Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 155 (2007) 1275 – 1293
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Tree-edges deletion problems with bounded diameter obstruction sets

Dekel Tsur

Department of Computer Science, Ben-Gurion University of the Negev, Israel

Received 28 November 2004; received in revised form 28 September 2006; accepted 24 October 2006
Available online 20 December 2006

Abstract

We study the following problem: given a tree G and a finite set of trees $\mathscr{H}$, find a subset O of the edges of G such that $G - O$ does not contain a subtree isomorphic to a tree from $\mathscr{H}$, and O has minimum cardinality. We give sharp boundaries on the tractability of this problem: the problem is polynomial when all the trees in $\mathscr{H}$ have diameter at most 5, while it is NP-hard when all the trees in $\mathscr{H}$ have diameter at most 6. We also show that the problem is polynomial when every tree in $\mathscr{H}$ has at most one vertex with degree more than 2, while it is NP-hard when the trees in $\mathscr{H}$ can have two such vertices.

The polynomial-time algorithms use a variation of a known technique for solving graph problems. While the standard technique is based on defining an equivalence relation on graphs, we define a quasiorder. This new variation might be useful for giving more efficient algorithm for other graph problems.

Keywords: Graph algorithms; Subgraph isomorphism

1. Introduction

Many graph problems can be formulated as a *maximum subgraph problem* with respect to some graph property P: given a graph G, find a subgraph of G that satisfies P and has maximum number of edges. Such problem can also be formulated as a deletion problem: given a graph G, find a subset O of the edges of G such that $G - O$ satisfies P and O has minimum cardinality among all such sets.

A graph property P is *hereditary* if for every graph satisfying P, all its vertex-induced subgraphs also satisfy P. Any hereditary graph property P can be characterized by the *obstruction set* $\mathscr{H}_P$ of all minimal graphs that do not satisfy P: a graph satisfies P if and only if it does not contain any graph from $\mathscr{H}_P$ as an induced subgraph.

Many maximum subgraph problems are NP-hard (for example, Maximum Clique and Longest Path). However, when restricting the input graph, some problems become polynomial. In particular, it has been shown that for every hereditary property P with a finite obstruction set, the corresponding maximum subgraph problem can be solved in linear time on series–parallel graphs [16]. This result has been extended to the family of graphs with bounded treewidth and to a larger family of properties [1,2,4–6,10,14].

A major problem with the above algorithms is that the constants hidden in the time complexity can be extremely large for some graph properties. In order to evaluate the effect of the property P on the time complexity, we shall consider

E-mail address: dekelts@cs.bgu.ac.il.

0166-218X/$ - see front matter
doi:10.1016/j.dam.2006.10.016

the property P as part of the input. We will deal with hereditary properties, so define the *edges deletion problem* as follows: given a graph G and an *obstruction set* $\mathscr{H}$, find an edge set O with minimum size such that $G - O$ does not contain an induced subgraph isomorphic to any graph H from $\mathscr{H}$. The edges deletion problem is NP-hard. We will be interested in special cases of the problem that can be solved in polynomial time in the input size. We note that the approach of making P part of the input resembles the research on fixed parameter tractability (cf. [7]).

In this work we concentrate on the edge deletion problem when all the input graphs are trees. We call this problem the *tree-edges deletion problem* (TEDP). Using the approach of Takamizawa et al. [16], the TEDP can be solved in $2^{2^{O(k)}}n$ time, where n is the number of vertices in the graph G and k is the total number of vertices in the graphs in $\mathscr{H}$. Shamir and Tsur [15] gave a $2^{O(k^2/\log k)}n$-time algorithm for TEDP.

In this paper we give sharp boundaries on the tractability of TEDP. Let l-TEDP denote the TEDP restricted to instances in which all the trees in $\mathscr{H}$ have diameter at most l. We show that 5-TEDP can be solved in polynomial time while 6-TEDP is NP-hard. Furthermore, let TEDP$_l$ denote the TEDP restricted to instances in which each tree in $\mathscr{H}$ has at most l vertices with degree more than two. We show that TEDP$_1$ can be solved in polynomial time, while TEDP$_2$ is NP-hard.

When dealing with approximation, one can consider the maximization version of TEDP, in which the objective function is the number of edges remaining in $G - O$. This problem is called the *maximum subforest problem* (MSP). MSP and TEDP are equivalent when seeking an optimal solution. However, their approximability is different: while MSP has a polynomial-time approximation scheme [15], we show that TEDP is hard to approximate within factor $c \log k$ for some constant c. This result holds also for 6-TEDP and TEDP$_2$.

Our approach for solving 5-TEDP and TEDP$_1$ is based on the approach used in previous work: a dynamic programming algorithm computes partial solutions for subtrees of the input graph G. A key ingredient of the algorithm is the definition of an equivalence relation according to the obstruction set $\mathscr{H}$. For each subtree G' of G processed by the algorithm, the algorithm finds partial solutions of G' from each equivalence class. Consequently, the time complexity of the algorithm depends on the number of equivalence classes. In our approach, we define a *quasiorder* instead of an equivalence class, and the time complexity of our algorithm depends on the "width" of the quasiorder. While the equivalence relation approach yields an exponential time algorithm to 5-TEDP (as the equivalent relation has exponential number of equivalence classes), our approach gives a polynomial-time algorithm.

We note that we do not have a direct application for the TDEP. However, a special case of TEDP can be used as a heuristic for solving the problem of finding the maximum interval subgraph of a bipartite graph, which has an application in computational biology [17,18]. This special case of TEDP is when $\mathscr{H}$ is fixed and consists of the minimum tree which is not an interval graph (this tree consists of a center vertex from which three paths of length 2 start). Our techniques can be used to obtain a linear time algorithm for this special case of TEDP, which is more efficient than previous linear time algorithms for the problem.

Finally, note that the edge deletion problem is a generalization of the *subgraph isomorphism problem*: given two graphs G and H, decide whether there is a subgraph of G that is isomorphic to H. The subgraph isomorphism problem is clearly NP-hard. It remains NP-hard even if G is a tree and H is a forest, or if G is a general graph and H is a tree [8]. The subgraph isomorphism problem is solvable in polynomial time if G and H are trees [12], or if G and H have treewidth at most p for some fixed p and H is p-connected [11].

The rest of the paper is organized as follows: Section 2 contains definitions. In Section 3 we give a general framework for algorithms for TEDP. In Section 4 we define a simple problem, called the set deletion problem, and give an algorithm that solves this problem. We use this algorithm in Section 5 in order to give polynomial-time algorithms to 5-TEDP and TEDP$_1$. We show hardness results for 6-TEDP and TEDP$_2$ (and other restrictions of TEDP) in Section 6. Finally, Section 7 contains concluding remarks and open problems.

2. Preliminaries

For a graph G, $E(G)$ denotes the set of edges of G, and $e(G) = |E(G)|$. For a graph G and a set of edges $S \subseteq E(G)$, $G - S$ is the graph obtained from G by deleting the edges from S. For a set of vertices S, $G - S$ is the graph obtained from G by deleting the vertices in S and the edges that are incident with these vertices.

A *rooted tree* (*forest*) is a triplet $G = (V, E, r)$, where (V, E) is a tree (forest), and r is some vertex in V which is called the *root*. We write G^r to denote the rooted tree G with root r. Also, for an unrooted tree G, we denote by G^r the

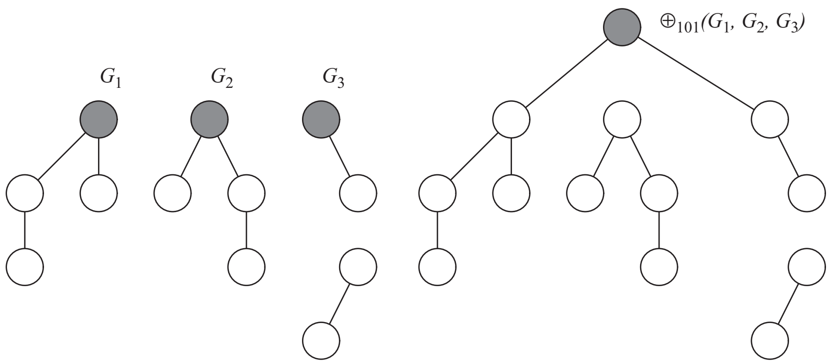

Fig. 1. Example for the definition of $\oplus_s$.

rooted tree formed by choosing the vertex r to be its root. We denote by G^r_v the rooted subtree of G^r whose vertices are all the descendants of v, and its root is v. For a rooted forest G and a vertex v in G, we use $c_G(v)$ to denote the number of children of v in G, and c_G to denote the number of children of the root of G.

Let $K_{1,l}$ be a tree that is composed by taking l vertices and a distinguished vertex called the *center*, and connecting the center to all other vertices. We also use $K_{1,l}$ to denote any rooted tree that is isomorphic to the tree $K_{1,l}$ defined above (this will be true also for the following definitions). A tree $K_{1,l}$ will be called a *star of size* $l+1$. We denote by $\hat{K}_{1,l}$ the rooted tree obtained by taking $K_{1,l}$ and selecting its center to be the root. We denote by $\hat{P}_l$ the rooted tree formed by taking a path with l vertices and choosing one of the two path endpoints as the root.

We say that two rooted forests G^r and H^s are *isomorphic* if there is an isomorphism between G and H which maps r to s. We write $H^s \subseteq_R G^r$ if there is a rooted subforest J^r of G^r which is isomorphic to H^s (note that the subtree J^r must have the same root as G^r). For a tree (rooted or unrooted) G and an unrooted tree H, we write $H \subseteq G$ if H is isomorphic to a subtree of G. For a tree G and a set of trees $\mathscr{H}$ we write $\mathscr{H} \subseteq_\exists G$ if $H \subseteq G$ for some $H \in \mathscr{H}$. Note that the relations $\subseteq$ and $\subseteq_R$ are transitive.

An *l-ary rooted forest operator* is a mapping f which acts on l rooted forests and yields a rooted forest. Given $G_1, \ldots, G_l$, the forest $f(G_1, \ldots, G_l)$ is built by taking the forests $G_1, \ldots, G_l$, and then performing some of the following operations:

1. Merging the roots of some of the input forests.
2. Adding new vertices.
3. Adding new edges, where each endpoint of a new edge is either the root of an input forest or a new vertex.

Finally, the root of $f(G_1, \ldots, G_l)$ is either the root of some input forest or a new vertex. We now give an example for definition of rooted forest operator. For every string $s = s_1 \cdots s_l$ over the alphabet $\{0, 1\}$, define the operator $\oplus_s$ as follows: given l rooted forests $G_1, \ldots, G_l$, $\oplus_s(G_1, \ldots, G_l)$ is the rooted forest obtained by taking $G_1, \ldots, G_l$, adding a new vertex v, connecting the root of G_i to v for every i such that $s_i = 1$, and making v the root. See Fig. 1 for an example.

For an operator f, let $a(f)$ denote the number of forests on which f operates. An operator f is a *suboperator* of an operator f' if $a(f) = a(f')$ and for every $G_1, \ldots, G_{a(f)}$, $f(G_1, \ldots, G_{a(f)})$ is a subgraph of $f'(G_1, \ldots, G_{a(f)})$. For an operator f, $\mathrm{sub}(f)$ is the set of all suboperators of f. A set of operators Φ is called *closed* if $\mathrm{sub}(f) \subseteq \Phi$ for every $f \in \Phi$. A set of operators Φ is called *complete* if every rooted forest can be built from the single-vertex rooted tree by a series of applications of the operators in Φ. The set $\{\oplus_s : s \in \{0, 1\}^*\}$ is closed and complete.

Let Φ be a closed and complete set of rooted forest operators. A *composition tree w.r.t.* Φ of a rooted forest G is a rooted tree H, where each internal vertex v of H is labeled by an operator $f \in \Phi$ such that $a(f)$ is equal to the number of children of v. Each vertex in the tree is associated with a rooted forest: a leaf is associated with the forest $\hat{P}_1$ and an internal vertex is associated with the rooted forest formed by applying the vertex' operator on the forests associated with the children of the vertex. The forest associated with the root of the composition tree is isomorphic to G. An example of a composition tree is shown in Fig. 2.

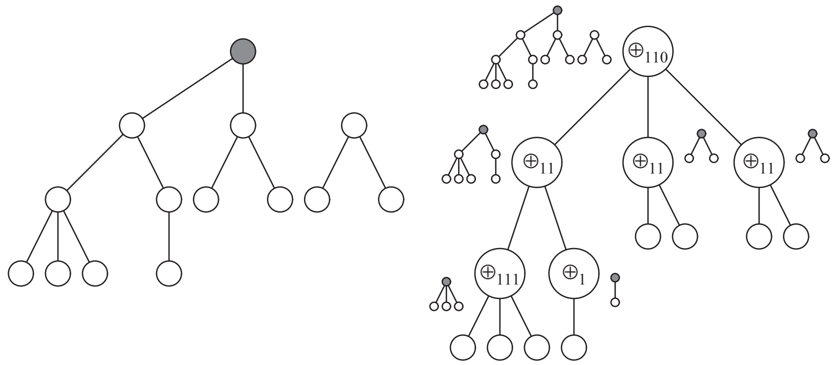

Fig. 2. A rooted forest (left) and its composition tree (right). For each internal vertex of the composition tree, the forest associated with the vertex is shown besides the vertex.

