Treedepth Parameterized by Vertex Cover Number

Yasuaki Kobayashi[*1] **and Hisao Tamaki**[2]

1 **Kyoto University, Kyoto, Japan**
kobayashi@iip.ist.i.kyoto-u.ac.jp
2 **Meiji University, Kanagawa, Japan**
tamaki@cs.meiji.ac.jp

Abstract

To solve hard graph problems from the parameterized perspective, structural parameters have commonly been used. In particular, *vertex cover number* is frequently used in this context. In this paper, we study the problem of computing the treedepth of a given graph G. We show that there are an $O(\tau(G)^3)$ vertex kernel and an $O(4^{\tau(G)}\tau(G)n)$ time fixed-parameter algorithm for this problem, where $\tau(G)$ is the size of a minimum vertex cover of G and n is the number of vertices of G.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Fixed-parameter algorithm, Polynomial kernelization, Structural parameterization, Treedepth, Vertex cover

Digital Object Identifier 10.4230/LIPIcs.IPEC.2016.18

1 Introduction

Treedepth is an important graph invariant which attracts a lot of attentions in several communities. One of the most important results related to treedepth is the work of Nešetřil and Ossona de Mendez [27, 28]. Roughly speaking, they showed that graphs in a sparse graph classes, called graphs of bounded expansion, can be decomposed into graphs of bounded treedepth. An important consequence of this result is a linear time algorithm for deciding first-order logic properties in graphs of bounded expansion [13].

The treedepth $\mathrm{td}(G)$ of an undirected graph G is defined to be the minimum height of a rooted tree T such that G can be embedded into T in such a way that the end vertices of each edge in G has an ancestor-descendant relationship in T. A formal definition of treedepth is given in Section 2. Treedepth has been studied in literature with different names such as vertex ranking numbers [4] and the minimum height of elimination trees [29]. This invariant has applications in solving linear systems [25] and VLSI layouts [24, 31]. Moreover, treedepth has a deep relation with other well-known graph invariants *treewidth* and *pathwidth*. Let $\mathrm{tw}(G)$ and $\mathrm{pw}(G)$ be the treewidth and the pathwidth of a graph G, respectively. Then, we have $\mathrm{tw}(G) \leq \mathrm{pw}(G) \leq \mathrm{td}(G) - 1 = O(\mathrm{tw}(G) \log n)$ [28], where n is the number of vertices of G. When these three invariants are bounded, we can develop efficient algorithms for many graph problems. More precisely, if the one of the above three invariants of the input graph is at most k, many NP-hard graph problems are *fixed-parameter tractable*, that is, there is an algorithm (called a *fixed-parameter algorithm*) that solves the target problem

* Y. K. was supported by JST, CREST.

in $f(k)n^{O(1)}$ time, where n is the number of vertices and the function f depends only on k (see, for example [3]). On the other hand, Gutin *et al.* [21] recently showed that the mixed Chinese postman problem is W[1]-hard parameterized by the pathwidth of the input graph and is fixed-parameter tractable parameterized by its treedepth.

Based on the above facts, it is natural to seek an efficient algorithm for computing the treedepth of graphs. Unfortunately, the problem of computing treedepth is known to be NP-hard [29] even when the input graphs are restricted to bipartite graphs [4] or chordal graphs [11]. On the other hand, there are polynomial time algorithms for some classes of graphs [1, 10]. This problem is studied from the perspective of parameterized complexity. In this context, we are asked whether the treedepth of the input graph is at most k. Since treedepth is monotone under taking minor operations, this problem is fixed-parameter tractable when k is given as a parameter. This follows from the celebrated work of Robertson and Seymour which proves, for every minor closed graph class $\mathcal{G}$, there is a fixed-parameter algorithm that deciding whether the input graph belongs to $\mathcal{G}$ in $f(\mathcal{G})n^{O(1)}$ time. Recently, Reidl *et al.* [30] gave an algorithm for the problem whose running time is $2^{O(k^2)}n$. Their algorithm in fact runs in time $2^{O(kt)}n$, where t is the treewidth of the input graph.

As we have already noted, many graph problems can be efficiently solved on graphs of bounded treewidth. However, there are some exceptions. For example, the Bandwidth problem is known to be NP-hard for graphs of pathwidth at most three [26]. Motivated by this hardness result [26], Fellows *et al.* [16] showed that the Bandwidth problem is fixed-parameter tractable when parameterized by the size $\tau(G)$ of a *minimum vertex cover* of the input graph G. Since, for every graph G with n vertices, $\mathrm{tw}(G) \leq \tau(G) = O(\mathrm{tw}(G) \log n)$, the parameterization of the size of a minimum vertex cover to the input graph is more restricted than that of treewidth. Therefore, this parameterization were often used to develop algorithms for various hard graph problems [9, 15, 18]. In particular, Chapelle *et al.* [8] gave algorithms for computing treewidth and pathwidth whose running time is $3^{\tau(G)}n^{O(1)}$. Although this parameterization is rather restrictive, their algorithms improve the running time of the best known exact exponential algorithms for treewidth [19, 20] and for pathwidth [23] on bipartite graphs. Moreover, this parameterization is used in the context of kernelizations. Here, a *kernelization* is a polynomial time algorithm that, given a pair of an instance I and a parameter k, computes a pair (I', k') such that (I, k) is a YES-instance if and only if (I', k') is a YES-instance for the same problem and the size of I' and k' are upper bounded by some function in k. An output of a kernelization is called a *kernel*. A kernelization is *polynomial* if the size of I' is upper bounded by a polynomial in k. Bodlaender *et al.* [7, 6] showed that the Treewidth and Pathwidth problem admit polynomial kernelizations when parameterized by the size of a minimum vertex cover, in contrast to the lower bound results of polynomial kernelizations [5, 12]. when parameterized by the solution size. Subsequently, Jansen [22] improved the size of kernel to $O(|\tau(G)|^2)$ for treewidth.

