccentricity using the naive method mentioned above. These two parts can be combined and balanced so that the requested running time  $2^{\tilde{O}(\sqrt{n})}$  is achieved.

Precisely, let  $l = d(s, t)$ , and let  $V_i = \{v \in V(G) : d(s, v) = i, d(v, t) = l - i\}$  for  $i \in \{0, \dots, l\}$ . The *layered subgraph*  $H$  of  $G$  with respect to  $s$  and  $t$  is defined as

#### 43:14 On the Complexity of Finding 1-Center Spanning Trees

- $V(H) = V_0 \cup \dots \cup V_l$  and
- $E(H) = \{uv \in E(G) : u \in V_{i-1}, v \in V_i, i \in [l]\}$ .

Each  $V_i$  is called a *layer*. A path  $u_0, \dots, u_k$  is said to be *forward* if for all  $i \in [k]$  there exists some  $j \in [l]$  such that  $u_{i-1} \in V_{j-1}$  and  $u_i \in V_j$ . Notice that there is a one-to-one correspondence between the set of shortest  $s, t$ -paths of  $G$  and that of forward  $s, t$ -paths of  $H$ . In the following we show how a 1-center forward  $s, t$ -path of  $H$  is computed.

A layer  $V_i$  is called *thin* if  $|V_i| \leq \sqrt{n}$  and *thick* otherwise. Let  $S$  be an *exact hitting set* of the thick layers; namely for all  $i \in [l]$

$$|S \cap V_i| = \begin{cases} 1, & \text{if } V_i \text{ is thick} \\ 0, & \text{otherwise.} \end{cases}$$

A forward path  $P$  is  *$S$ -governed* if every node  $u$  on  $P$  that belongs to a thick layer is in  $S$ . Let  $\mathcal{P}_{u,v}$  be the set of all forward  $u, v$ -paths in  $H$ , and, for a given exact hitting set  $S$  of thick layers, let  $\mathcal{S}_{s,t}$  be the set of  $S$ -governed forward  $s, t$ -paths in  $H$ .

For every possible  $S$ , we compute an  $S$ -governed forward  $s, t$ -path  $P^*$  with minimum eccentricity; namely, such a path  $P^*$  satisfies

$$\min_{Q \in \mathcal{P}_{s,t}} |P^* \cap Q| = \max_{P \in \mathcal{S}_{s,t}} \min_{Q \in \mathcal{P}_{s,t}} |P \cap Q|.$$

Let  $U_0, \dots, U_k$  be the thin layers. For each  $S$ -governed forward  $s, t$ -path  $P$ , we associate  $k+1$  functions  $g_i|_P: U_i \rightarrow \{0, \dots, l\}$ , the index function of  $P$  to layer  $U_i$ , for  $i \in \{0, \dots, k\}$ , and  $k$  functions  $h_i|_P: U_{i-1} \times U_i \rightarrow \{0, \dots, l\}$ , for  $i \in [k]$ , where

$$g_i|_P(v) = \min_{Q \in \mathcal{P}_{s,v}} |P \cap Q|$$

and

$$h_i|_P(u, v) = \begin{cases} \infty, & \text{if } \mathcal{P}_{u,v} = \emptyset \\ \min_{Q \in \mathcal{P}_{u,v}} |P \cap Q| & \text{otherwise.} \end{cases}$$

Notice that  $g_0|_P = \mathbf{0}$ , the zero function, and  $g_k|_P(t) = \min_{Q \in \mathcal{P}_{s,t}} |P \cap Q|$ . For  $i \in [k]$ , observe that

$$g_i|_P(v) = \min_{u \in U_{i-1}} (g_{i-1}|_P(u) + h_i|_P(u, v)). \quad (7)$$

Let

$$\mathcal{G}_i|_x = \{g_i|_P : P \in \mathcal{S}_{s,t}, P \text{ passes through } x, x \in U_i\}$$

and

$$\mathcal{H}_i|_{x,y} = \{h_i|_P : P \in \mathcal{S}_{s,t}, P \text{ passes through } x \text{ and } y, x \in U_{i-1}, y \in U_i\}.$$