If G^r and H^s are two rooted forests, then we define $G^r + H^s$ to be the unrooted forest formed by taking G^r and H^s and joining their roots by an edge.

Let G be a tree and P be a graph property. We define the characteristic function $P(G)$ to have value 1 if G has property P, and 0 otherwise. A set of edges S such that $P(G-S)=1$ is called a *deletion set of* (G, P) (or a deletion set of G if P is clear from the context). S is called an *optimal deletion set of* (G, P) if it is a deletion set of minimum size.

3. A framework for solving TEDP

In this section we describe a general method for solving TEDPs based on decomposition. We will use this method in Section 5 to give polynomial-time algorithms to several restrictions of TEDP. The general idea behind our framework is similar to the one used by Bern et al. [3] and others (e.g., [2,5]), although some aspects are different, as will be explained later.

We now describe the basic idea of our algorithm. For convenience we describe an algorithm for solving the MSP. Suppose that we have a fixed property P. Let G be the input tree to the MSP, and consider some composition tree of G. Let G' be a rooted tree that corresponds to some vertex in the composition tree. We want to create a set of candidate subforests of G', such that the optimal solution for G will contain one of these candidates as an induced subgraph. In other words, we want to find all "pieces" within G' that may take place in an optimal solution. To do this, we need a way to choose the set. The key to the efficiency of the approach is eliminating as many candidate subforests ("pieces") as possible. The candidate elimination is done by performing pairwise comparisons between candidates and removing candidates according to the results of the comparisons. As an example, consider the tree G' in Fig. 3, and the property P of not containing a path of length 5. Suppose that we start with a set $\mathscr{C}$ containing all the subforests of G'. Clearly, $\mathscr{C}$ has the property that there is an optimal solution for the MSP on G and P that contains one of the forests of $\mathscr{C}$ as an induced subgraph. Now, consider the two subforests $G' - \{a\}$ and $G' - \{b\}$. If there is an optimal solution G^* to the MSP such that the subforest of G^* induced by the vertices of G' is $G' - \{b\}$, then $G_2^* = G^* \cup \{b\} - \{a\}$ is also an optimal solution. Note that the subforest of G_2^* induced by the vertices of G' is $G' - \{a\}$.Hence, if we remove $G - \{b\}$ from the set $\mathscr{C}$, we still have the property that there is an optimal solution for G that contains one of the forests of $\mathscr{C}$ as an induced subgraph. We can therefore say that $G - \{b\}$ is "no better than" $G - \{b\}$. We can continue this process and compare all pairs of candidates. If one candidate is no better than the other, we can remove the former candidate from the set. Note that it is possible to have two candidates such that each one is no better than the other. In this case one candidate is removed arbitrarily.

We now formalize the idea above: we will use quasiorders (recall that a quasiorder is a reflexive and transitive binary relation) to compare candidates for a set. Note that in the discussion above, "no better than" is a quasiorder. We shall define the properties that the quasiorder should have in order to correctly compare candidates. Let $\leqslant$ be some quasiorder on rooted forests. We say that $\leqslant$ is *preserved by* Φ if for every $f \in \Phi$, and every $G_1, \ldots, G_{a(f)}, G'_1, \ldots G'_{a(f)}$ such that $G_i \leqslant G'_i$ for $i = 1, \ldots, a(f)$, there is an operator $f' \in \mathrm{sub}(f)$ such that $f(G_1, \ldots, G_{a(f)}) \leqslant f'(G'_1, \ldots, G'_{a(f)})$. The quasiorder $\leqslant$ is *strongly preserved by* Φ if $f(G_1, \ldots, G_{a(f)}) \leqslant f'(G'_1, \ldots, G'_{a(f)})$ for every $f \in \Phi$ and every

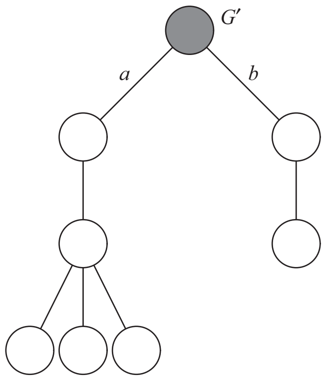

Fig. 3. Example for the definition of a complete forests set. Let P be the property of not containing a path of length 5. The set $\mathscr{C}=\{G'-\{a\}, G'-\{a,b\}\}$ is a complete forests set of G' w.r.t. $\leqslant_2$.

```
1 Arbitrarily choose a vertex r in G.
2 Compute a composition tree for G^r.
3 Scan the vertices of the composition tree in postorder.
4 For every vertex v do
5    Let G' be the tree that corresponds to v.
6    Build a full forests set C_f(G') of G'.
7    Build a complete forests set C(G') of G' from C_f(G').
8 Check for each forest in C(G^r) if it has property P, and output a forest
  from C(G^r) that has property P and has maximum number of edges.
```

Fig. 4. Algorithm MaxSubforest(G).

$G_1,\ldots,G_{a(f)}, G'_1,\ldots G'_{a(f)}$ for which $G_i \leqslant G'_i$ for $i=1,\ldots,a(f)$. We say that $\leqslant$ *preserves P* if $G \leqslant G'$ implies $P(G) \leqslant P(G')$. We say that $\leqslant$ *preserves P with size* if $\leqslant$ preserves P, and additionally, $G \leqslant G'$ and $P(G)=1$ implies that $e(G) \leqslant e(G')$. A (P, Φ)-*order* is a quasiorder that preserves P with size, and is preserved by Φ.

For example, let $\Phi=\{\oplus_s : s \in \{0,1\}^*\}$ and let P be the property of not containing a path of length 4. For a rooted forest G, let $h(G)$ denote the height of the forest G, namely, the length of a longest path that starts at the root of G. We define a quasiorder $\leqslant_1$ by $G \leqslant_1 G'$ if $h(G) \geqslant h(G')$. It is easy to verify that $\leqslant_1$ is preserved by Φ. Moreover, $\leqslant_1$ does not preserve P, since $\hat{P}_2 \leqslant_1 \oplus_0(\hat{P}_5)$, but $P(\hat{P}_2) > P(\oplus_0(\hat{P}_6))$. The quasiorder $\leqslant_2$ defined by

$$G \leqslant_2 G' \quad\Longleftrightarrow\quad P(G)=0 \vee (P(G')=1 \wedge h(G) \geqslant h(G') \wedge e(G) \leqslant e(G'))$$

is a (P, Φ)-order.

Let $\leqslant$ be a quasiorder. For a rooted tree G, a set $\mathscr{C}$ of subforests of G is called a *full forests set of G* (*w.r.t.* $\leqslant$) if for every subforest G_1 of G there is some $G_2 \in \mathscr{C}$ such that $G_1 \leqslant G_2$. A full forests set $\mathscr{C}$ that does not contain two comparable forests is called a *complete forests set*. See Fig. 3 for an example. A set $\mathscr{C}$ of sets of edges of a rooted tree G is called a *complete* (*full*) *set of* G if $\{G-O : O \in \mathscr{C}\}$ is a complete (full) forests set of G.

Suppose that $\leqslant$ is a (P, Φ)-order and we have a procedure to compute $\leqslant$. The MSP with respect to P can be solved by algorithm MaxSubforest that is given in Fig. 4. Building a complete forests set $\mathscr{C}(G')$ of G' from $\mathscr{C}_f(G')$ (line 6) is done by taking $\mathscr{C}(G')=\mathscr{C}_f(G')$, and then, for every pair of forests $G_1, G_2 \in \mathscr{C}(G')$ such that $G_1 \leqslant G_2$, removing G_1 from $\mathscr{C}(G')$. The complete forests set $\mathscr{C}_f(G')$ can be built in several ways. A straightforward way to build $\mathscr{C}_f(G')$ is to take all the subforests of G'. Clearly, this approach is inefficient. A more efficient way to build $\mathscr{C}_f(G')$ is given in the following lemma.

Lemma 3.1. *Let $G'=f(G_1,\ldots,G_l)$ where $f \in \Phi$. If $\mathscr{C}(G_1),\ldots,\mathscr{C}(G_l)$ are complete forests sets of $G_1,\ldots,G_l$ then*

$$\mathscr{C}=\{f'(H_1,\ldots,H_l) : f' \in \mathrm{sub}(f), H_1 \in \mathscr{C}(G_1),\ldots,H_l \in \mathscr{C}(G_l)\}$$

is a full forests set of G'.

Proof. We need to show that for every subforest H of G', there is a subforest $H' \in \mathscr{C}$ such that $H \leqslant H'$. Let H be some subforest of G'. We have that $H = f'(H_1, \ldots, H_l)$, where $f' \in \mathrm{sub}(f)$ and H_i is a subforest of G_i for $i = 1, \ldots, l$. By definition, there are $H'_1 \in \mathscr{C}(G_1), \ldots, H'_l \in \mathscr{C}(G_l)$ such that $H_i \leqslant H'_i$ for $i = 1, \ldots, l$. Since $\leqslant$ is preserved by Φ, it follows that $H \leqslant f''(H'_1, \ldots, H'_l)$ for some $f'' \in \mathrm{sub}(f')$. From the fact that $\mathrm{sub}(f') \subseteq \mathrm{sub}(f)$ we conclude that $f''(H'_1, \ldots, H'_l) \in \mathscr{C}$. $\square$

The correctness of algorithm MaxSubforest follows from the definition of a complete forests sets: Let H^* be an optimal solution for the MSP on the input G and P. From the fact that $\mathscr{C}(G^r)$ is a complete forests set of G^r, it follows that there is a subforest $H \in \mathscr{C}(G^r)$ such that $H^* \leqslant H$, and since $\leqslant$ preserves P with size, it follows that H is an optimal solution. The algorithm returns a forest H' from $\mathscr{C}(G^r)$ that has property P and has maximum number of edges, and in particular, $e(H) \leqslant e(H')$. Therefore, H' is an optimal solution.

Let $|\leqslant|$ denote the size of the largest complete forests set of some rooted tree w.r.t. $\leqslant$. Clearly, the time complexity of algorithm MaxSubforest depends on $|\leqslant|$. Naturally, given P and Φ, our goal will be to a (P, Φ)-order $\leqslant$ such that $|\leqslant|$ is as small as possible. Note that if we build complete sets using the approach of Lemma 3.1, the time complexity for building one complete set is $\Omega(|\leqslant|^d)$, where d is the maximum degree of the composition tree. In Section 5 we will use special properties of 5-TEDP in order to give a different way for building complete sets, whose time complexity does not have exponential dependency on d.

The difference between the approach we described in this section and the approach of Bern et al. [3], is that in the latter, an equivalence relation is used instead of a quasiorder, and the time complexity depends on the number of equivalence classes of the relation. For some properties, the number of equivalence classes in the appropriate equivalence relation is large, while the value of $|\leqslant|$ for the appropriate quasiorder is small.

For the rest of this section assume that P is hereditary and that all the graphs in the obstruction set of P are trees. We use the operators set $\Phi = \{\oplus_s : s \in \{0, 1\}^*\}$. Our goal is to define a (P, Φ)-order $\leqslant'_P$. This (P, Φ)-order will be used in our algorithms in Section 5. We will first define a quasiorder $\leqslant_P$ and show that $\leqslant_P$ is preserved by Φ. Then, we will use $\leqslant_P$ to define a quasiorder $\leqslant'_P$, and we will show that $\leqslant'_P$ is a (P, Φ)-order.

Define the quasiorder $\leqslant_P$ by

$$G \leqslant_P G' \iff P(G) \leqslant P(G') \quad \text{and} \quad P(G + J) \leqslant P(G' + J) \quad \text{for every rooted tree } J.$$

To simplify the notation, we define $G_\emptyset$ to be a special rooted tree such that $G + G_\emptyset = G$ for every rooted forest G (note that for a "true" rooted tree H, $G + H \neq G$). We can now write the definition of $\leqslant_P$ as follows:

$$G \leqslant_P G' \iff P(G + J) \leqslant P(G' + J) \quad \text{for every rooted tree } J.$$

Clearly, $\leqslant_P$ preserves P. The following lemma shows another property of $\leqslant_P$ which we need in order to build the (P, Φ)-order $\leqslant'_P$.