In this paper, we give counterparts for treedepth to the results of kernelizations [7, 6] and fixed-parameter algorithms [8] for treewidth and pathwidth. Our results are as follows.

▶ **Theorem 1.** *There is a polynomial time algorithm that, given a graph G, a vertex cover C of G, and an integer k, computes a graph H with $|V(H)| = O(|C|^3)$ such that the treedepth of G is at most k if and only if that of H is at most k. Moreover, $V(H) \cap C$ is a vertex cover of H.*

▶ **Theorem 2.** *The treedepth of G can be computed in $O(4^{\tau(G)}\tau(G)n)$ time, where $\tau(G)$ is the size of a minimum vertex cover of G and n is the number of vertices of G.*

Theorem 1 implies together with a well-known 2-approximation algorithm for vertex cover that the treedepth problem admits a kernel with $O(\tau(G)^3)$ vertices. Let us note that, for the problem deciding whether $\mathrm{td}(G) \leq k$ with parameter k, there is no polynomial kernelization [5, 12] unless NP $\subseteq$ coNP/poly and the running time of the fastest known fixed-parameter algorithm is $2^{O(k^2)}n$ time [30]. Our kernelization and algorithm are useful for the case $\tau(G) = td(G)^{O(1)}$ and $\tau(G) = o(\mathrm{td}(G)^2)$, respectively. In contrast to treewidth and pathwidth, our result does not improve the running time of the best known exact exponential algorithm for treedepth [17] even on bipartite graphs. However, we believe that our approach is relevant for improving the bipartite case.

The technique behind our algorithm in Theorem 2 is as follows. We are given a graph G and a vertex cover C of G. Our algorithm constructs an *optimal elimination tree*, defined in Section 2, of G by a bottom-up dynamic programming. To this end, we need to define subproblems. The first attempt to define subproblems is that for each $X \subseteq C$, construct an optimal elimination tree of an induced subgraph H of G with $V(H) \cap C = X$. However, this strategy does not work since we cannot know which vertex in the independent set $V(G) \setminus C$ is in $V(H) \setminus X$. The second attempt is that for each $X \subseteq C$ and each $P \subseteq C \setminus X$, construct an optimal elimination tree of H such that $V(H) \cap C = X$ and every vertex in P is committed to be an ancestor of the vertices of H in an optimal elimination tree of the whole graph G. Using this pair X and P, we can identify the vertices of H and therefore a subproblem on (X, P) for each $X \subseteq C$ and $P \subseteq C \setminus X$ is well-defined. For each subproblem (X, P), the remaining task is to compute an optimal elimination tree T of a graph corresponding to (X, P) from optimal elimination trees of graphs corresponding to smaller subproblems (X', P') for $X' \subset X$ and $P' \subseteq C \setminus X'$. To do this, we will exploit some nontrivial property of an optimal elimination tree (See Section 4).

This paper is organized as follows. The next section describes some notations and terminologies we use. In Section 3, we design a polynomial kernelization for proving Theorem 1. In Section 4, we show Theorem 2 by giving a fixed-parameter algorithm for treedepth. Finally, in Section 5, we conclude this paper.

2 Preliminaries

For an undirected graph G, $V(G)$ denotes the set of vertices of G and $E(G)$ denotes the set of edges of G. For $v \in V(G)$, the set of neighbors of v is denoted by $N_G(v)$. For $X \subseteq V(G)$, we set $N_G(X) = \bigcup_{x \in X} N_G(x) \setminus X$. We may drop the reference to G when it is clear from the context. For disjoint sets $X, Y \subseteq V(G)$, we use $E(X, Y)$ to denote the set of all edges with one end in X and the other end in Y. The induced subgraph by $X \subseteq V(G)$ of G is denoted by $G[X]$. For two graphs G and H, $H \subseteq G$ means that H is a subgraph of G.

Let T be a rooted tree. For $v \in V(T)$, the subtree rooted at v is denoted by T_v and the unique path between v and the root of T is denoted by P_v. A *branching point* of T is a vertex that has at least two children in T. For two vertices $u, v \in V(T)$, we say u is an *ancestor* of v (v is a *descendant* of u) if $u \in V(P_v)$. A set of disjoint rooted trees is called a *rooted forest*. For a rooted forest F, we use $V(F)$ to denote $\bigcup_{T \in F} V(T)$. The *depth* of v in T is defined by the number of vertices of P_v. The *height* of T is the maximum depth among the vertices in T and the height of rooted forest F is the maximum depth of a rooted tree that belongs to F. The height of rooted tree T and rooted forest F are denoted by $\mathrm{height}(T)$ and $\mathrm{height}(F)$, respectively.