► **Lemma 24.** *Let  $f: U_i \rightarrow \{0, \dots, l\}$  and  $y \in U_i$ . Then,  $f \in \mathcal{G}_i|_y$  if and only if there exists  $x \in U_{i-1}$ ,  $g \in \mathcal{G}_{i-1}|_x$ , and  $h \in \mathcal{H}_i|_{x,y}$  such that*

$$f: v \mapsto \min_{u \in U_{i-1}} (g(u) + h(u, v)).$$

**Proof.** The necessity follows immediately from the definition. For sufficiency, we claim that there is an  $S$ -governed forward  $s, t$ -path  $P$  such that  $g = g_{i-1}|_P$  and  $h = h_i|_P$ , and then by Eq. (7)

$$f(v) = \min_{u \in U_{i-1}} (g(u) + h(u, v)) = \min_{u \in U_{i-1}} (g_{i-1}|_P(u) + h_i|_P(u, v)) \stackrel{(7)}{=} g_i|_P(v),$$

for all  $v \in U_i$ . Such a path  $P$  can be constructed in a straightforward manner. Let  $\{P', P''\} \subseteq \mathcal{S}_{s,t}$  such that  $g = g_{i-1}|_{P'}$  and  $h = h_i|_{P''}$ . Clearly, the path  $s \rightsquigarrow_{P'} x \rightsquigarrow_{P''} y \rightsquigarrow_{P''} t$  is the requested one.  $\blacktriangleleft$

■ **Algorithm 1** An exact algorithm to find a 1-center shortest  $s, t$ -path of a graph  $G$ .

---

**Input:**  $(G, s, t)$   
**Output:** The minimum eccentricity of a shortest  $s, t$ -path

```

1  $H \leftarrow$  the layered subgraph of  $G$  w.r.t  $s$  and  $t$ 
2 for each exact hitting set  $S$  of the thick layers do
3   for  $x \in S$  do
4      $\mathcal{G}_0|_x \leftarrow \{\mathbf{0}\}$ 
5   for  $i \leftarrow 1$  to  $k$  do
6     for  $x \in U_{i-1}, y \in U_i$  do
7       compute  $\mathcal{H}_i|_{x,y}$ 
8     for  $y \in U_i$  do
9        $\mathcal{F} \leftarrow \emptyset$ 
10      for  $x \in U_{i-1}$  do
11        for  $g \in \mathcal{G}_{i-1}|_x, h \in \mathcal{H}_i|_{x,y}$  do
12          for  $v \in U_i$  do
13             $f(v) \leftarrow \min_{u \in U_{i-1}} (g(u) + h(u, v))$ 
14             $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$ 
15       $\mathcal{G}_i|_y \leftarrow \mathcal{F}$ 
16 return  $\max_S \max_{g \in \mathcal{G}_k|_t} g(t)$ 

```

---

**Time complexity.** A thick layer has size greater than  $\sqrt{n}$ , so there are no more than  $\sqrt{n}$  ones. Since each thick layer has size at most  $n$ , there are  $O(n^{\sqrt{n}})$  different exact hitting sets of the thick layers. For each exact hitting set  $S$  of the thick layers, since all  $S$ -governed forward  $s, t$ -paths  $P$  coincide on  $S$ , there is exactly one function in  $\mathcal{H}_i|_{x,y}$ , given  $x \in U_{i-1}$  and  $y \in U_i$ . This function can be computed in polynomial time by finding a forward  $x, y$ -path that uses the edges on  $P$  as little as possible. For  $x \in U_{i-1}$ , the set  $\mathcal{G}_{i-1}|_x$  contains  $O((l+1)^{\sqrt{n}})$  functions so along with Lemma 24 computing  $\mathcal{G}_i|_y$  for all  $y \in U_i$  takes  $O(|U_{i-1}| \cdot |U_i|^2 \cdot (l+1)^{\sqrt{n}})$  time. The overall running time is  $O(n^{\sqrt{n}} \cdot (\sqrt{n} + \sum_{i=1}^k (\text{poly}(n) + (|U_{i-1}| \cdot |U_i|^2 \cdot (l+1)^{\sqrt{n}})))) = 2^{\tilde{O}(\sqrt{n})}$ . See Algorithm 1 for the pseudocode.