Lemma 3.2. *$\leqslant_P$ is strongly preserved by Φ.*

Proof. Let $s \in \{0, 1\}^*$ be a string of length l, and let $G_1, \ldots, G_l, G'_1, \ldots, G'_l$ be rooted forests such that $G_i \leqslant_P G'_i$ for $i = 1, \ldots, l$. Let J be some rooted forest. If the first letter of s is 1, then since $\oplus_s(G_1, \ldots, G_l) + J = G_1 + \oplus_s(J, G_2, \ldots, G_l)$ and $\oplus_s(G'_1, G_2, \ldots, G_l) + J = G'_1 + \oplus_s(J, G_2, \ldots, G_l)$, it follows that

$$\begin{aligned} P(\oplus_s(G_1, \ldots, G_l) + J) &= P(G_1 + \oplus_s(J, G_2, \ldots, G_l)) \leqslant P(G'_1 + \oplus_s(J, G_2, \ldots, G_l)) \\ &= P(\oplus_s(G'_1, G_2, \ldots, G_l) + J). \end{aligned}$$

We now consider the case when the first letter of s is 0. If $P(G_1) = 0$, from the fact that P is hereditary we obtain that $P(\oplus_s(G_1, \ldots, G_l) + J) = 0$, so

$$P(\oplus_s(G_1, \ldots, G_l) + J) \leqslant P(\oplus_s(G'_1, G_2, \ldots, G_l) + J).$$

Suppose now that $P(G_1) = 1$. As $\leqslant_P$ preserves P, we have that $P(G'_1) = 1$. Thus,

$$P(\oplus_s(G_1, G_2, \ldots, G_l) + J) = P(\oplus_t(G_2, \ldots, G_l) + J) = P(\oplus_s(G'_1, \ldots, G_l) + J),$$

where t is the suffix of length $l - 1$ of s.

For all the cases above we have shown that

$$P(\oplus_s(G_1, \dots, G_l) + J) \leqslant P(\oplus_s(G'_1, G_2, \dots, G_l) + J).$$

Repeating the same argument gives that

$$P(\oplus_s(G_1, \dots, G_l) + J) \leqslant P(\oplus_s(G'_1, \dots, G'_l) + J). \qquad \square$$

The operators set Φ has the following property: for a rooted G' that corresponds to some vertex in a composition tree of a tree G, the root of G' has at most one neighbor in G which is not in G'. Consider the example in Fig. 3. Let e be the edge between the root of G' and its parent in G. By the above property, we have that if there is an optimal solution G^* such that the subforest of G^* induced by the vertices of G' is $G' - \{a, b\}$, then $G^* \cup \{a\} - \{e\}$ is also an optimal solution. Hence, only $G' - \{a\}$ will be a candidate in this case. We use this fact to define the (P, Φ)-order. For two rooted forests G and G',

$$\begin{aligned} G \leqslant^1_P G' &\iff P(G) = 0, \\ G \leqslant^2_P G' &\iff G \leqslant_P G' \quad \text{and} \quad e(G) \leqslant e(G'), \\ G \leqslant^3_P G' &\iff P(G') = 1 \quad \text{and} \quad e(G) < e(G') \end{aligned}$$

and

$$G \leqslant'_P G' \iff G \leqslant^1_P G' \quad \text{or} \quad G \leqslant^2_P G' \text{ or } G \leqslant^3_P G'.$$

Lemma 3.3. *$\leqslant'_P$ is a (P, Φ)-order.*

Proof. We first show that $\leqslant'_P$ is transitive. Suppose that $G \leqslant'_P G' \leqslant'_P G''$. We consider three cases:

Case 1: $G \leqslant^1_P G'$. In this case we have that $G \leqslant^1_P G''$. In the following two cases we assume that $G \nleqslant^1_P G'$, namely $P(G) = 1$.

Case 2: $G \leqslant^2_P G'$. From the fact that $\leqslant_P$ preserves P and since $P(G) = 1$, we have that $P(G') = 1$, so $G' \nleqslant^1_P G''$. If $G' \leqslant^2_P G''$ then $G \leqslant_P G' \leqslant_P G''$ and $e(G) \leqslant e(G') \leqslant e(G'')$. Therefore, $G \leqslant_P G''$ and $e(G) \leqslant e(G'')$, and it follows that $G \leqslant^2_P G'$. Otherwise, if $G' \leqslant^3_P G''$, $e(G) \leqslant e(G') < e(G'')$ and $P(G'') = 1$, and it follows that $G \leqslant^3_P G''$.

Case 3: $G \leqslant^3_P G'$. Again, we have that $G' \nleqslant^1_P G''$. Suppose that $G' \leqslant^2_P G''$. From the fact that $\leqslant_P$ preserves P we have that $P(G'') = 1$. Furthermore, $e(G) < e(G') \leqslant e(G'')$. Thus, $G \leqslant^3_P G''$. We now consider the case when $G' \leqslant^3_P G''$. In this case, $P(G'') = 1$ and $e(G) < e(G') < e(G'')$. Hence, $G \leqslant^3_P G''$. This completes the proof that $\leqslant'_P$ is transitive.

It is easy to verify that $\leqslant'_P$ preserves P. Moreover, if $G \leqslant'_P G'$ and $P(G) = 1$, then either $G \leqslant^2_P G'$ or $G \leqslant^3_P G'$. In both cases, $e(G) \leqslant e(G')$. Therefore, $\leqslant'_P$ preserves P with size.

Finally, to show that $\leqslant'_P$ is preserved by Φ, let $G_1, \dots, G_l, G'_1, \dots, G'_l$ be rooted forests such that $G_i \leqslant'_P G'_i$ for $i = 1, \dots, l$, and let $\oplus_s$ be some operator from Φ with $|s| = l$. We need to show that there is a string s' of length l such that $\oplus_{s'}$ is a suboperator of $\oplus_s$ (i.e., for every $i \leqslant l$, the ith letter of s' is less than or equal to the ith letter of s) and $\oplus_s(G_1, \dots, G_l) \leqslant'_P \oplus_{s'}(G'_1, \dots, G'_l)$. Let a denote the first letter of s, and let t denote the suffix of s of length $l - 1$.

Let $G = \oplus_s(G_1, \dots, G_l)$, $G' = \oplus_s(G'_1, G_2, \dots, G_l)$, and $G^0 = \oplus_{0t}(G'_1, G_2, \dots, G_l)$. We will show that either $G \leqslant'_P G'$ or $G \leqslant'_P G^0$. Repeated use of the same arguments gives that $G \leqslant'_P \oplus_{s'}(G'_1, \dots, G'_l)$ for some string s'.

If $G_1 \leqslant^1_P G'_1$ then $P(G) = P(G_1) = 0$, so $G \leqslant^1_P G'$. If $G_1 \leqslant^2_P G'_1$ then $G_1 \leqslant_P G'_1$ and by Lemma 3.2 we have that $G \leqslant_P G'$. Furthermore, $e(G_1) \leqslant e(G'_1)$ so $e(G) \leqslant e(G')$. Therefore, $G \leqslant^2_P G'$.

We now consider the case when $G_1 \leqslant^3_P G'_1$. We will show that $G \leqslant_P G^0$. Let J be some rooted tree J such that $P(G^0 + J) = 0$. $G^0 + J$ contains a subforest isomorphic to a tree from the obstruction set of P, and this subforest is contained in one of the connected components of $G^0 + J$. This component cannot be a component of G'_1 as $P(G'_1) = 1$, so the subforest must be a subgraph of $\oplus_t(G_2, \dots, G_l) + J$. Therefore, $P(G^0 + J) = 0$. Since this is true for all J, it follows that $G \leqslant_P G^0$. Moreover, $e(G) \leqslant e(G_1) + 1 + e(\oplus_t(G_2, \dots, G_l)) \leqslant e(G'_1) + e(\oplus_t(G_2, \dots, G_l)) = e(G^0)$. Hence, $G \leqslant^2_P G^0$. $\square$

To implement algorithm MaxSubforest we need an efficient way to decide for two rooted forests G_1 and G_2, whether $G_1 \leqslant'_P G_2$. To decide whether $G_1 \leqslant'_P G_2$, we need to decide whether $G_1 \leqslant_P G_2$. The definition of $\leqslant_P$ requires computing $P(G_1+J)$ and $P(G_2+J)$ for an infinite number of rooted trees J. However, we will show that it suffices to consider a finite number (that depends on P) of rooted trees, and therefore computing whether $G_1 \leqslant'_P G_2$ can be done efficiently.

Let $\mathscr{H}_P$ be the obstruction set of the property P. Let $F_{P,0}$ be a set that contains for every $H \in \mathscr{H}_P$ and every edge e in H which is not incident of a leaf, the two rooted trees obtained by removing e from H and choosing the two endpoints of e as the roots. Let F_P be a set that contains of all distinct (i.e., non-isomorphic) rooted trees in $F_{P,0}$. Additionally, F_P contains $\hat{K}_{1,0}$ (a rooted tree with one vertex) and the rooted tree $G_\emptyset$, which we will also denote by $\hat{K}_{1,-1}$.

The following lemma allows us to compute whether $G_1 \leqslant_P G_2$ efficiently.

Lemma 3.4. *For two rooted forests G_1 and G_2, $G_1 \leqslant_P G_2$ if and only if $P(G_1+J) \leqslant P(G_2+J)$ for every $J \in F_P$.*

Proof. The lemma follows from the fact that for a rooted tree $J \notin F_P$, $P(G+J)=1$ if and only if $J' \subseteq_R J$ for some $J' \in F_P$. Therefore, the values $P(G_1+J)$ and $P(G_2+J)$ can be ignored when checking whether $G_1 \leqslant_P G_2$.

Formally, suppose that $P(G_1+J) \leqslant P(G_2+J)$ for every $J \in F_P$. We need to show that for every rooted tree J, $P(G_1+J) \leqslant P(G_2+J)$. In other words, we need to show that $\mathscr{H}_P \subseteq_\exists G_2+J$ implies $\mathscr{H}_P \subseteq_\exists G_1+J$ for every rooted tree J. Let J be some rooted tree for which $\mathscr{H}_P \subseteq_\exists G_2+J$. There is a subgraph H of G_2+J which is isomorphic to a tree from $\mathscr{H}_P$. Let J_H be the rooted subtree of J which is induced by the vertices of H (if H does not contain vertices from J then $J_H = G_\emptyset$). We have that $H \subseteq G_2+J_H$ and therefore $\mathscr{H}_P \subseteq_\exists G_2+J_H$. If the root of G_2 is the only vertex from H in G_2 then $J_H+\hat{P}_1$ is isomorphic to a tree in $\mathscr{H}_P$, and we obtain that $\mathscr{H}_P \subseteq_\exists G_1+J_H$. Otherwise, $J_H \in F_P$. From the fact that $J_H \in F_P$ and $\mathscr{H}_P \subseteq_\exists G_2+J_H$ it follows that $\mathscr{H}_P \subseteq_\exists G_1+J_H$. Therefore $\mathscr{H}_P \subseteq_\exists G_1+J$.

The second direction of the lemma follows directly from the definition of $\leqslant_P$. □

We finish this section by showing a property of the quasiorder $\leqslant'_P$. This property will be used later in Section 5.

Lemma 3.5. *Let $\mathscr{C}$ be a complete forests set of a rooted tree G w.r.t. $\leqslant'_P$. Then, every $G' \in \mathscr{C}$ is a maximum subforest of G with property P.*

Proof. Let G^* be a maximum subforest of G with property P.

We first show that every G' in $\mathscr{C}$ has property P. Suppose conversely that $P(G')=0$ for some $G' \in \mathscr{C}$. If $|\mathscr{C}|>1$ then $G' \leqslant'_P G''$ for every $G'' \in \mathscr{C}-\{G'\}$. Otherwise, since $\leqslant'_P$ preserves P, it follows that $G^* \nleqslant'_P G'$. In both cases we obtain a contradiction the fact that $\mathscr{C}$ is a complete set. Therefore, every G' in $\mathscr{C}$ has property P.

All the forests in $\mathscr{C}$ have the same size, otherwise, if $G_1, G_2 \in \mathscr{C}_v$ and $e(G_2)<e(G_1)$ then $G_1 \leqslant'_P G_2$, a contradiction. Moreover, we have that $G^* \leqslant'_P G'$ for some $G' \in \mathscr{C}$. From the fact that $\leqslant'_P$ preserves P with size it follows that $e(G')=e(G^*)$. Therefore, every forest in $\mathscr{C}$ is a maximum subforest of G with property P. □

4. The set deletion problem

In this section we define a problem on weighted sets which will be used in Section 5 in the algorithms for 5-TEDP and TEDP_1. A *weighted set* is a set S of elements with a weight function $w: S \to \mathrm{N}$. We will use $\{a_1,\ldots,a_n\}$ to denote a weighted set with n elements, where the weights of the element are $a_1,\ldots,a_n$. A mapping $f: P \to S$ between two weighted set is called a *weight increasing mapping* if f is injective and $w(f(p)) \geqslant w(p)$ for every $p \in P$. For two weighted sets P and S, we say that S is *larger* than P, denoted $P \preccurlyeq S$, if there is a weight increasing mapping from P to S. For example, $\{2,3,4\} \preccurlyeq \{3,3,3,5,5\}$. Clearly, the relation $\preccurlyeq$ is a partial order. If $\mathscr{P}$ is a set of sets, then we write $\mathscr{P} \preccurlyeq S$ if $P \preccurlyeq S$ for some $P \in \mathscr{P}$.

Let S be a weighted set, and $\mathscr{P}$ be a set of weighted sets. Let f and g be two mappings that map the elements of S to subsets of $\mathscr{P}$. An element $x \in S$ is called a *bad element of* (S,f,g) if either $f(x) \preccurlyeq S$ or $g(x) \preccurlyeq S-\{x\}$. A set $O \subseteq S$ is called a *deletion set of* (S,f,g) if there are no bad elements of $(S-O,f,g)$. Let $\mathrm{OPT}(S,f,g)$ denote the minimum size of a deletion set of (S,f,g).