Let F be a rooted forest. The *closure* of F is a graph with vertex set $V(F)$ such that the graph contains an edge $\{u, v\}$ if and only if u is an ancestor of v in F or vice versa. We

denote by $\mathrm{clos}(F)$ the closure of F. In particular, when F consists of a single rooted tree T, we may write $\mathrm{clos}(T)$ instead of $\mathrm{clos}(\{T\})$. For a (not necessary connected) graph G, the *treedepth* $\mathrm{td}(G)$ of G is the minimum integer k such that there is a rooted forest F with $\mathrm{height}(F) = k$ and $G \subseteq \mathrm{clos}(F)$.

As mentioned before, the notion of treedepth has equivalent definitions. In this paper, we frequently use the notion of *elimination trees*. For a rooted forest F and a vertex v not in $V(F)$, we use the notation $F \circ v$ to denote the rooted tree with root v obtained from F by adding an edge between v and the root of T for each $T \in F$. In particular, if F consists of a single rooted tree T, we write $T \circ v$ instead of $\{T\} \circ v$.

▶ **Definition 3.** An *elimination tree* of a connected graph G is recursively defined as follows.

1. If G consists of a single vertex then the elimination tree of G is itself.
2. Otherwise, choose $v \in V(G)$ arbitrary. Let $F = \{T_1, T_2, \ldots, T_t\}$ be elimination trees of the connected components of $G[V(G) \setminus \{v\}]$. Then, $F \circ v$ is an elimination tree of G.

We say that an elimination tree T of G is *optimal* if there is no elimination trees of G whose height is smaller than the height of T. It is easy to see that, for every elimination tree T of G, the closure of T contains G as a subgraph. The following proposition shows that the converse also holds when G is connected.

▶ **Proposition 4** ([27]). *Let G be a connected graph and let T be a rooted tree. Then, T is an elimination tree of G if and only if $G \subseteq \mathrm{clos}(T)$. In particular, the height of an optimal elimination tree of G is $\mathrm{td}(G)$.*

When we refer to an elimination tree T of G, we use the fact $G \subseteq \mathrm{clos}(T)$ without the reference to Proposition 4 and vice versa.

3 Kernelization

The aim of this section is to develop a polynomial kernelization for treedepth for proving Theorem 1. The technique that we use is very similar to the kernelization result for pathwidth [6].

Fix a vertex cover C of G. Let $I = V(G) \setminus C$ and let k be a positive integer. We consider the problem of deciding whether $\mathrm{td}(G) \leq k$. We assume that C is not empty since otherwise the problem is trivial.

We will describle three reduction rules. The following rule trivially does not change the treedepth of G.

▶ **Rule 5.** *Let $u \in I$ be an isolated vertex. Then, delete u from G.*

▶ **Lemma 6.** *Suppose $\mathrm{td}(G) \leq k$. Let u, v be vertices with $|N_G(u) \cap N_G(v)| \geq k$ and let G' be the graph obtained by adding an edge $\{u, v\}$ to G. Then, $\mathrm{td}(G) = \mathrm{td}(G')$.*

Proof. Since G is a subgraph of G', $\mathrm{td}(G) \leq \mathrm{td}(G')$. For the inverse direction, let T be an optimal elimination tree of G. When T_u and T_v are not disjoint, T is also an elimination tree of G' and hence $\mathrm{td}(G) \geq \mathrm{td}(G')$. Here, we assume otherwise. This assumption implies the vertices $N_G(u) \cap N_G(v)$ are common ancestors of u and v. This contradicts the fact that $\mathrm{height}(T) \leq k$ and $|N_G(u) \cap N_G(v)| \geq k$. ◀

This lemma verifies the following rule does not change the treedepth of G.

▶ **Rule 7.** *Let u, v be vertices with $|N(u) \cap N(v)| \geq k$. Suppose at least one of $u \in C$ and $v \in C$ holds. Then, add an edge $\{u, v\}$ to G.*

Let us note that C is also a vertex cover of the result of an application of Rule 7 to G.

For the next rule, we need the following observation, which is clear from the definition of closure.

▶ **Observation 8.** *Let K be a clique in G and let T be an elimination tree of G. Then, there is $v \in K$ with $K \subseteq V(P_v)$.*

A vertex v is *simplicial* in G if $N_G(v)$ forms a clique.

▶ **Lemma 9.** *Suppose* $\mathrm{td}(G) \leq k$. *Let u be a simplicial vertex such that $N(u)$ is not empty and $|N(v)| \geq k+1$ for each $v \in N(u)$. Then* $\mathrm{td}(G) = \mathrm{td}(G[V(G) \setminus \{u\}])$.

Proof. The subgraph relation proves $\mathrm{td}(G) \geq \mathrm{td}(G[V(G) \setminus \{u\}])$. In the following, we show the converse inequality. Let T be an optimal elimination tree of $G[V(G) \setminus \{u\}]$. Since $N(u)$ forms a clique in $G[V(G) \setminus \{u\}]$, by Observation 8, there is $v \in N(u)$ with $N(u) \subseteq V(P_v)$. From the assumption of this lemma, v has at least k neighbors different from u. Since $\mathrm{height}(T) \leq k$, at least one of them is a descendant of v in T. Observe that a rooted tree T' obtained from T by adding u as a child of v is an elimination tree of G. This follows from $N(u) \subseteq V(P_v)$. Moreover, since v is not a leaf in T, $\mathrm{height}(T) = \mathrm{height}(T')$ and hence $\mathrm{td}(G) \leq \mathrm{td}(G[V(G) \setminus \{u\}])$ holds. ◀

▶ **Rule 10.** *Let $u \in I$ be a simplicial vertex of G. Suppose each $v \in N(u)$ has at least $k+1$ neighbors. Then, delete u from G.*

The above two rules give us a small kernel for treedepth.