#### 4 1-Center Star Subgraph $S_r$

We consider the problem of finding a discrete 1-center among the extreme points in the star subgraph polytope of an undirected graph. Given an undirected graph  $G = (V, E)$  and an integer  $r \geq 2$ , one can identify all the star subgraphs with  $r$  leaves. Each star subgraph



$S_r$  corresponds to an indicator vector  $v(S_r) \in 2^E$ , where the  $i$ th coordinate of  $v(S_r)$  is 1 if the  $i$ th edge in  $E$  is included in  $S_r$ , and 0 otherwise. Placing the starting points of these indicator vectors at the origin in the space  $2^E$ , the convex hull of their endpoints is the *star subgraph polytope*. Note that the indicator vector of each star subgraph  $S_r$  corresponds to an extreme point in the star subgraph polytope because every  $S_r$  contains exactly  $r$  edges.

The two conditions for the polytopes we considered are clearly satisfied since a star subgraph can be output in time linear to the input size, whereas enumerating all star subgraphs requires  $\Omega(n^r)$  time in the worst case, which is superpolynomial for superconstant  $r$ .

We show that:

► **Lemma 25.** *It takes linear time to find a star subgraph  $S_r$  such that every other star subgraph  $S'_r$  can be obtained from  $S_r$  with removing  $k$  edges and subsequently adding  $k$  edges and  $k$  is minimized over all choices of star subgraphs with  $r$  leaves.*

**Proof.** Assume that  $x \in V$  is the center node of the target  $S_r^*$ . Classify the star subgraph  $S_r$  other than  $S_r^*$  into the following three categories. Let  $\deg(x)$  denotes the degree of node  $x$  in  $G$ .

- $S_r$  shares the same center node with  $S_r^*$ . Thus,  $\deg(x) \geq r$ . Then the largest Hamming distance from any such  $S_r$  to  $S_r^*$  is  $2r$  if  $\deg(x) \geq 2r$  or  $2\deg(x) - 2r$  otherwise. Let  $k_1$  be the largest Hamming distance in this case.
- $S_r$  has the center node  $y$  and  $y \neq x$ , and  $G$  does not contain  $\{x, y\}$  as an edge. Thus,  $\deg(x) \geq r$  and  $\deg(y) \geq r$ . Then the largest Hamming distance from any such  $S_r$  to  $S_r^*$  is  $2r$ . Let  $k_2$  be the largest Hamming distance in this case.
- $S_r$  has the center node  $y$  and  $y \neq x$ , and  $G$  contains  $\{x, y\}$  as an edge. Thus,  $\deg(x) \geq r$  and  $\deg(y) \geq r$ . If  $\deg(y) \geq r + 1$ , then the largest Hamming distance from any such  $S_r$  to  $S_r^*$  is  $2r$ . Otherwise  $\deg(y) = r$ , such a center node  $y$  is the only chance that the choices of leaves of  $S_r^*$  matter. If there are more than  $r$  such center nodes  $y$ , then the largest Hamming distance from  $S_r^*$  to some star subgraph is  $2r$ . Otherwise, set  $k_3$  as  $2r - 1$  in this case.

Taking the maximum of  $k_1, k_2$ , and  $k_3$  gives the desired  $k$  under the assumption that  $x$  is the center node of the target  $S_r^*$ . To find the optimal center, we iterate over all possible nodes as candidates. Since each category above requires  $O(1)$  time, and the process is performed for each of the  $O(m)$  edges in  $G$ , the total running time remains linear in the input size. ◀

Lemma 25 proves the results on star subgraph polytopes stated in Theorem 2.

## 5 An Extremal Bound

In this section, we show that the polytope depicted in Figure 1 contains the maximum number of vertices in three-dimensional space.

► **Lemma 26.** *Any spanning tree polytope that is embeddable in three-dimensional space contains at most four vertices.*

**Proof.** We say a point set  $P$  is a  *$k$ -distance set* if the number of distinct pairwise distances between points in  $P$  is exactly  $k$ . In three-dimensional space, any 1-distance set contains at most 4 points and any 2-distance set contains at most 6 points [22].

Let  $G$  be an undirected simple graph whose spanning tree polytope is embeddable in 3D. Then  $G$  cannot contain any of the following graphs as a subgraph.