As an example of the definitions above, let $S=\{a_1,\ldots,a_6\}$ be a weighted set, where the weights of $a_1,\ldots,a_6$ are 8, 7, 6, 5, 4, 3, respectively. Let $\mathscr{P}=\{\{4,4\},\{4,4,4\}\}$, $f(x)=\{\{4,4,4\}\}$ for all $x \in S$, $g(a_5)=\{\{4,4\}\}$, and $g(x)=\emptyset$ for all $x \neq a_5$. In this example, $\mathrm{OPT}(S,f,g)=3$ as $\{a_1,a_2,a_5\}$ is a deletion set of (S,f,g), and there are no deletion sets of size less than 3.

A deletion set O of (S, f, g) is called a *maximum deletion set of* (S, f, g) if $O' \preccurlyeq O$ for every deletion set O' of (S, f, g) such that $|O| = |O'|$. We will later show that for every $l \geqslant \mathrm{OPT}(S, f, g)$, a maximum deletion set of (S, f, g) of size l exists. For a weighted set P, define $[P]_l$ be the set obtained from P by deleting an element in P with the maximum weight among the elements with weight less than or equal to l, if there are such elements. For a set of weighted sets $\mathscr{P}$, let $[\mathscr{P}]_l = \{[P]_l : P \in \mathscr{P}\}$ and for a mapping f between a weighted set S to sets of weighted sets, $[f]_l$ is the mapping defined by $[f]_l(x) = [f(x)]_l$ for every $x \in S$. Let $\alpha(S, f, g) = \min(\{l : \mathrm{OPT}(S, [f]_l, [g]_l) > 0\} \cup \{\infty\})$.

We now prove that for every $l \geqslant \mathrm{OPT}(S, f, g)$, there is a maximum deletion set of (S, f, g) of size l. Our proof is constructive, and moreover, it gives a polynomial-time algorithm for finding maximum deletion sets. This also implies that $\alpha(S, f, g)$ can be computed in polynomial time.

We need the following property of the relation $\preccurlyeq$.

Lemma 4.1. *If* $T, T' \subset S$ *and* $T \preccurlyeq T'$ *then* $S - T \succcurlyeq S - T'$.

Proof. We first claim that there is a weight increasing mapping $f : T \to T'$ such that $f(x) = x$ for every $x \in T \cap T'$. To probe this claim, let $g : T \to T'$ be some weight increasing mapping. Let x_1 be some element of $T - T'$, and define a sequence $x_1, x_2, \ldots$, where $x_i = g(x_{i-1})$ for $i > 1$, and the sequence is terminated at an element x_k for which $x_k \notin T$. Since $x_1 \notin T'$, we have that $x_i \neq x_1$ for every i. Moreover, from the fact g is injective, we obtain that the elements $x_1, x_2, \ldots$ are distinct, and therefore the sequence terminates. We now define a mapping $f : T \to T'$ as follows: for $x_1 \in T - T'$, let $x_1, \ldots, x_k$ be the sequence as defined above, and define $f(x_1) = x_k$. For $x \in T \cap T'$ define $f(x) = x$. It is easy to verify that f is weight increasing function.

Now, define $f' : S - T' \to S - T$ as follows: $f'(x) = x$ for every $x \in S - (T \cup T')$, and $f'(x) = f(x)$ for every $x \in T - T'$. It is easy to verify that f' is weight increasing function and therefore $S - T \succcurlyeq S - T'$. $\square$

Let $s_1, \ldots, s_n$ be the elements of S, where $w(s_1) \geqslant w(s_2) \geqslant \cdots \geqslant w(s_n)$. Consider a simple case when the constraints of f and g are the same for all the elements of S, i.e. $f(s_1) = f(s_2) = \cdots = f(s_n)$ and $g(s_1) = g(s_2) = \cdots = g(s_n)$. In this case, Lemma 4.1 indicates that for every $l \geqslant \mathrm{OPT}(S, f, g)$, the set $\{s_1, \ldots, s_l\}$ (namely, the l heaviest elements of S) is a maximum deletion set of (S, f, g) of size l. The case of general f and g is not so simple. Consider the example given above, namely $S = \{a_1, \ldots, a_6\}$ where the weights of the elements of S are 8, 7, 6, 5, 4, 3, $\mathscr{P} = \{\{4, 4\}, \{4, 4, 4\}\}$, $f(x) = \{\{4, 4, 4\}\}$ for all $x \in S$, $g(a_5) = \{\{4, 4\}\}$, and $g(x) = \emptyset$ for all $x \neq a_5$. We have $\mathrm{OPT}(S, f, g) = 3$, but the set $\{a_1, a_2, a_3\}$ containing the three heaviest elements of S is not a deletion set of (S, f, g).

Even though the set of l heaviest elements may not be a deletion set, it is still desirable to take heaviest elements of S into a deletion set O since these elements will make $S - O$ small w.r.t. the relation $\preccurlyeq$ and thus $S - O$ will be larger than only few of the sets in $\mathscr{P}$. Moreover, the heaviest elements will make O larger than other deletion sets of the same size as O. Thus, to build a maximum deletion set of (S, f, g) of size l, we take the k heaviest elements of S for some $k \leqslant l$. To these elements, we add $l - k$ elements of S that are needed to make the set a deletion set. As we do not know the value of k, we will try all possible values.

We now formally define the sets that we build. For every $i \leqslant n$, let A_i be the set of $n - i$ heaviest elements of S (that is, $A_i = \{s_1, \ldots, s_{n-i}\}$). Define $B_0 = \emptyset$ and

$$B_i = B_{i-1} \cup \{x \in S - (A_i \cup B_{i-1}) : x \text{ is a bad element of } (S - (A_i \cup B_{i-1}), f, g)\}.$$

Let $O_i = A_i \cup B_i$. Note that $A_i \cap B_i = \emptyset$ for all i. The definition of B_i implies that the sets $O_0, \ldots, O_n$ are deletion sets of (S, f, g). We will show that for every $l \geqslant \mathrm{OPT}(S, f, g)$, one of the sets $O_0, \ldots, O_n$ is a maximum deletion set of (S, f, g) of size l. To prove this, we need the following lemma.

Lemma 4.2. *Let* O *be a deletion set of* (S, f, g) *and let* $i \geqslant 0$ *be some integer. If* $|O_j| > |O|$ *for every* $0 \leqslant j < i$, *then* $O \preccurlyeq O_i$ *and* $B_i \subseteq O$.

Proof. We prove the lemma using induction on i. The base of the induction is satisfied since $O_0 = S$ and $B_0 = \emptyset$. We now prove the lemma for some $i > 0$. By the induction hypothesis, we have that $B_{i-1} \subseteq O$. Therefore, $|A_{i-1}| = |O_{i-1}| - |B_{i-1}| > |O| - |B_{i-1}| = |O - B_{i-1}|$, where the first equality is due to the fact that $A_{i-1} \cap B_{i-1} = \emptyset$. Hence, $|A_i| = |A_{i-1}| - 1 \geqslant |O - B_{i-1}|$, and since A_i consists of elements of largest weights, it follows that $A_i \succcurlyeq O - B_{i-1}$. Since $A_i \cap B_{i-1} = \emptyset$, we obtain that $A_i \cup B_{i-1} \succcurlyeq O$. Therefore, $O_i \succcurlyeq O$.

We now prove that $B_i \subseteq O$. Suppose conversely that $B_i - O \neq \emptyset$ and let $x \in B_i - O$. From the induction hypothesis, $B_{i-1} \subseteq O$, so $x \in B_i - B_{i-1}$, namely x is a bad element of $(S - (A_i \cup B_{i-1}), f, g)$. It follows from Lemma 4.1 that x is a bad element of $(S - O, f, g)$, contradicting the fact that O is a deletion set. □

We now prove the main result of this section.

Theorem 4.3. *For every $l \geqslant \mathrm{OPT}(S, f, g)$, one of the sets $O_0, \ldots, O_n$ is a maximum deletion set of size l of (S, f, g).*

Proof. Fix some $l \geqslant \mathrm{OPT}(S, f, g)$. We first claim that there is a set among $O_0, \ldots, O_n$ of size at most l. Assume conversely that $|O_0|, \ldots, |O_n| > l$, and let O be a deletion set of size l. As $A_n = \emptyset$, we have $O_n = B_n$. By Lemma 4.2 we obtain that $B_n \subseteq O$, hence $|O| \geqslant |O_n| > l$, a contradiction. Thus, there is a set among $O_0, \ldots, O_n$ of size at most l, and let O_i be the first such set.

For every deletion set O of size l, using Lemma 4.2 we get that $O \preccurlyeq O_i$, and therefore $l = |O| \leqslant |O_i|$. It follows that O_i is a maximum deletion set of size l. □

From Theorem 4.3 we conclude that finding maximum deletion sets can be done in polynomial time. Computing $\alpha(S, f, g)$ can also be done in polynomial time.

5. Algorithms for 5-TEDP and TEDP$_1$

In this section we give polynomial-time algorithms for 5-TEDP and TEDP$_1$. We first give an algorithm for 5-TEDP. We will use algorithm MaxSubforest and the relation $\leqslant'_P$ from Section 3.

Before describing the algorithm for 5-TEDP, we give some intuition for the fact that 5-TEDP can be solved in polynomial time, while 6-TEDP is NP-hard. For clarity, some of the statements in the following discussion will not be accurate. A rigorous analysis of 5-TEDP will be given later in this section. Recall that the time complexity of algorithm MaxSubforest depends on the size of the largest complete forests set of some rooted tree w.r.t. $\leqslant'_P$, which is denoted by $|\leqslant'_P|$. In particular, in order to have a polynomial-time algorithm for some variant of TEDP, it is necessary for $|\leqslant'_P|$ to be polynomial in the number of vertices in the obstruction set $\mathscr{H}$.

Consider first the simple case when $\mathscr{H}$ contains only trees of diameter 4. By Lemma 3.5, we can bound $|\leqslant'_P|$ by giving a bound on the maximum number of rooted forests $G_1, \ldots, G_l$ such that each G_i is maximum subforest of a common rooted tree G^r that satisfies P, and each two forests G_i and G_j are incomparable by the quasiorder $\leqslant_P$. By Lemma 3.4, we need to consider the values of $P(G_i + H)$ for all $i \leqslant l$ and $H \in F_P$. We introduce a new definition to simplify the following discussion: for a rooted forest $\hat{G}$, let

$$h(\hat{G}) = \{J \in F_P : P(\hat{G} + J) = 0\}.$$

Clearly, $\hat{G}_1 \leqslant_P \hat{G}_2$ if and only if $h(\hat{G}_1) \supseteq h(\hat{G}_2)$. Therefore, the sets $h(G_1), \ldots, h(G_l)$ are pairwise incomparable by the $\subseteq$ relation.

Using Lemmas 3.1 and 3.5, we can assume that $c_{G_1} = c_{G_2} = \cdots = c_{G_l}$. We now use the fact that for a tree of diameter 4, removing an edge that is not incident with a leaf (and making its endpoint the roots of the two resulting trees) creates two rooted trees, where at least one of the trees is a star. Consequently, we split the set F_P into two sets F_{stars} and $F_{\text{non-stars}}$, where F_{stars} is the set of all rooted stars in F_P, and $F_{\text{non-stars}}$ contains the rest of the trees of F_P. For every $H \in F_{\text{non-stars}}$, the value of $P(\hat{G} + H)$ depends only on $c_{\hat{G}}$. Therefore, $P(G_1 + H) = P(G_2 + H) = \cdots = P(G_l + H)$ for every $H \in F_{\text{non-stars}}$. In other words, $h(G_1) \cap F_{\text{non-stars}} = h(G_2) \cap F_{\text{non-stars}} = \cdots = h(G_l) \cap F_{\text{non-stars}}$. Now, consider some rooted star $H \in F_{\text{stars}}$. If $P(\hat{G} + H) = 1$, then $P(\hat{G} + H') = 1$ for every star $H' \in F_{\text{stars}}$ with more vertices than H. Therefore, for two rooted forests $\hat{G}_1$ and $\hat{G}_2$, one of the sets $h(\hat{G}_1) \cap F_{\text{stars}}$ and $h(\hat{G}_2) \cap F_{\text{stars}}$ contains the other set. It follows that $l = 1$ (otherwise, every two forests from $G_1, \ldots, G_l$ are comparable in the quasiorder $\leqslant_P$, a contradiction).

The analysis becomes more complicated when $\mathscr{H}$ contains trees of diameter 5. A tree of diameter 5 contains an edge (not incident with a leaf) whose removal gives two rooted trees that are not rooted stars. Such edge will be called *special*. Note that there is exactly one special edge in a tree of diameter 5. The two trees that are obtained by removing a special edge will be called *mates*. We now partition the set F_P into three sets: F_{special} contains the rooted trees that are obtained by removing the special edges in the diameter 5 trees of $\mathscr{H}$, F_{stars} is the set of all rooted stars in F_P, and

```
1 Arbitrarily choose a vertex r in G.
2 Scan the vertices of G^r in postorder.
3 For every vertex v do
4    Build a full set C_v^f of G_v^r.
5    Build a complete set C_v of G' from C_v^f.
6 Output an arbitrary set from C_r.
```

Fig. 5. Algorithm MaxSubforest(G).