▶ **Lemma 11.** *Suppose* $\mathrm{td}(G) \leq k$, $|C| \geq k$, *and neither Rule 5, Rule 7, nor Rule 10 are applicable to G. Then, the number of vertices of G is $O(|C|^3)$.*

Proof. Let S be the set of simplicial vertices of G, let $P = S \cap I$, and let $Q = I \setminus P$. By Rule 5, each vertex in P has at least one neighbor in C. For each $u \in P$, by Rule 10, there is a vertex $v \in N(u)$ whose degree is at most k. We associate u with $v \in C$. Each vertex in P is associated with some vertex in C and at most k vertices in P are associated with each vertex in C. Hence, $|P| \leq k \cdot |C| \leq |C|^2$. Next, consider non-adjacent vertices u, v of C. Observe that u and v have at most $k-1$ common neighbors in G. This follows from Rule 7. As each vertex in Q has at least one pair of non-adjacent vertices in C, we have $|Q| \leq (k-1) \cdot |C|(|C|-1)/2 \leq |C|^3/2$. Therefore, $V(G) = |C| + |P| + |Q| \leq 3 \cdot |C|^3$. ◀

Let (G, C, k) be an instance of our decision problem. Suppose $|C| < k$. Obviously, (G, C, k) is a YES-instance. In this case, we output a constant-sized YES-instance. Suppose otherwise. We exhaustively apply Rule 7 and Rule 10 to G until both of them are not applicable to G. By Lemmas 6 and 9, the resulting graph H satisfies $\mathrm{td}(H) \leq k$ if and only if $\mathrm{td}(G) \leq k$. Moreover, $V(H) \cap C$ is a vertex cover of H. Thus, $(H, V(H) \cap C, k)$ is a valid instance. By Lemma 11, if $|V(H)| > 3 \cdot |V(H) \cap C|^3$, then (G, C, k) is a NO-instance. In this case, we output a constant-sized NO-instance. Overall, we have Theorem 1.

4 Fixed-Parameter Algorithm

The aim of this section is to develop an algorithm computing the treedepth of G whose running time is upper bounded by $O(4^{\tau(G)}\tau(G)n)$, where $n = |V(G)|$. First, we use the following algorithm to obtain a minimum vertex cover in advance.

▶ **Proposition 12** (folklore). *There is a $O(2^{\tau(G)}(n+m))$ time algorithm that finds a minimum vertex cover of a graph G, where n and m are respectively the number of vertices and edges of G.*

Note that $m = O(n \cdot \tau(G))$. In the rest of this section, fix a vertex cover C of G. We assume that C is not empty since otherwise the problem is trivial. Let $I = V(G) \setminus C$. For $X \subseteq C$, we set $I(X) = \{v \in I \mid N(v) \subseteq X\}$. We say that a rooted tree is *atomic* if it consists of a single vertex.

▶ **Definition 13.** For $X \subseteq C$ and $P \subseteq C \setminus X$, we say that a rooted forest F is *compatible* with (X, P) if the following three conditions are satisfied:
C1. $V(F) = X \cup I(X \cup P)$,
C2. $G[V(F)] \subseteq \mathrm{clos}(F)$, and
C3. every vertex in $I(P)$ forms an atomic rooted tree in F.

Note that a rooted forest that is compatible with (X, P) does not contain any vertex in P and contains every vertex in $I(P)$. We denote by $\mathrm{td}(X, P)$ the minimum height over all rooted forests that are compatible with (X, P). We say that F is *optimal for* (X, P) if F is compatible with (X, P) and $\mathrm{height}(F) = \mathrm{td}(X, P)$. Also, note that if $G[X \cup I(X \cup P)]$ is connected, there is a rooted tree T that is optimal for (X, P).

In what follows, we will give recurrences for computing $\mathrm{td}(X, P)$ for $X \subseteq C$ and $P \subseteq C \setminus X$. The algorithm evaluates those recurrences by a straight forward dynamic programming. The following lemma is the base case of our recurrences and is easy to verify.

▶ **Lemma 14.** *Let $P \subseteq C$. If $I(P)$ is not empty, then $\mathrm{td}(\emptyset, P) = 1$. Otherwise, $\mathrm{td}(\emptyset, P) = 0$.*

Let $X \subseteq C$ and $P \subseteq C \setminus X$. From now on, we consider the case $X \neq \emptyset$.

▶ **Lemma 15.** *Let $x \in X$ be arbitrary and let F be a rooted forest that is compatible with $(X \setminus \{x\}, P \cup \{x\})$. Then the rooted forest $F' := I(P) \cup (F \setminus I(P)) \circ x$ is compatible with (X, P). Here and in similar situations later, $I(P)$ is also interpreted as the set of atomic rooted trees.*

Proof. Clearly, F' satisfies condition C3 for (X, P). As $V(F) \setminus V(F') = \{x\}$, F' satisfies condition C1 for (X, P). Let $T = (F \setminus I(P)) \circ x$. Since every vertex in $N_{G[V(T)]}(x)$ is a descendant of x in T, T is an elimination tree of $G[V(T)]$. As $X \cap P = \emptyset$, we have $E(X, I(P)) = \emptyset$. Thus, condition C2 holds for (X, P) and hence the lemma follows. ◀

We say that a bipartition (Y, Z) of X is *separated* if neither Y nor Z is empty and $E(Y, Z) = \emptyset$. The following observation is easy to verify.