- Any cycle of length  $\ell \geq 5$ . Here is why. Let  $H$  be a spanning pseudotree of  $G$  that contains the cycle  $C_\ell$ . Then the number of vertices on the spanning tree polytope of  $H$  is  $\ell$  and each pair of the vertices has distance 2. This forms a 1-distance set of  $\ell \geq 5$  points, but every 1-distance set in 3D contains at most four points. Because these vertices is a subset of the vertices on the spanning tree polytope of  $G$ , then the spanning tree polytope of  $G$  cannot be embedded in 3D.
- Any pair of edge-disjoint simple cycles. Suppose that  $G$  contains a pair of edge-disjoint simple cycles  $C_\ell$  and  $C_t$  for some  $\ell, t \geq 3$ . Let  $H$  be a connected spanning subgraph of  $G$  that contains only the two cycles  $C_\ell$  and  $C_t$ . Then the number of vertices on the spanning tree polytope of  $H$  is  $\ell t$  and each pair of the vertices has distance either 2 or 4. This forms a 2-distance set of  $\ell t \geq 9$  points, but every 2-distance set in 3D contains at most 6 points. By a similar reason, the spanning tree polytope of  $G$  cannot be embedded in 3D.
- The diamond graph. Let  $H$  be a connected spanning subgraph of  $G$  that contains only the three simple cycles on the diamond. Then the number of vertices on the spanning tree polytope of  $H$  is 8 and each pair of the vertices has distance either 2 or 4. This forms a 2-distance set of 8 points, but every 2-distance set in 3D contains at most 6 points. By a similar reason, the spanning tree polytope of  $G$  cannot be embedded in 3D.
- The  $K_{2,3}$  complete bipartite graph. Let  $H$  be a connected spanning subgraph of  $G$  that contains only the three simple cycles on the  $K_{2,3}$ . Then the number of vertices on the spanning tree polytope of  $H$  is 12 and each pair of the vertices has distance either 2 or 4. This forms a 2-distance set of 12 points, but every 2-distance set in 3D contains at most 6 points. By a similar reason, the spanning tree polytope of  $G$  cannot be embedded in 3D.

Thus, to have the spanning tree polytope of  $G$  embeddable in 3D,  $G$  contains at most one cycle of length 3 or 4. The latter attains the claimed extremal bound. ◀

## 6 Concluding Remarks

As a final remark, we relate the two graph parameters: treeradius and treewidth, as stated in Theorem 28. Consequently, if a problem admits a fixed-parameter algorithm parameterized by treewidth, it also admits a fixed-parameter algorithm parameterized by treeradius.

► **Lemma 27** ([4]). *For any vertex  $v$  in a graph  $G$ , we have*

$$\text{treewidth}(G) \leq \text{treewidth}(G - v) + 1.$$

► **Theorem 28.** *For any graph  $G$ ,*

$$\text{treewidth}(G) \leq 2 \cdot \text{treeradius}(G) + 1.$$

**Proof.** Let  $F$  be an edge subset that induces a forest in  $G$ , and let  $\rho = n - c(G - F)$ , where  $n$  is the number of vertices of  $G$  and  $c(G - F)$  is the number of components of  $G - F$ . By Proposition 3, it suffices to show that  $\text{treewidth}(G) \leq 2\rho + 1$ . Denote by  $C_1, \dots, C_r$  the vertex sets of the *nontrivial components* of  $G - F$ , i.e., components of at least 2 vertices. Then

$$\rho = n - c(G - F) = \left( c(G - F) - r + \sum_{i=1}^r |C_i| \right) - c(G - F) = \sum_{i=1}^r (|C_i| - 1),$$

Since  $|C_i| - 1 \geq 1$  for  $i \in [r]$ , we have  $r \leq \rho$ . Hence  $\sum_{i=1}^r |C_i| = \rho + r \leq 2\rho$ .

Let  $H$  be the graph obtained by removing all vertices contained in the nontrivial components of  $G$ . As the remaining vertices in  $H$  can only be joined by edges in  $F$ , the graph  $H$  is a forest and thus has treewidth 1. Applying Lemma 27 repeatedly yields  $\text{treewidth}(G) \leq 2\rho + 1$ .  $\blacktriangleleft$

► **Remark 29.** Theorem 28 shows that if the treeradius of a graph is bounded then its treewidth is also bounded. Nonetheless, the converse does not hold in general. For example, as shown in Lemma 13, the complete tripartite graph  $K_{1,1,n}$  has treeradius of  $n$ , which grows unboundedly with  $n$ , even though its treewidth is 2.