$F_{\text{non-stars}} = F_P - (F_{\text{special}} \cup F_{\text{stars}})$. The properties of F_{stars} and $F_{\text{non-stars}}$ described above also remain true in this case. Moreover, for a tree $H \in F_{\text{special}}$, $P(\hat{G} + H) = 1$ if and only if $H' \subseteq_R \hat{G}$, where H' is a mate of H. Using the fact that all the trees in F_{special} have height 2, we have that $P(\hat{G} + H)$ depends only on the number of children of the root of $\hat{G}$, and the number of children of each child of the root of $\hat{G}$. This fact gives a connection to the set deletion problem (see Claim 5.1 and Lemma 5.2). From this connection we obtain that the set $\{h(G_1) \cap F_{\text{special}}, \ldots, h(G_l) \cap F_{\text{special}}\}$ is totally ordered by the partial order $\subseteq$, namely$h(G_{\pi(1)}) \cap F_{\text{special}} \subseteq h(G_{\pi(2)}) \cap F_{\text{special}} \subseteq \cdots \subseteq h(G_{\pi(l)}) \cap F_{\text{special}}$ for some permutation π. Since the set $\{h(G_1) \cap F_{\text{star}}, \ldots, h(G_l) \cap F_{\text{star}}\}$ is also totally ordered by $\subseteq$ and $G_1, \ldots, G_l$ are pairwise incomparable by $\leqslant_P$, it follows that $h(G_{\pi(1)}) \cap F_{\text{special}} \subset h(G_{\pi(2)}) \cap F_{\text{special}} \subset \cdots \subset h(G_{\pi(l)}) \cap F_{\text{special}}$ and $h(G_{\pi(1)}) \cap F_{\text{star}} \supset h(G_{\pi(2)}) \cap F_{\text{star}} \supset \cdots \supset h(G_{\pi(l)}) \cap F_{\text{star}}$. Therefore $l \leqslant |F_{\text{special}}| + 1$.

Finally, our method does not give a polynomial-time algorithm for 6-TEDP: for a tree of diameter 6, a special edge gives two rooted trees, one with height 2 and one with height 3. Due to the height 3 trees, the set $\{h(G_1) \cap F_{\text{special}}, \ldots, h(G_l) \cap F_{\text{special}}\}$ may not be totally ordered by $\subseteq$. Therefore, in this case l can be exponential in $|F_{\text{special}}|$.

We now describe the algorithm for 5-TEDP. We shall use a slightly different formulation of algorithm MaxSubforest. The new formulation uses the fact that a rooted tree G^r has a unique composition tree under the operators set Φ. Every vertex u in the composition tree corresponds to some vertex v in G, and the tree associated with u is G^r_v. It will be more convenient to describe the algorithm using complete sets instead of complete forests sets, namely for each vertex v, the algorithm computes a complete set of G^r_v. The new formulation of algorithm MaxSubforest is given in Fig. 5. From Lemma 3.4, step 5 of the algorithm can be implemented in polynomial time in the input size (assuming that $|\mathscr{C}^f_v|$ is polynomial in the input size). In the rest of the section, we will show how to perform step 4 of the algorithm in polynomial time (in particular, this implies that $|\mathscr{C}^f_v|$ is polynomial in the input size).

We begin with some definitions. For a weighted set $L = \{l_1, \ldots, l_d\}$, define the rooted tree

$$\hat{S}(L) = \oplus_{11\cdots 1}(\hat{K}_{1,l_1-1}, \ldots, \hat{K}_{1,l_d-1}).$$

The root of $\hat{S}(L)$ is called the *center vertex*. We also define $S(L)$ to be the unrooted tree obtained from $\hat{S}(L)$. As an example, the tree $S(\{2, 3, 4\})$ is shown in Fig. 6(a). If $H = S(L)$ then L is called a *representation* of H. Note that every tree with diameter 4 has a unique representation, while a tree with diameter 2 or 3 has at most two representations.

Define $S(L_1, L_2) = S(L_1) + S(L_2)$. Every tree H with diameter 5 is the form $S(L_1, L_2)$ for some L_1 and L_2. The centers of $S(L_1)$ and $S(L_2)$ are called the centers of H. $\hat{S}(L_1, L_2)$ denotes the rooted tree obtained from $\hat{S}(L_1, L_2)$ by making the center of $S(L_1)$ the root.

For the rest of this section, we will show how to solve 5-TEDP for some fixed obstruction set $\mathscr{H} = \{H_1, \ldots, H_p\}$. Without loss of generality, we assume that every tree in $\mathscr{H}$ contains at most n vertices (if $\mathscr{H}$ contains trees with more than n vertices, we can remove these trees from $\mathscr{H}$). We also assume that the trees in $\mathscr{H}$ with diameter 5 are $H_1, \ldots, H_{p'}$. Let L^i_j be weighted sets such that $H_i = S(L^i_1, L^i_2)$ for $i = 1, \ldots, p'$. Let $\mathscr{L}$ be the set of all the representations of $H_{p'+1}, \ldots, H_p$, and let $\mathscr{L}' = \{L^i_j : i = 1, \ldots, p', j = 1, 2\}$.

Recall that for a rooted forest $\hat{G}$,

$$h(\hat{G}) = \{J \in F_P : \mathscr{H} \subseteq_\exists \hat{G} + J\}.$$

We have shown above that the set $h(\hat{G}) \cap F_{\text{star}}$ has an important role. This set can be represented by a single number: define

$$h_2(\hat{G}) = \min(\{i \geqslant 0 : \mathscr{H} \subseteq_\exists \hat{G} + \hat{K}_{1,i-1}\} \cup \{\infty\}).$$

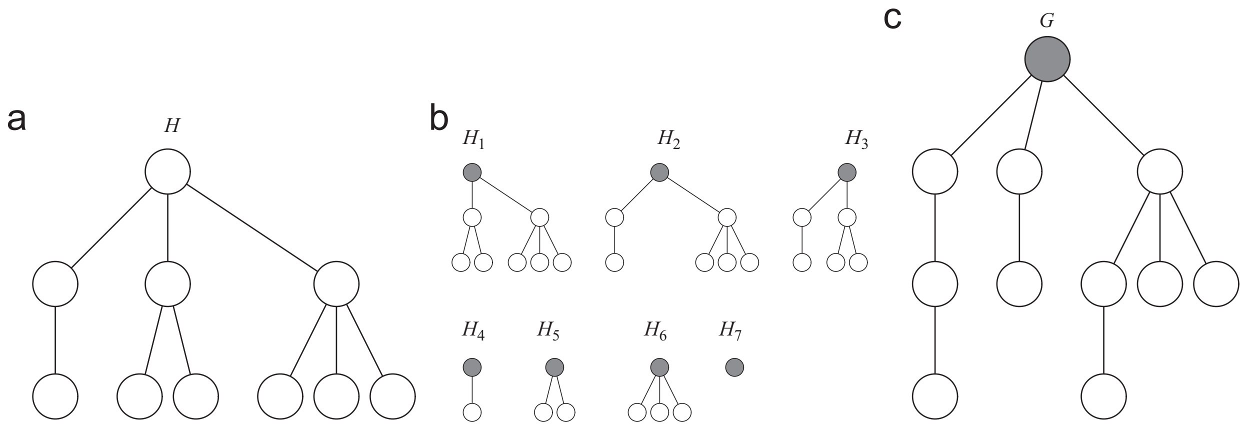

Fig. 6. An example for the definition of h_1 and h_2. Suppose that $\mathscr{H}$ consists of a single tree $H = S(\{2, 3, 4\})$ (a). The set $F_P = \{H_1, H_2, H_3, H_4, H_5, H_6, H_7, G_\emptyset\}$ is shown in (b). For the rooted tree G shown in (c), $h(G) = \{H_1, H_2, H_3, H_5, H_6\}$. We have $h_2(G) = 3$ as $H \subseteq G + \hat{K}_{1,2}$ and $H \nsubseteq G + \hat{K}_{1,1}$. Furthermore, $h_1(G) = \emptyset$.

We also define

$$h_1(\hat{G}) = \{H \in h(\hat{G}) : c_H \leqslant h_2(\hat{G}) - 2\}.$$

See Fig. 6 for an example. Note that

$$h(\hat{G}) = h_1(\hat{G}) \cup \{H \in F : c_H \geqslant h_2(\hat{G}) - 1\},$$

so we have that $G_1 \leqslant_P G_2$ if and only if $h_1(G_2) \subseteq h(G_1)$ and $h_2(G_2) \leqslant h_2(G_1)$.

We first show the connection between the quasiorder $\leqslant'_P$ and the set deletion problem from Section 4. Let $\hat{G}^x$ be some rooted forest. Define $S_{\hat{G}}$ to be a weighted set whose elements are the edges between the root x of $\hat{G}$ and its children, and the weight of an edge $(x, u) \in S_{\hat{G}}$ is $c_{\hat{G}}(u) + 1$. We have the following simple connection between subforest isomorphism and the $\preccurlyeq$ relation on weighted sets:

Claim 5.1. *Let $\hat{G}^x$ be a rooted forest such that $\mathscr{H} \nsubseteq_\exists \hat{G} - \{x\}$. Then*

1. *For every weighted set L, $\hat{S}(L) \subseteq_R \hat{G}$ if and only if $L \preccurlyeq S_{\hat{G}}$.*
2. *For every weighted sets L_1 and L_2, $\hat{S}(L_1, L_2) \subseteq_R \hat{G}$ if and only if there is a child u of x such that $\hat{S}(L_2) \subseteq_R \hat{G}^x_u$ and $L_1 \preccurlyeq S_{\hat{G}} - \{u\}$.*

We now show a connection between the mapping h_2 and the set deletion problem. For a rooted forest $\hat{G}^x$, and a child u of x, let $Z_{u,\hat{G}}$ be a weighted set containing $h_2(\hat{G}^x_u) - 1$ elements of weight 1 each. We define two mappings as follows: for every child u of x,

$$f(u) = \mathscr{L}$$

and

$$g_{\hat{G}}(u) = \{L \in \mathscr{L}' : \hat{S}(L) \in h_1(\hat{G}^x_u)\} \cup \{Z_{u,\hat{G}}\}.$$

Lemma 5.2. *Let $\hat{G}^x$ be a rooted forest such that $\mathscr{H} \nsubseteq_\exists \hat{G} - \{x\}$. Then, $h_2(\hat{G}) = \alpha(S_{\hat{G}}, f, g_{\hat{G}})$.*

Proof. Before we prove the lemma, consider a special case of the lemma that states that $h_2(\hat{G}) = 0$ if and only if $\alpha(S_{\hat{G}}, f, g_{\hat{G}}) = 0$. In other words, $\hat{G}$ contains a subtree isomorphic to a tree in $\mathscr{H}$ if and only if there is a bad element of $(S_{\hat{G}}, f, g_{\hat{G}})$. In one direction, if $\hat{G}$ contains a subtree isomorphic to a tree $H \in \mathscr{H}$, then using Claim 5.1 we obtain that $L \preccurlyeq S_{\hat{G}}$ or $L \preccurlyeq S_{\hat{G}} - \{u\}$ for some weighted set L that depends on H. We have that $L \in f(u)$ or $L \in g_{\hat{G}}(u)$ (since the mappings f and $g_{\hat{G}}$ were designed to contain all the sets L that are obtained from having a subtree of $\hat{G}$ isomorphic to a tree in $\mathscr{H}$). Therefore, u is a bad element of $(S_{\hat{G}}, f, g_{\hat{G}})$. The other direction of thestatement above also follows from Claim 5.1.

We now give the proof for the lemma. We first show that $\alpha(S_{\hat{G}}, f, g_{\hat{G}}) \leqslant h_2(\hat{G})$. Suppose that $h_2(\hat{G}) = l$. We need to show that $\alpha(S_{\hat{G}}, f, g_{\hat{G}}) \leqslant l$, namely there is a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$.

By the definition of h_2, there is a subtree H of $\hat{G} + \hat{K}_{1,l-1}$ which is isomorphic to a tree in $\mathscr{H}$. By the minimality of l, H contains all the vertices of $\hat{K}_{1,l-1}$. If the diameter of H is 5 then its two centers are vertices of $\hat{G}$. Otherwise, we can choose a representation of H in which the center vertex is a vertex of $\hat{G}$.

In the proof we consider two cases, according to whether x is a center vertex of H. If x is a center vertex of H, we consider two sub-cases: if the diameter of H is at most 4, then $H = S(L)$ for some weighted set L. We have that $\hat{S}([L]_l) \subseteq_R \hat{G}$, and by Claim 5.1, $[L]_l \preccurlyeq S_{\hat{G}}$. Since $L \in f(u)$ for every $u \in S_{\hat{G}}$, it follows that every element of $S_{\hat{G}}$ is a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$.