▶ **Observation 16.** *Let (Y, Z) be a bipartition of X. Then, $I(X \cup P)$ is partitioned into $N(Y) \cap N(Z) \cap I(X \cup P)$, $I(Y \cup P) \setminus I(P)$, $I(Z \cup P) \setminus I(P)$, and $I(P)$.*

Let (Y, Z) be a separated bipartition of X. We define the rooted forest $F_{Y,Z}$ from F_Y and F_Z, where F_Y and F_Z are rooted forests that are compatible with (Y, P) and (Z, P), respectively, as follows. Let $v_0, v_1, \ldots v_p$ be the vertices in $N(Y) \cap N(Z) \cap I(X \cup P)$ with arbitrary order. Note that F_Y and F_Z may share the set of vertices $I(P)$. Set $F_0 := (F_Y \cup F_Z) \setminus I(P)$ and, for each $0 \leq i \leq p$, set $F_{i+1} := F_i \circ v_i$. Finally, set $F_{Y,Z} := F_{p+1} \cup I(P)$. By the above definition, if $N(Y) \cap N(Z) \cap I(X \cup P)$ is empty, $F_{Y,Z}$ is indeed $F_Y \cup F_Z$.

▶ **Lemma 17.** *Let (Y, Z) be a separated bipartition of X and let F_Y and F_Z be rooted forests that are compatible with (Y, P) and (Z, P), respectively. Then, $F_{Y,Z}$ is compatible with (X, P).*

Proof. We show that $F_{Y,Z}$ satisfies the three conditions of compatibility with (X, P).

Clearly, $F_{Y,Z}$ satisfies condition C3. By the construction of G, $V(F_{Y,Z})$ is composed of $(V(F_Y) \cup V(F_Z)) \setminus I(P)$, $N(Y) \cap N(Z) \cap I(X \cup P)$, and $I(P)$. Since $V(F_Y) \cap I = I(Y \cup P)$ and $V(F_Z) \cap I = I(Z \cup P)$, by Observation 16, we have $V(F_{Y,Z}) \cap I = I(X \cup P)$. Recall that $Y \cup Z = X$. Thus, we have $V(F_{Y,Z}) = X \cup I(X \cup P)$, that is, $F_{Y,Z}$ satisfies condition C1. To show the condition C2, it is enough to show that for each edge $\{u, v\}$ in $G[X \cup I(X \cup P)]$, u is an ancestor of v or vice versa in $F_{Y,Z}$. Since X is a vertex cover of $G[X \cup I(X \cup P)]$, at least one of u and v is in X. Assume without loss of generality u is in Y. Since F_Y is compatible with (Y, P), if $\{u, v\} \subseteq V(F_Y)$, we are done. Otherwise, as $u \notin V(F_Z)$, u must be in $N(Y) \cap N(Z) \cap I(X \cup P)$, and hence u is an ancestor of v. This finishes the proof. ◀

Let T be a rooted tree that is compatible with (X, P) and let $v \in V(T)$ be a maximum depth vertex such that every vertex in $V(P_v) \setminus \{v\}$ is not a branching point of T. That is v is either a leaf or the minimum depth branching point of T. Observe that a rooted tree that is obtained by swapping the positions of an arbitrary pair of vertices in $V(P_v)$ is also compatible with (X, P). This implies the following observation.

▶ **Observation 18.** *Let T be a rooted tree that is compatible with (X, P). Let $v \in V(T)$ be a maximum depth vertex such that every vertex in $u \in V(P_v) \setminus \{v\}$ is not a branching point of T. Suppose there is a vertex $x \in X \cap V(P_v)$. Then, there is a rooted tree T' with root x that is compatible with (X, P) whose depth is* $\mathrm{height}(X, P)$.

Observation 18 implies that if there is no rooted tree T that is optimal for (X, P) whose root belongs to X, then there is the minimum depth branching point v in T such that $V(P_v) \subseteq I(X \cup P)$ for every rooted tree T that is optimal for (X, P).

The following lemma plays a key role for the construction of optimal elimination trees. To this end, we need some operation on rooted trees. A *vertex removal* of a non-root vertex u (or u is *removed*) from a rooted tree T results a rooted tree that is obtained from T by deleting u and adding an edge between the parent of u and each child of u. When u is the root of T and has only one child, the vertex removal of u is simply deleting u from T.

▶ **Lemma 19.** *Assume that $G[X \cup I(X \cup P)]$ is connected. Then, there exists an optimal rooted tree T for (X, P) such that either (1) the root x of T is in X and every vertex in $I(P \cup \{x\})$ is a child of x or (2) there is a separated bipartition (Y, Z) of X such that $V(P_v) = N(Y) \cap N(Z) \cap I(X \cup P)$, where v is the minimum depth branching point in T.*

Proof. Let T be an optimal rooted tree for (X, P). From Observation 18, we can assume that either the root of T is in X or there is the minimum depth branching point v of T such that $V(P_v) \subseteq I(X \cup P)$.