---

## References

- 1 Amir Abboud, MohammadHossein Bateni, Vincent Cohen-Addad, Karthik C. S., and Saeed Seddighin. On complexity of 1-center in various metrics. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 275 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.1.
- 2 Amir Abboud, Nick Fischer, Elazar Goldenberg, Karthik C. S., and Ron Safier. Can you solve closest string faster than exhaustive search? In *31st Annual European Symposium on Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 3:1–3:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ESA.2023.3.
- 3 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and fixed parameter subquadratic algorithms for radius and diameter in sparse graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–391. SIAM, 2016. doi:10.1137/1.9781611974331.CH28.
- 4 Lowell W. Beineke, Robin J. Wilson, Ortrud R. Oellermann, Dirk Meierling, Lutz Volkmann, Matthias Kriesell, Kiyoshi Ando, R. J. Faudree, Michael Ferrara, Ronald J. Gould, Dieter Rautenbach, Bruce Reed, Ian Anderson, Keith Edwards, Graham Farr, Béla Bollobás, Oliver Riordan, F. T. Boesch, A. Satyanarayana, C. L. Suffel, Abdol-Hossein Esfahanian, R. A. Bailey, and Peter J. Cameron, editors. *Topics in Structural Graph Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, UK, 2012.
- 5 Amir Ben-Dor, Giuseppe Lancia, Jennifer Perone, and R. Ravi. Banishing bias from consensus sequences. In *Combinatorial Pattern Matching, 8th Annual Symposium (CPM)*, volume 1264 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 1997. doi:10.1007/3-540-63220-4\_63.
- 6 Piotr Berman and Toshihiro Fujito. On approximation properties of the independent set problem for degree 3 graphs. In *Algorithms and Data Structures*, pages 449–460, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
- 7 Paul S. Bonsma. Spanning trees with many leaves in graphs with minimum degree three. *SIAM J. Discret. Math.*, 22(3):920–937, 2008. doi:10.1137/060664318.
- 8 Hajo Broersma and Xueliang Li. Spanning trees with many or few colors in edge-colored graphs. *Discuss. Math. Graph Theory*, 17(2):259–269, 1997. doi:10.7151/DMGT.1053.
- 9 R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, April 1941.
- 10 Yi-Jun Chang, Martin Farach-Colton, Tsan-sheng Hsu, and Meng-Tsung Tsai. Streaming complexity of spanning tree computation. In *37th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 154 of *LIPIcs*, pages 34:1–34:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.STACS.2020.34.
- 11 Miroslav Chlebík and Janka Chlebíková. Inapproximability results for bounded variants of optimization problems. In *Fundamentals of Computation Theory, 14th International Symposium (FCT)*, volume 2751 of *Lecture Notes in Computer Science*, pages 27–38. Springer, 2003. doi:10.1007/978-3-540-45077-1\_4.

- 12 Moti Frances and Ami Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997. doi:10.1007/S002240000044.
- 13 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- 14 Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Efficient approximation algorithms for the hamming center problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 905–906. ACM/SIAM, 1999. URL: <http://dl.acm.org/citation.cfm?id=314500.315081>.
- 15 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.
- 16 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/JCSS.2000.1727.
- 17 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/JCSS.2001.1774.
- 18 David S. Johnson and Mario Szegedy. What are the least tractable instances of max independent set? In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '99*, pages 927–928, USA, 1999. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=314500.315093>.
- 19 Daniel J. Kleitman and Douglas B. West. Spanning trees with many leaves. *SIAM J. Discret. Math.*, 4(1):99–106, 1991. doi:10.1137/0404010.
- 20 J. Kevin Lanctôt, Ming Li, Bin Ma, Shaojiu Wang, and Louxin Zhang. Distinguishing string selection problems. *Inf. Comput.*, 185(1):41–55, 2003. doi:10.1016/S0890-5401(03)00057-9.
- 21 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002. doi:10.1145/506147.506150.
- 22 Petr Lisonek. New maximal two-distance sets. *J. Comb. Theory A*, 77(2):318–338, 1997. doi:10.1006/JCTA.1997.2749.
- 23 Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
- 24 Nikola Stojanovic, Piotr Berman, Deborah Gumucio, Ross C. Hardison, and Webb Miller. A linear-time algorithm for the 1-mismatch problem. In *Algorithms and Data Structures, 5th International Workshop (WADS)*, volume 1272 of *Lecture Notes in Computer Science*, pages 126–135. Springer, 1997. doi:10.1007/3-540-63307-3\_53.
- 25 Bang Ye Wu and Kun-Mao Chao. *Spanning trees and optimization problems*. Discrete mathematics and its applications. Chapman & Hall, 2004.