The second sub-case is when the diameter of H is 5. Suppose that $H = S(L_1, L_2)$, where x is the center that corresponds to L_1. By Claim 5.1, $\hat{S}(L_2) \subseteq_R \hat{G}^x_u$ for some child u of x, and $[L_1]_l \preccurlyeq S_{\hat{G}} - \{u\}$. If $|L_1| \geqslant h_2(G^x_u) - 1$ then $[Z_{u,\hat{G}}]_l \preccurlyeq [L_1]_l \preccurlyeq S_{\hat{G}} - \{u\}$, so u is a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$. Otherwise, $\mathscr{H} \subseteq_\exists \hat{G}^x_u + \hat{S}(L_1)$ and $c_{\hat{S}(L_1)} = |L_1| \leqslant h_2(\hat{G}^x_u) - 2$, and therefore$L_1 \in g_{\hat{G}}(u)$. Thus, u is a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$.

If x is not a center vertex of H, then $l \leqslant 1$ (as every vertex of H has distance at most two to a center of H), and there is a child u of x such that every vertex of H is either x, a child of x, a descendent of u, or the single vertex of the star $\hat{K}_{1,l-1}$ (if $l = 1$). It follows that $\mathscr{H} \subseteq_\exists \hat{G}^x_u + \hat{K}_{1,t-1+l}$, so $h_2(\hat{G}^x_u) \leqslant t + l$. Therefore, $[Z_{u,\hat{G}}]_l \preccurlyeq S_{\hat{G}} - \{u\}$, so u is a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$ and $\alpha(S_{\hat{G}}, f, g_{\hat{G}}) \leqslant l$.

We now show that $h_2(\hat{G}) \leqslant \alpha(S_{\hat{G}}, f, g_{\hat{G}})$. Suppose that $\alpha(S_{\hat{G}}, f, g_{\hat{G}}) = l$ and let u be a bad element of $(S_{\hat{G}}, [f]_l, [g_{\hat{G}}]_l)$. By definition, either $[L]_l \preccurlyeq S_{\hat{G}}$ for some $L \in f(u)$, or $[L]_l \preccurlyeq S_{\hat{G}} - \{u\}$ for some $L \in g_{\hat{G}}(u)$. If $[L]_l \preccurlyeq S_{\hat{G}}$ for some $L \in f(u) = \mathscr{L}$, then by Claim 5.1, $\hat{S}([L]_l) \subseteq_R \hat{G}$. It follows that $S(L) \subseteq \hat{G} + K_{1,l-1}$ and therefore $h_2(\hat{G}) \leqslant l$. Otherwise, $[L]_l \preccurlyeq S_{\hat{G}} - \{u\}$ for some $L \in g_{\hat{G}}(u)$. By Claim 5.1, $\hat{S}([L]_l) \subseteq_R \hat{G}^u_x$ (note that $\hat{G}^u_x$ is the rooted tree obtained by removing u and its descendants from $\hat{G}^x$), so $\hat{S}(L) \subseteq_R \hat{G}^u_x + K_{1,l-1}$. If $L = Z_{u,\hat{G}}$, then the definition of α implies that $l \leqslant 1$ and $c_{\hat{G}} \geqslant h_2(\hat{G}^x_u) - l$. By the definition of h_2, $\mathscr{H} \subseteq_\exists \hat{G} + \hat{K}_{1,l-1}$, so $h_2(\hat{G}) \leqslant l$. If $L \neq Z_{u,\hat{G}}$, then $\mathscr{H} \subseteq_\exists \hat{G}^x_u + \hat{S}(L)$. Since $\hat{G}^x_u + \hat{S}(L) \subseteq \hat{G}^x_u + (\hat{G}^u_x + K_{1,l-1})^u = \hat{G} + K_{1,l-1}$, it follows that $\mathscr{H} \subseteq_\exists G + K_{1,l-1}$, and $h_2(\hat{G}) \leqslant l$. $\square$

We now describe how to perform step 4 of algorithm MaxSubforest. To simplify the notation, we show how to build the set $\mathscr{C}^f_v$ for $v = r$. Building the set for the other vertices of G is done in the same way. Let $u_1, \ldots, u_t$ be the children of v. We use the following idea to build $\mathscr{C}^f_v$: For each l, we will build a set X_l such that $h_2(G - X_l) \geqslant l$ and $h_1(G - X_l)$ is minimal. To build X_l, we split it into two sets A and B, where A contains the edges of X_l that are incident with v, and B contains the rest of the edges. Suppose that we already found B and we want to choose A. From Lemma 5.2 we need to take A that is a deletion set of $(S_{G-B}, [f]_l, [g_{G-B}]_l)$. We will show in Lemma 5.3 that taking A to be maximal deletion set of $(S_{G-B}, [f]_l, [g_{G-B}]_l)$ will make $h_1(G - (A \cup B))$ minimal. The opposite question is how to choose the set B assuming that A was chosen. In Lemma 5.4 we show that we need to choose B such that $h_1(G^v_{u_i} - B)$ is minimal for all i.

Lemma 5.3. *Let B be a set of edges in G that are not incident with v such that $\mathscr{H} \not\subseteq_\exists G - B$.* Let $A, A' \subseteq S^e_G$ *be two sets of edges. If $A' \preccurlyeq A$, then $h_1(G - (A \cup B)) \subseteq h(G - (A' \cup B))$.*

Proof. Fix $J \in h_1(G - (A \cup B))$. Let H be a subtree of $(G - (A \cup B)) + J$ which is isomorphic to a tree from $\mathscr{H}$, and let G_H and J_H be the rooted subtrees of G and J, respectively, that are induced by the vertices of H. From the fact that $c_{J_H} \leqslant c_J \leqslant h_2(G - (A \cup B)) - 2$ we have that J_H is not a rooted star. Therefore, the height of G_H is at most 2, so $G_H = \hat{S}(L)$ for some weighted set L. From Claim 5.1 $L \preccurlyeq S_{G-B} - A$. Since we have that $S_{G-B} - A \preccurlyeq S_{G-B} - A'$ (Lemma 4.1) and $\preccurlyeq$ is transitive, it follows that $L \preccurlyeq S_{G-B} - A'$. By Claim 5.1, $G_H \subseteq_R G - (A' \cup B)$. Thus, $\mathscr{H} \subseteq_\exists (G - (A' \cup B)) + J$, namely $J \in h(G - (A' \cup B))$. $\square$

Lemma 5.4. *Let A be a set of edges in G that are incident with v. Let $B_1, \ldots, B_t, B'_1, \ldots, B'_t$ be sets of edges, where $B_i \subseteq E(G^v_{u_i})$ and $B'_i \subseteq E(G^v_{u_i})$ for all i. If $h_1(G^v_{u_i} - B_i) \subseteq h(G^v_{u_i} - B'_i)$ and $h_2(G^v_{u_i} - B_i) > t + 1 - |A|$ for every $u_i \notin A$, and $\mathscr{H} \not\subseteq_\exists G^v_{u_i} - B_i$ for every $u_i \in A$, then $h_1(G - (A \cup B_1 \cup \cdots \cup B_l)) \subseteq h(G - (A \cup B'_1 \cup \cdots \cup B'_l))$.*

Proof. Let J be a tree in $h_1(G - (A \cup B_1 \cup \cdots \cup B_l))$, and let H be a subgraph of $(G - (A \cup B_1 \cup \cdots \cup B_l)) + J$ which is isomorphic to a tree in $\mathscr{H}$. Suppose that $u_1 \notin A$. Let H_1 be the rooted tree obtained by taking H, removing the vertices of H that are in $G^v_{u_1}$, and making v the root. If H does not contain vertices from $G^v_{u_1}$, or contains only the root of $G^v_{u_1}$, then clearly $\mathscr{H} \subseteq_\exists (G^v_{u_1} - B'_1) + H_1$. Otherwise, $H_1 \in h(G^v_{u_1} - B_1)$ and $c_{H_1} \leqslant t - |A| \leqslant h_2(G^v_{u_1} - B_1) - 2$. Therefore $H_1 \in h_1(G^v_{u_1} - B_1)$. Thus, we have that $H_1 \in h(G^v_{u_1} - B'_1)$, so $\mathscr{H} \subseteq_\exists (G^v_{u_1} - B'_1) + H_1 \subseteq (G - (A \cup B'_1 \cup B_2 \cup \cdots \cup B_l)) + J$. Therefore, $J \in h(G - (A \cup B'_1 \cup B_2 \cup \cdots \cup B_l))$.

If $u_1 \in A$, then H does not contain vertices from $G^v_{u_1}$, so again we have that $J \in h(G - (A \cup B'_1 \cup B_2 \cup \cdots \cup B_l))$. Repeating these arguments gives that $J \in h(G - (A \cup B'_1 \cup \cdots \cup B'_l))$. $\square$

We now define $\mathscr{C}^f_v = \{A_1 \cup B, \ldots, A_r \cup B\}$, where the sets $A_1, \ldots, A_r$ and B will be defined later. The set $\mathscr{C}^f_v$ is ordered according to the indices of the sets A_i, namely the order is $A_1 \cup B, A_2 \cup B, \ldots, A_r \cup B$. The set $\mathscr{C}_v$ will also be ordered, and the ordering of its elements will be according to the order of $\mathscr{C}^f_v$. Since the same process was used by algorithm MaxSubforest for building the sets $\mathscr{C}_{u_1}, \ldots, \mathscr{C}_{u_t}$, then each set $\mathscr{C}_{u_i}$ is ordered (recall that algorithm MaxSubforest builds the set $\mathscr{C}_{u_1}, \ldots, \mathscr{C}_{u_t}$ before building $\mathscr{C}^f_v$).

For $i = 1, \ldots, t$ and $k = 0, \ldots, t$, let B^k_i be the first set from $\mathscr{C}_{u_i}$ (according to the order of $\mathscr{C}_{u_i}$) such that $h_2(G^v_{u_i} - B^k_i) \geqslant t - k + 2$. If no such set exists, then B^k_i is the first set from $\mathscr{C}_{u_i}$. Let $B^k = \cup^t_{i=1} B^k_i$.

By Lemma 3.1, the set $\mathscr{C} = \{A \cup C_1 \cup \cdots \cup C_t : A \subseteq S_G, C_1 \in \mathscr{C}_{u_1}, \ldots, C_t \in \mathscr{C}_{u_t}\}$ is a full set of G. Let $\mathscr{C}' \subseteq \mathscr{C}$ be some complete set of G. From Lemma 3.5, all the sets in $\mathscr{C}'$ have the same number of edges, and for every i, all the sets in $\mathscr{C}_{u_i}$ have the same number of edges. Thus, all the sets in $\mathscr{C}'$ have the same number of edges incident with v, and denote this number by j. We denote $B_i = B^j_i$ and $B = B^j$.

Let A_l be a maximum deletion set of $(S_{G-B}, [f]_{l-1}, [g_{G-B}]_{l-1})$ of size j, if such set exists. If such set does not exist, we say that A_l is undefined. Define $r = \min(n, \max\{l : A_l \text{ is defined}\})$. Note that for every $l < r$, a deletion set of $(S_{G-B}, [f]_{r-1}, [g_{G-B}]_{r-1})$ is also a deletion set of $(S_{G-B}, [f]_{l-1}, [g_{G-B}]_{l-1})$, so A_l is defined.

Lemma 5.5. *$\mathscr{C}^f_v$ is a full set of G.*

Proof. We need to show that for every set $O \in \mathscr{C}'$ there is a set $A_l \cup B$ such that $G - O \leqslant'_P G - (A_l \cup B)$. We will show this using Lemmas 5.3 and 5.4: We will first take the set O and replace its edges that are not incident with v by the edges of B. Lemma 5.4 will give us that $G - O \leqslant'_P G - O'$, where O' is the new set. Then we will take O' and replace the edges of O' that are incident with v by the edges of A_l. Note that the set we obtain is $A_l \cup B$. We will get from Lemma 5.3 that $G - O' \leqslant'_P G - (A_l \cup B)$.

Let O be some set from $\mathscr{C}'$. Denote $A = O \cap S_G$, and $O_i = O \cap E(G^v_{u_i})$ for $i = 1, \ldots, t$. Note that $O = A \cup O_1 \cup \cdots \cup O_t$ and $O_i \in \mathscr{C}_{u_i}$ for all i. W.l.o.g. suppose that $A = \{(v, u_{t-j+1}), \ldots, (v, u_t)\}$.

We consider two cases. In the first case, suppose that $h_2(G - O) > 1$. Denote $l = h_2(G - (A \cup B))$. We will show that $G - O \leqslant_P G - (A_l \cup B)$. Since $|O| = |A_l \cup B|$, it will follow that $G - O \leqslant'_P G - (A_l \cup B)$. To show that $G - O \leqslant_P G - (A_l \cup B)$, we will show that $G - O \leqslant_P G - (A \cup B)$ and $G - (A \cup B) \leqslant_P G - (A_l \cup B)$.