Suppose first that the root of T is $x \in X$. Let u be the vertex in $I(P \cup \{x\})$ that is not a child of x. Since u has exactly one neighbor x in $G[X \cup I(X \cup P)]$, the rooted tree obtained from T by removing u and adding u as a child of x is also an elimination tree of height not larger than T. The repeated applications of the above argument show that every vertex in $I(P \cup \{x\})$ is a child of x.

Suppose otherwise. Let T be an arbitrary optimal rooted tree for (X, P). Recall that we have $V(P_v) \subseteq I(X \cup P)$ in this case. We construct a separated bipartition (Y, Z) of X as follows. Let v be the minimum depth branching point of T and let W be the set of children of v. Observe that for each $w \in W$, $V(T_w)$ contains at least one vertex of X as otherwise $G[X \cup I(X \cup P)]$ is not connected, which contradicts our assumption. Choose a non-empty proper subset $W' \subset W$. Let $Y = \bigcup_{w \in W'} (V(T_w) \cap X)$ and let $Z = X \setminus Y$. Clearly, (Y, Z) is a separated bipartition of X. This implies that every vertex in $N(Y) \cap N(Z) \cap I(X \cup P)$

belongs to $V(P_v)$. Hence, we have $V(P_v) \supseteq N(Y) \cap N(Z) \cap I(X \cup P)$. In the following we will show that, under an appropriate choice of T, this separated bipartition (Y, Z) satisfies $V(P_v) \subseteq N(Y) \cap N(Z) \cap I(X \cup P)$.

We choose an optimal rooted tree T for (X, P) and a separated bipartition (Y, Z) of X in such a way as to minimize the number of vertices in $V(P_v) \setminus (N(Y) \cap N(Z) \cap I(X \cup P))$. Let $Q = V(P_v) \setminus (N(Y) \cap N(Z) \cap I(X \cup P))$ be the set of vertices in $V(P_v)$ each of which has a neighbor neither in Y nor in Z. We claim that Q is empty and hence such T and (Y, Z) satisfy $V(P_v) \subseteq N(Y) \cap N(Z) \cap I(X \cup P)$. Suppose, for contradiction, let $u \in Q$. As $G[X \cup I(X \cup P)]$ is connected, $V(P_v)$ must have at least two vertices and hence u is removable from T (either u is not the root of T or u is the root which has only one child). We assume, without loss of generality, u has a neighbor in Z. Let W be the set of children w of v with $V(T_w) \cap Y \neq \emptyset$ and let $F_W = \{T_w : w \in W\}$. We construct another rooted tree from T as follows. Remove u and delete the vertices of F_W from T. Then, combine T with $F_W \circ u$ by adding an edge between w and u. It is easy to see that the result of the above operations is an elimination tree of $G[X \cup I(X \cup P)]$ and the height is not larger than the original T. Moreover, the size of Q in the result is strictly smaller than the original one. This is contracting to the minimality of Q. Hence, the lemma follows. ◀

▶ **Lemma 20.** *Let $X \subseteq C$ with $X \neq \emptyset$ and let $P \subseteq C \setminus X$. Then, $\mathrm{td}(X, P)$ is equal to the smaller value of*

$$\min_{x \in X} \mathrm{td}(X \setminus \{x\}, P \cup \{x\}) + 1 \tag{1}$$

and

$$\min_{Y,Z} \max(\mathrm{td}(Y, P), \mathrm{td}(Z, P)) + |N(Y) \cap N(Z) \cap I(X \cup P)|, \tag{2}$$

where, in expression (2), the minimum is taken among all separated bipartitions (Y, Z) of X and if there is no separated bipartition of X then the value of (2) is defined to be infinity.

Proof. First, we show that $\mathrm{td}(X, P)$ is not smaller than the value of both (1) and (2). Consider the case where the value of (1) is not smaller than that of (2). Let x be a vertex in C that attains the minimum of expression (1) and let F be a minimum height rooted forest among all rooted forests that are compatible with $(X \setminus \{x\}, P \cup \{x\})$. By Lemma 15, $F \circ x$ is compatible with (X, P). Therefore, we have

$$\mathrm{td}(X, P) \leq \mathrm{height}(F \circ x) = \mathrm{height}(F) + 1 = \min_{x \in X} \mathrm{td}(X \setminus \{x\}, P \cup \{x\}) + 1.$$

Suppose the other case: the value of (1) is larger than that of (2). In this case, there is a separated bipartition (Y, Z) of X that attains the minimum of expression (2). Let F_Y and F_Z be minimum height rooted forests among all rooted forests that are compatible with (Y, P) and (Z, P), respectively. Let $F_{Y,Z}$ be a rooted forest constructed from F_Y and F_Z as in Lemma 17. Since $F_{Y,Z}$ is compatible with (X, P), we have

$$\begin{aligned} \mathrm{td}(X, P) &\leq \mathrm{height}(F_{Y,Z}) \\ &= \max(\mathrm{height}(F_Y), \mathrm{height}(F_Z)) + |N(Y) \cap N(Z) \cap I(X \cup P)| \\ &= \max(\mathrm{td}(Y, P), \mathrm{td}(Z, P)) + |N(Y) \cap N(Z) \cap I(X \cup P)| \end{aligned}$$

Hence, $\mathrm{td}(X, P)$ is not smaller than the value of both (1) and (2).