We first show that $G - O \leqslant_P G - (A \cup B)$. By Lemma 3.4, it suffices to show that $h_1(G - (A \cup B)) \subseteq h(G - O)$ and $h_2(G - (A \cup B)) \geqslant h_2(G - O)$. Consider some index $i \leqslant t - j$. Recall that $\mathscr{C}_{u_i} \subseteq \mathscr{C}^f_{u_i}$ and $\mathscr{C}^f_{u_i} = \{A'_l \cup B' : l \leqslant r'\}$ for some sets $A'_1, \ldots, A'_{r'}$ and B'. By the definition of h_2, we have that $h_2(G^v_{u_i} - O_i) \geqslant t - j + 2$. Therefore, the set B_i appears before O_i in the order of $\mathscr{C}_{u_i}$ and $h_2(G^v_{u_i} - B_i) \geqslant t - j + 2$. In other words, $B_i = A'_l \cup B'$ and $O_i = A'_{l'} \cup B'$ for some $l < l'$. By definition, every deletion set of $(S_{G^v_{u_i} - B'}, [f]_{l'-1}, [g_{G^v_{u_i} - B'}]_{l'-1})$is a deletion set of $(S_{G^v_{u_i} - B'}, [f]_{l-1}, [g_{G^v_{u_i} - B'}]_{l-1})$. In particular, $A'_{l'}$ is a deletion set of $(S_{G^v_{u_i} - B'}, [f]_{l-1}, [g_{G^v_{u_i} - B'}]_{l-1})$. From the maximality of A'_l we have that $A'_{l'} \preccurlyeq A'_l$. By Lemma 5.3, $h_1(G^v_{u_1} - B_i) \subseteq h(G^v_{u_i} - O_i)$. Since the previous inequality is true for all i, by Lemma 5.4, $h_1(G - (A \cup B)) \subseteq h(G - O)$.

We now show that $h_2(G - (A \cup B)) \geqslant h_2(G - O)$. By Lemma 5.2, $h_2(G - (A \cup B)) = \alpha(S_{G-B} - A, f, g_{G-B})$ and $h_2(G - O) = \alpha(S_{G-O} - A, f, g_{G-O})$. Clearly, for some index $i \leqslant t - j$, a set $L \in g_{G-B}(u_i)$ does not influence the value of $\alpha(S_{G-O}-A,\, f,\, g_{G-O})$ (since $L\not\preceq S_{G-B}-\{u_i\}$). Therefore, $\alpha(S_{G-B}-A,\, f,\, g_{G-B})=\alpha(S_{G-B}-A,\, f,\, g'_{G-B})$, where the mapping $g'_{\hat{G}}$ (for some rooted forest $\hat{G}$) is defined by $g'_{\hat{G}}(u)=\{L\in g_{\hat{G}}(u): |L|\leqslant c_{\hat{G}}\}$. Similarly, $\alpha(S_{G-O}-A,\, f,\, g_{G-O})=\alpha(S_{G-O}-A,\, f,\, g'_{G-O})$.

As $h_1(G^r_{u_i}-B_i)\subseteq h(G^r_{u_i}-O_i)$ and $h_2(G^r_{u_i}-O_i)\geqslant t-j+2$ for every $i\leqslant t-j$, we have that $g'_{G-B}(u_i)\subseteq g'_{G-O}(u_i)$ for every $i\leqslant t-j$. Furthermore, by Lemma 3.5 all the sets in each set $\mathscr{C}_{u_i}$ have the same number of edges incident with u_i. Thus, $S_{G-B}=S_{G-O}$. Therefore, $\alpha(S_{G-B}-A,\, f,\, g'_{G-B})\geqslant\alpha(S_{G-O}-A,\, f,\, g'_{G-O})$. It follows that $h_2(G-(A\cup B))\geqslant h_2(G-O)$. We conclude that $G-O\leqslant_P G-(A\cup B)$.

Finally, we show that $G-(A\cup B)\leqslant_P G-(A_l\cup B)$. Again, we will show that $h_1(G-(A_l\cup B))\subseteq h(G-(A\cup B))$ and $h_2(G-(A_l\cup B))\geqslant h_2(G-(A\cup B))=l$. By Lemma 5.2, we have that A is a deletion set of $(S_{G-B}, [f]_{l-1}, [g_{G-B}]_{l-1})$, so A_l is defined. By the maximality of A_l we get that $A\preccurlyeq A_l$. Moreover, by Lemma 5.2, $h_2(G-(A_l\cup B))\geqslant l$. Therefore, $G-(A\cup B)\leqslant_P G-(A_l\cup B)$.

If $h_2(G-O)=1$ then $h(G-O)=F-\{G_\emptyset\}$. From Lemma 5.2, $A_1\cup B\in\mathscr{C}^f_v$ is an optimal deletion set of G, and we have that $G-O\leqslant'_P G-(A_1\cup B)$. □

In order to build the set $\mathscr{C}^f_v$ in polynomial time, we need to show how to compute the value of j. The following lemma gives an efficient way for computing this value.

Lemma 5.6. $j=\min\{\mathrm{OPT}(S_{G-B^k}, f, g_{G-B^k}): k=0,\dots,t\}$.

Proof. Let O be some set from $\mathscr{C}_v$. From Lemma 3.5, $G-O$ is a maximum subforest of G with property P. By definition, $|O|=j+|B^j|$. Fix some $k\leqslant t$. From Lemma 5.2, if a set X is a deletion set of $(S_{G-B^k}, f, g_{G-B^k})$, then $G-(X\cup B^k)$ has property P, so $|X|+|B^k|\geqslant|O|=j+|B^j|$. By Lemma 3.5, the sets $B^0,\dots,B^t$ have the same size. Therefore, $|X|\geqslant j$ and since this is true for all k, we have that $\min\{\mathrm{OPT}(S_{G-B^k}, f, g_{G-B^k}): k=0,\dots,t\}\geqslant j$. Moreover, by Lemma 5.2, $\mathrm{OPT}(S_{G-B^j}, f, g_{G-B^j})=j$, and the lemma follows. □

From Lemmas 5.5 and 5.6, the following theorem follows.

Theorem 5.7. *The problem 5-TEDP can be solved in polynomial time.*

We note that we can give an implementation of the algorithm for 5-TEDP whose time complexity is $\mathrm{O}(pn^3)$. Using similar ideas, 4-TEDP can be solved in $\mathrm{O}(pn)$ time.

We now give the key idea of the algorithm for TEDP_1. As the algorithm is similar to the algorithm for 5-TEDP, we omit the details.

Define $\hat{P}(\{l_1,\dots,l_d\})=\oplus_{1\dots1}(\hat{P}_{l_1},\dots,\hat{P}_{l_d})$, and let $P(\{l_1,\dots,l_d\})$ be the unrooted tree formed from $\hat{P}(\{l_1,\dots,l_d\})$. Every tree with at most one vertex with degree more than 2 has a representation of the form $P(\{l_1,\dots,l_d\})$ for some $l_1,\dots,l_d$. Let $\mathscr{L}$ be the set of all the representations of the trees in the obstruction set.

Define

$$h_2(G)=\min(\{i\geqslant 0:\mathscr{H}\subseteq_\exists G+\hat{P}_i\}\cup\{\infty\}),$$

where $\hat{P}_0=G_\emptyset$.

Assume that G^v is a rooted tree satisfying $\mathscr{H}\not\subseteq_\exists G-v$ and let $u_1,\dots,u_t$ denote the children of v. Let S_G be a weighted set $\{(v,u_1),\dots,(v,u_t)\}$, where the weight of (v,u_i) is the height of $G^v_{u_i}$. We define mappings f and g_G by $f(u_i)=\mathscr{L}$ and $g_G(u_i)=\{\{h_2(G^v_{u_i})-1\}\}$. We have that $h_2(G)=\alpha(S_G, f, g_G)$. From this fact, the following theorem follows.

Theorem 5.8. *The problem TEDP_1 can be solved in polynomial time.*

6. Hardness of the TEDP

In this section we prove several hardness results for the problem. We will show that several variants of TEDP are NP-hard, and moreover, they are hard to approximate.

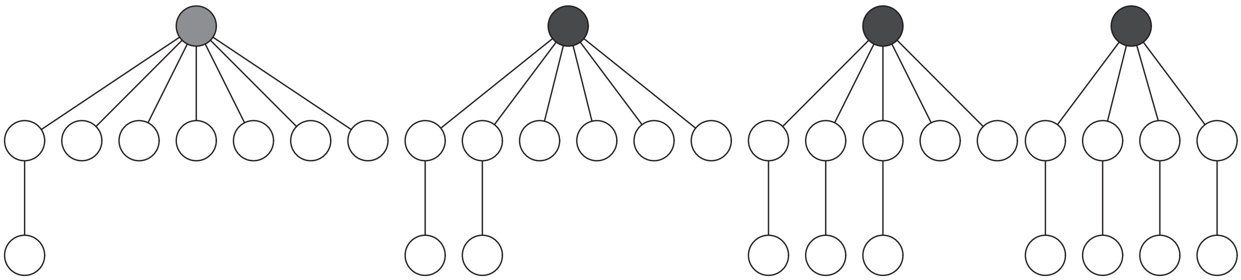

Fig. 7. The constructions of the rooted trees $T_1, \ldots, T_n$ in case 1 for $n = 4$.

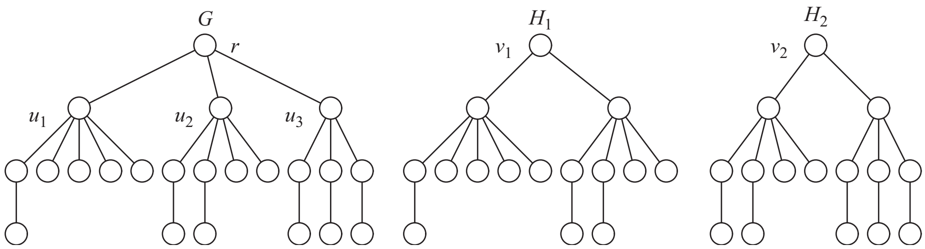

Fig. 8. The reduction of case 1 for the input $R_1 = \{1, 2\}$, $R_2 = \{2, 3\}$.

Theorem 6.1. *If* P $\neq$ NP, *then there is a constant* c *such that there is no polynomial-time approximation algorithm for TEDP with approximation factor less than* $c \log k$. *This result holds even under each of the following restrictions of TEDP*:

1. *G and each tree in $\mathscr{H}$ has diameter* 6.
2. *$\mathscr{H}$ consists of one tree with diameter* 8.

Furthermore, if P $\neq$ NP *then there is a constant* c' *such that there is no polynomial-time approximation algorithm with approximation factor less than* $1 + c'$ *for the following restrictions of TEDP*:

3. *Each tree in $\mathscr{H}$ has maximum degree of* 3 *and at most two vertices with degree* 3.
4. *$\mathscr{H}$ consists of one tree with maximum degree of* 3.

Note that the first restriction is a special case of 6-TEDP, and the third restriction is a special case of TEDP_2.

Proof. All the reductions in this proof are from Hitting Set or a restriction of this problem. The input to the hitting set problem is a collection of sets $R_1, \ldots, R_m$ which are subsets of $S = \{1, \ldots, n\}$. The goal is to find a minimal subset $U \subseteq S$ such that $U \cap R_i \neq \emptyset$ for $i = 1, \ldots, m$. We can assume w.l.o.g. that each R_i has at least two elements. It was shown in [13] that if P $\neq$ NP then there is a constant c_0 such that there is no approximation algorithm for Hitting Set with approximation factor less than $c_0 \log n$.

We first prove case 1. Given an input $R_1, \ldots, R_m$ to the hitting set problem, we build rooted trees $T_1, \ldots, T_n$ by taking $T_i = B_{2n-2i,i}$, where $B_{x,y}$ is the rooted tree obtained by taking x copies of $\hat{P}_1$ and y copies of $\hat{P}_2$, adding a new vertex and connecting it by edges to all the roots of the trees, and making the new vertex the new root (see Fig. 7). Note that $T_i \nsubseteq_R T_j$ for every $i \neq j$, and all the trees have height 2. For every set R_i, we build a tree H_i by taking a vertex named v_i, and a copy of the tree T_j for every $j \in R_i$, and adding edges between the roots of these trees and v_i. We define $\mathscr{H} = \{H_1, \ldots, H_m\}$. We build the tree G by taking the trees $T_1, \ldots, T_n$, adding a vertex r, and adding edges between r and the roots of $T_1, \ldots, T_n$. Denote by $u_1, \ldots, u_n$ the roots of the trees $T_1, \ldots, T_n$ in G, respectively. See Fig. 8 for an example of this reduction. Clearly, G and the trees in $\mathscr{H}$ have diameter 6.

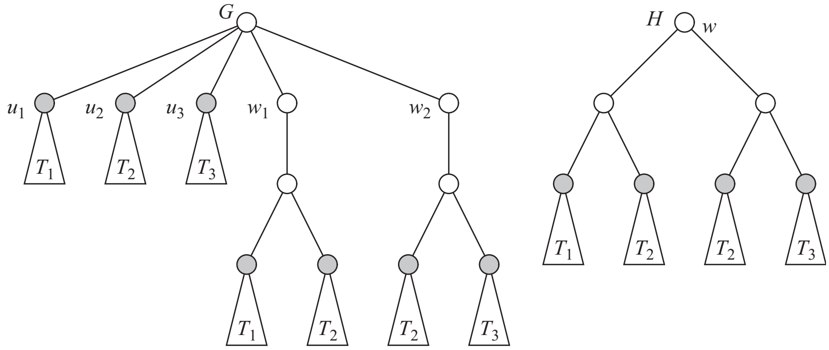

Fig. 9. The reduction of case 2 for the input $R_1 = \{1, 2\}$, $R_2 = \{2, 3\}$. The heavy vertices are highlighted.