Next, we show the other direction. We distinguish the two cases: $G[X \cup I(X \cup P)]$ is connected or not.

Consider the case where $G[X \cup I(X \cup P)]$ is connected. Let T be an elimination tree described in Lemma 19. Then, either the root of T is in X or there exists a separated bipartition (Y, Z) of X with $V(P_v) = N(Y) \cap N(Z) \cap I(X \cup P)$, where v is the minimum depth branching point in T.

Case 1: The root x of T is in X. Let F' be a rooted forest which is obtained from T by deleting the root x. To prove our desired inequality $\mathrm{height}(T) \geq \mathrm{td}(X \setminus \{x\}, P \cup \{x\}) + 1$, we will show that F' is compatible with $(X \setminus \{x\}, P \cup \{x\})$. It is easy to verify that $V(F') \cap C = X \setminus \{x\}$, $V(F') \cap I = I((X \setminus \{x\}) \cup (P \cup \{x\}))$, and $G[V(F')] \subseteq \mathrm{clos}(F')$. By Lemma 19, each vertex $v \in I(P \cup \{x\})$ is a child of x, that is, $I(P \cup \{x\})$ is the set of atomic rooted trees in F'. Hence, F' is compatible with $(X \setminus \{x\}, P \cup \{x\})$.

Case 2: There is a separated bipartition (Y, Z) of X with $V(P_v) = N(Y) \cap N(Z) \cap I(X \cup P)$, where v is the minimum depth branching point of T. Let F' be the rooted forest obtained from T by deleting all the vertices in $V(P_v)$. We claim that F' is partitioned into two rooted forests F_Y and F_Z such that F_Y and F_Z are compatible with (Y, P) and (Z, P), respectively. Note that this claim establishes the inequality $\mathrm{height}(T) \geq \max(\mathrm{td}(Y, P), \mathrm{td}(Z, P)) + |N(Y) \cap N(Z) \cap I(X \cup P)|$. Consider the two rooted forests $F_Y = \{T' \in F' : V(T') \cap Y \neq \emptyset\}$ and $F_Z = \{T' \in F' : V(T') \cap Z \neq \emptyset\}$. Observe that F_Y and F_Z are disjoint since (Y, Z) is separated and every common neighbor of Y and Z must be in $V(P_v)$. As $V(P_v) \subseteq I$, for each child w of v, T_w contains at least one vertex of X. Thus, (F_Y, F_Z) is a bipartition of F'. It is easy to see that $V(F_Y) \cap C = Y$ and $G[V(F_Y)] \subseteq \mathrm{clos}(F_Y)$. Recall that $I(P)$ is empty. By Observation 16, $I(X \cup P)$ is partitioned into $I(Y \cup P)$, $I(Z \cup P)$, and $N(Y) \cap N(Z) \cap I(X \cup P)$. Since every vertex in $I(Y \cup P)$ has a neighbor in Y, $V(F_Y) \cap I = I(Y \cup P)$. Hence, F_Y is compatible with (Y, P). A similar argument shows that F_Z is compatible with (Z, P). Therefore, we have the claim.

Finally, we consider the case where $G[X \cup I(X \cup P)]$ is not connected. Let $\mathcal{C}$ be the set of connected components of $G[X \cup I(X \cup P)]$. We apply Lemma 19 to each component in $\mathcal{C} \setminus I(P)$ and obtain a rooted forest F from $\mathcal{C} \setminus I(P)$. Note that every component in $\mathcal{C} \setminus I(P)$ has at least one vertex of X. If F consists of a single rooted tree, we can apply the same argument with the connected case to the unique rooted tree $T \in F$. Otherwise, the connected components of F naturally induce some separated bipartition of X, which satisfies the desired inequality since $N(Y) \cap N(Z) \cap I(X \cup P)$ is empty. Hence, we have the lemma. ◀

By Lemmas 14 and 20, for $X \subseteq C$ and $P \subseteq C \setminus X$, we can compute $\mathrm{td}(X, P)$ via a standard dynamic programming. When the values $\mathrm{td}(X', P')$ are stored in a table for $X' \subset X$ and $P' \subseteq C \setminus X'$, the value $\mathrm{td}(X, P)$ is computed in $O(2^{|X|}|X|n)$ time. Hence, the running time of our dynamic programming is

$$\begin{aligned}\sum_{X \subseteq C} \sum_{P \subseteq C \setminus X} O(2^{|X|}|X|n) &= \sum_{X \subseteq C} 2^{|C|-|X|} \cdot O(2^{|X|}|X|n) \\ &= \sum_{X \subseteq C} O(2^{|C|}|C|n) \\ &= O(4^{|C|}|C|n).\end{aligned}$$

5 Conclusion

In this paper, we have given a polynomial kernelization and a fixed-parameter algorithm for treedepth when the minimum size of vertex cover of the input graph is parameterized.

The main open questions are improving on the size of kernel and the running time of fixed-parameter algorithm. Jansen [22] showed that there is a kernel for treewidth whose size is quadratic with respect to vertex cover number, which improves the previous result of polynomial kernelization due to Bodlaender *et al.* [7]. He concluded in his paper that the key lemma to the kernelization does not work for pathwidth. This obstacle also appears in the case of treedepth. Chapelle *et al.* [8] improved the running time of their algorithm for treewidth using the fast subset convolution technique due to Björklund *et al.* [2]. One may expect that the bottleneck of our running time (the computation of expression (2) of Lemma 20) can be broken by the fast subset convolution technique. However, this technique does not seem to apply to our fixed-parameter algorithm directly.