For each $i \leqslant m$, we denote by G_i the subtree induced from G by the vertex r and the vertices of T_j for all $j \in R_i$. Clearly, G_i is isomorphic to H_i. Moreover, we claim that G_i is the only subgraph of G which is isomorphic to H_i. To show this claim, suppose that G' is a subtree of G which is isomorphic to H_i. We now argue which vertices in G can match v_i under the isomorphism. Since the trees $T_1, \ldots, T_n$ have height 2, it follows that in H_i the vertex v_i is the center of a path of length 6. Furthermore, r is the only vertex in G with this property. Therefore, G' must contain r, and the isomorphism between H_i and G' matches v_i to r. Each neighbor v of v_i is a root of a copy of some tree T_j, and is matched by the isomorphism to the root of some $T_{j'}$. Since $T_j \nsubseteq_R T_{j'}$ for $j \neq j'$, it follows that $j = j'$, and therefore $G' = G_i$. This completes the proof of the claim.

Thus, for a set A of edges of G that are incident on r, we have that $H_i \nsubseteq G - A$ iff A contains an edge (r, u_j) for some $j \in R_i$. Therefore, given a hitting set U of $T_1, \ldots, T_m$ of size k, the set $\{(r, u_i) : i \in U\}$ is a deletion set of $(G, \mathscr{H})$. Conversely, let A be a deletion set of $(G, \mathscr{H})$ of size k, and suppose that A contains an edge $e = (u, v)$ which is not incident on r. Let w be vertex after r on the path from r to u in G. Then, $A \cup \{(r, w)\} - \{e\}$ is also a deletion set of $(G, \mathscr{H})$. By repeating this argument, we obtain a deletion set A' of $(G, \mathscr{H})$ such that all the edges in A' are incident on r and $|A'| \leqslant |A|$. Then, $\{i : (r, u_i) \in A'\}$ is a hitting set of $T_1, \ldots, T_m$ of size k. The correctness of the reduction follows.

We now deal with case 2. Given subsets $R_1, \ldots, R_m$, we build tree $T_1, \ldots, T_n$ where $T_i = B_{2n-2i+m+1,i}$. Then, we build trees $H_1, \ldots, H_m$ using $T_1, \ldots, T_n$ as in case 1. We build a tree H by taking a vertex w, and the trees $H_1, \ldots, H_m$, and adding an edge between w and v_i for every $i \leqslant m$. Let $\mathscr{H} = \{H\}$. For every $i \leqslant m$, define $H^i = H_w^{v_i}$, i.e., H^i is the rooted tree formed by taking H, removing the vertices of H_i, and choosing w as the root. The tree G is built by taking a vertex named r, the trees $T_1, \ldots, T_n$, and the trees $H^1, \ldots, H^m$ and adding edges between r and the roots of $T_1, \ldots, T_n, H^1, \ldots, H^m$. Denote by $u_1, \ldots u_n$ the roots of $T_1, \ldots, T_n$ in G and by $w_1, \ldots, w_m$ the roots of $H^1, \ldots, H^m$. See Fig. 9 for an example. Note that H has diameter 8 and G has diameter 10.

For each $i \leqslant m$, we denote by G_i the subtree induced from G by the vertex r, the vertices of H^i, and the vertices of T_j for all $j \in R_i$. Each subtree G_i is isomorphic to H. We claim that no other subtree of G is isomorphic to H. The correctness of the reduction follows from this claim in the same fashion as in the proof of case 1. We now prove the claim: let G' be a subtree of G which is isomorphic to H. We say that a vertex is *heavy* if its degree is at least $m + n + 1$. Clearly, the isomorphism between H and G' must map a heavy vertex in H to a heavy vertex in G'. In H, the heavy vertices are those with distance 2 from w, or in other words, the roots of all the copies of $T_1, \ldots, T_n$ in H. In G, the heavy vertices are $u_1, \ldots, u_n$ and descendents of w_i with distance 2 from w_i for $i = 1, \ldots, m$, or in other words, the roots of all the copies of $T_1, \ldots, T_n$ in G. We now argue which vertices in G can match w under the isomorphism. w is the center of a path of length 4 whose end vertices are heavy. The only vertices in G with this property are $u_1, \ldots, u_m$. Therefore, the isomorphism between H and G' maps w to some vertex w_i. Furthermore, the heavy vertices with distance 2 from w_i are $u_1, \ldots, u_n$ and the descendents of w_i with distance 2 from w_i. It follows that $G' = G_i$.

For case 3, we give a reduction from a restriction of the hitting set problem in which the sets $R_1, \ldots, R_m$ have size exactly 2. This problem is equivalent to Vertex Cover, and therefore, if P $\neq$ NP it cannot be approximated within a factor of 1.166 [9]. Given sets $R_1, \ldots, R_m$ and a positive integer k, we build trees $T_1, \ldots, T_n$ as follows: the tree T_i is built by taking a copy of the tree $\hat{P}_{n+1}$ (the rooted path on $n + 1$ vertices) and adding a new vertex which is connected

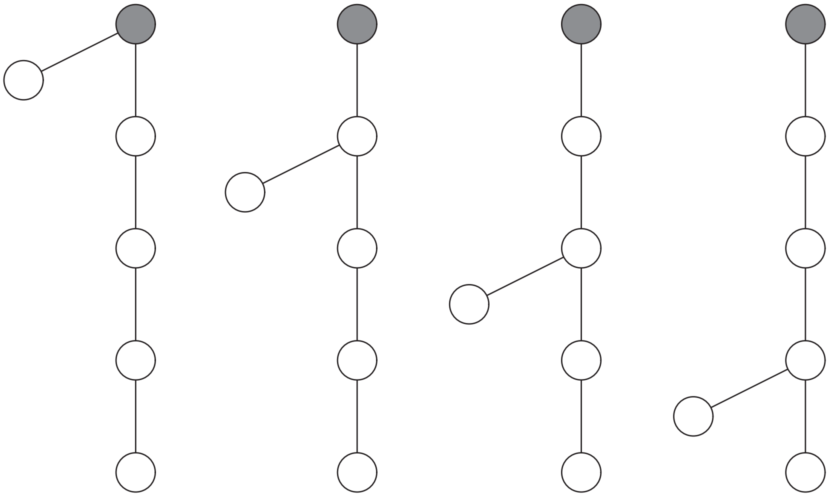

Fig. 10. The constructions of the rooted trees $T_1, \ldots, T_n$ in case 3 for $n = 4$.

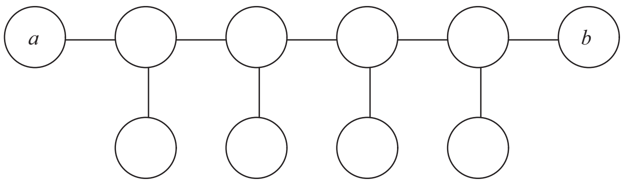

Fig. 11. The tree J.

to the vertex at distance $i-1$ from the root. See Fig. 10 for an example. We then generate $\mathscr{H} = \{H_1, \ldots, H_m\}$ and G from $T_1, \ldots, T_n$ like in the reduction of case 1. Clearly, as each T_i has one vertex of degree 3 and the rest of its vertices have degree at most 2, the restrictions are satisfied. The proof of the correctness of the reduction is similar to the proof in case 1, and thus is omitted.

We also provide a reduction from Vertex Cover in case 4. We begin by building trees $T_1, \ldots, T_n$ as in the construction of case 3. For each $i \leqslant n$ we build a tree T_i' by taking T_i and a copy for the tree J which is given in Fig. 11, adding an edge between the root of T_i and a, and making b the new root. We build the trees $H_1, \ldots, H_m$ from $T_1', \ldots, T_n'$ like in case 1. Then, we take a path of length $2m-1$ whose vertices are $w_1, \ldots, w_{2m-1}$. We add an edge $w_{2i-1}v_i$ for every $i \leqslant m$. The result is the tree H, and $\mathscr{H} = \{H\}$. For $i \leqslant m$, define $H^i = H^{v_i}_{w_{2i-1}}$. We build the tree G from $T_1', \ldots, T_n'$ and $H^1, \ldots, H^m$ like in case 2. Note that all the subgraphs of H and G that are isomorphic to J are due to copies of $T_1', \ldots T_n'$ in H and G. Hence the copies of J play the same role of restricting the isomorphism as the heavy vertices in case 2. The correctness of the reduction follows from this fact, as in case 2. $\square$

7. Concluding remarks and open problems

We have shown sharp boundaries on the tractability of TEDP: 5-TEDP and TEDP$_1$ can be solved in polynomial time, while 6-TEDP and TEDP$_2$ are NP-hard.

As described in Section 3, our algorithms are based on quasiorders. Previous papers use the following equivalence relations: for a property P, the equivalence relation $\sim_P$ is defined by

$$G \sim_P G' \iff P(G+J) = P(G'+J) \quad \text{for every rooted tree } J.$$

Let P be the property of not containing a tree from $\mathscr{H}$ as an induced subforest, where $\mathscr{H}$ consists of trees with diameter at most 5. As discussed in Section 5, we can show that $|\leqslant_P'| \leqslant 2|\mathscr{H}|+1$. On the other hand, the number of equivalence classes of $\sim_P$ can be $\Omega(2^{2^{|\mathscr{H}|}})$ (we omit the details). It would be interesting to find other graph problems for which our technique yields faster algorithms. In other words, are there other graph properties (on bounded treewidth graphs, or on some restricted family of graphs) for which there is a large gap between $|\leqslant|$ for some (P, Φ)-order $\leqslant$, and the number of equivalence classes in $\sim_P$?

Acknowledgment

We thank Ron Shamir for helpful discussions.

References

[1] S. Arnborg, B. Courcelle, A. Proskurowski, D. Seese, An algebraic theory of graph reduction, J. Assoc. Comput. Mach. 40 (1993) 1134–1164.
[2] S. Arnborg, J. Lagergren, D. Seese, Problems easy for tree-decomposable graphs, J. Algorithms 12 (1991) 308–340.
[3] M.W. Bern, E.L. Lawler, A.L. Wong, Linear-time computation of optimal subgraphs of decomposable graphs, J. Algorithms 8 (2) (1987) 216–235.
[4] H.L. Bodlaender, Dynamic programming on graphs with bounded treewidth, in: Proceedings of the 15th International Colloquim Automata, Languages and Programming, Lecture Notes in Computer Science, vol. 317, Springer, Berlin, 1988, pp. 105–118.
[5] R.B. Borie, R.G. Parker, C.A. Tovey, Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursively constructed graphs, Algorithmica 7 (1992) 555–581.
[6] B. Courcelle, The monadic second-order logic of graphs I: recognizable sets of finite graphs, Inform. Comput. 85 (1990) 12–75.
[7] R. Downey, M. Fellows, Parameterized Complexity, Springer, Berlin, 1997.
[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., San Francisco, 1979.
[9] J. HAAstad, Some optimal inapproximability results, in: Proceedings of the 29th Symposium on the Theory of Computing (STOC 97), ACM Press, New York, 1997, pp. 1–10.
[10] S. Mahajan, J.G. Peters, Algorithms for regular properties in recursive graphs, in: Proceedings of the 25th Allerton Conference on Communications, Control and Computing, 1987, pp. 14–23.
[11] J. Matoušek, R. Thomas, On the complexity of finding iso- and other morphisms for partial *k*-trees, Discrete Math. 108 (1992) 343–364.
[12] D.W. Matula, An algorithm for subtree identification, SIAM Rev. 10 (1968) 273–274.
[13] R. Raz, S. Safra, A sub-constant error-probability low-degree test. and a sub-constant error-probability PCP characterization of NP, in: Proceedings of the 29th Symposium on the Theory of Computing (STOC 97), ACM Press, New York, 1997, pp. 475–484.
[14] P. Scheffler, D. Seese, A combinatorial and logical approach to linear-time computability, in: Proceedings of the European Conference on Computer Algebra, Lecture Notes in Computer Science, vol. 378, Springer, Berlin, 1989, pp. 379–380.
[15] R. Shamir, D. Tsur, The maximum subforest problem: approximation and exact algorithms, in: Proceedings of the Ninth Symposium on Discrete Algorithms (SODA 98), ACM Press, New York, 1998, pp. 394–399.
[16] K. Takamizawa, T. Nishizeki, N. Saito, Linear-time computability of combinatorial problems on series-parallel graphs, J. Assoc. Comput. Mach. 29 (1982) 623–641.
[17] C. Wang, A subgraph problem from restriction maps of DNA, J. Comput. Biol. 1 (3) (1994) 227–234.
[18] M.S. Waterman, J.R. Griggs, Interval graphs and maps of DNA, Bull. Math. Biol. 48 (1986) 189–195.