Finally, extending our result to more general cases would be interesting. One of such extensions is to use another structural parameterization. Feedback vertex set number would be a good candidate for this line. Another extension is to consider the problem on directed graphs. *Cycle rank* [14] is known to be a directed version of treedepth. To the best of our knowledge, no fixed-parameter tractability result for cycle rank is known.

Acknowledgements. We are grateful to anonymous referees for the suggestions for improving the presentation of the paper.

References

1 B. Aspvall and P. Heggernes. Finding minimum height elimination trees for interval graphs in polynomial time. *BIT Numerical Mathematics*, 34(4):484–509, 1994.

2 A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets möbius: Fast subset convolution. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC'07, pages 67–74. ACM, 2007.

3 H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybern.*, 11(1-2):1–21, 1993.

4 H. L. Bodlaender, J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, and Z. Tuza. Rankings of graphs. *SIAM J. Discrete Math.*, 11(1):168–181, 1998.

5 H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

6 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Kernel bounds for structural parameterizations of pathwidth. In *Algorithm Theory – SWAT 2012 – 13th Scandinavian Symposium and Workshops, Helsinki, Finland, July 4-6, 2012. Proceedings*, pages 352–363, 2012.

7 H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. *SIAM J. Discrete Math.*, 27(4):2108–2142, 2013.

8 M. Chapelle, M. Liedloff, I. Todinca, and Y. Villanger. Treewidth and pathwidth parameterized by the vertex cover number. In *Algorithms and Data Structures – 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, pages 232–243, 2013.

9 M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. On cutwidth parameterized by vertex cover. *Algorithmica*, 68(4):940–953, 2014.

10 J. S. Deogun, T. Kloks, D. Kratsch, and H. Müller. On the vertex ranking problem for trapezoid, circular-arc and other graphs. *Discrete Applied Mathematics*, 98(1-2):39–63, 1999.

11 D. Dereniowski and A. Nadolski. Vertex rankings of chordal graphs and weighted trees. *Inf. Process. Lett.*, 98(3):96–100, 2006.

12 Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015.

13 Z. Dvořák, D. Král, and R. Thomas. Testing first-order properties for subclasses of sparse graphs. *J. ACM*, 60(5):36, 2013.

14 L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10(4):385–397, 12 1963.

15 M. R. Fellows, D. Hermelin, F. A. Rosamond, and H. Shachnai. Tractable parameterizations for the minimum linear arrangement problem. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 457–468, 2013.

16 M. R. Fellows, D. Lokshtanov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *Algorithms and Computation, 19th International Symposium, ISAAC 2008, Gold Coast, Australia, December 15-17, 2008. Proceedings*, pages 294–305, 2008.

17 F. V. Fomin, A. C. Giannopoulou, and M. Pilipczuk. Computing tree-depth faster than 2^n. *Algorithmica*, 73(1):202–216, 2015.

18 F. V. Fomin, B. M. P. Jansen, and M. Pilipczuk. Preprocessing subgraph and minor problems: When does a small vertex cover help? *J. Comput. Syst. Sci.*, 80(2):468–495, 2014.

19 F. V. Fomin and Y. Villanger. Finding induced subgraphs via minimal triangulations. In *27th International Symposium on Theoretical Aspects of Computer Science, STACS 2010, March 4-6, 2010, Nancy, France*, pages 383–394, 2010.

20 F. V. Fomin and Y. Villanger. Treewidth computation and extremal combinatorics. *Combinatorica*, 32(3):289–308, 2012.

21 G. Gutin, M. Jones, and M. Wahlström. Structural parameterizations of the mixed chinese postman problem. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 668–679, 2015.

22 B. M. P. Jansen. On sparsification for computing treewidth. *Algorithmica*, 71(3):605–635, 2015.

23 K. Kitsunai, Y. Kobayashi, K. Komuro, H. Tamaki, and Toshihiro Tano. Computing directed pathwidth in $o(1.89^n)$ time. *Algorithmica*, 75(1):138–157, 2016.

24 C. E. Leiserson. Area-efficient graph layouts. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 270–281, Oct 1980.

25 J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM J. Matrix Anal. Appl.*, 11(1):134–172, January 1990.

26 B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM J. Algebraic Discrete Methods*, 7(4):505–512, October 1986.

27 J. Nešetřil and P. O. de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *Eur. J. Comb.*, 27(6):1022–1041, 2006.

28 J. Nešetřil and P. O. de Mendez. *Sparsity – Graphs, Structures, and Algorithms*, volume 28. Springer, 2012.

29 A. Pothen. The complexity of optimal elimination trees. Technical Report CS-88-13, Pennsylvania State University, 1988.

30 F. Reidl, P. Rossmanith, F. Sanchez Villaamil, and S. Sikdar. A faster parameterized algorithm for treedepth. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 931–942, 2014.

31 A. Sen, H. Deng, and S. Guha. On a graph partition problem with application to vlsi layout. *Inf. Process. Lett.*, 43(2):87–94, August 1992.