The maximum subforest problem: Approximation and exact algorithms

Ron Shamir* Dekel Tsur*

Abstract

We study the *maximum subforest problem*: Given a tree G and a set of trees $\mathcal{H}$, find a subgraph G' of G such that G' does not contain a subtree isomorphic to a tree from $\mathcal{H}$, and the number of edges in G' is maximum. We give a polynomial time approximation scheme for this problem. We also give an exact algorithm for this problem whose time complexity is $2^{O(k^2/\log k)}n$, where n is the number of vertices in G, and k is the total number of vertices in $\mathcal{H}$.

1 Introduction

A very common problem in algorithmic graph theory and combinatorial optimization is to find in a given graph an optimal subgraph. Examples include the maximum matching, the maximum clique and the traveling salesman problems. For a graph property P, the *maximum subgraph problem with respect to* P is to find, in a given graph $G = (V, E)$, a largest possible subset of edges $E' \subseteq E$ such that (V, E') satisfies P. Such problems are also called *edge deletion problems*, as they can be formulated as finding a minimum subset O of the edges such that $G - O$ satisfies P.

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* $\mathcal{H}_P$ of all minimal graphs that do not satisfy P: A graph satisfies P if and only if it does not contain any graph from $\mathcal{H}_P$ as an induced subgraph.

Many maximum subgraph problems are NP-hard. However, when restricting the input graph, some problems become polynomial. In particular, it has been shown that many problem are polynomial on trees (e.g., [15, 18]) or on series-parallel graphs (e.g., [17]). Takamizawa et al. [26] gave a general result that states that for a hereditary property P with a finite obstruction set, the maximum

*Dept. of Computer Science, Tel-Aviv University. E-Mail:{rshamir,dekelts}@tau.ac.il

subgraph problem can be solved in linear time on series-parallel graphs. While there are many graph properties that satisfy this condition, there are many other properties which are not hereditary (e.g., having a Hamiltonian cycle), or are hereditary but have an infinite obstruction set (e.g., the property of being a bipartite graph, whose obstruction set consists of all odd cycles).

Another limitation of the result of Takamizawa et al. is that the family of series-parallel graphs is rather small. Therefore, the question arises whether there are graph families that generalize both trees and series-parallel graphs, and still optimization problem are easy on them. Such a generalization is the family of graph with bounded treewidth. For examples of algorithms for specific problems on graphs with bounded treewidth see [1, 2, 5]. The result given in [7, 20, 22, 24, 23] can be viewed as a generalization of the result of Takamizawa et al. to the family of bounded treewidth graphs (in some of these papers, the input graph is also required to have bounded degree). Moreover, the latter results apply to larger classes of properties.

Here is one concrete example for this type of results. Scheffler [22] studied properties P for which there is an additive function f, constants m and c, and a relation P' such that

$$P(G) = \exists G_1, \ldots, G_m \forall v \, P'(G|_{N_c(v)}, G_1|_{N_c(v)}, \ldots, G_m|_{N_c(v)}, v) \wedge f(G_1) \geq B,$$

where $N_c(v)$ is the set of all vertices that are at distance at most c from v (including v). Scheffler showed that for such properties, deciding if a graph G has property P can be done in linear time on graphs with bounded treewidth and bounded degree. In particular, by defining $f(G)$ to be the number of edges in G, this result shows that for many graph properties, the corresponding maximum subgraph problem is polynomial. Among these properties are all hereditary properties with a fixed obstruction set, and bipartiteness.

Courcelle [16] studied the *property identification problem*, namely deciding whether a graph G has property P. Courcelle showed that for a property P that is defined by a monadic second order logic formula, the property identification problem can be solved in linear time when the input graph has bounded treewidth. A *monadic second order formula* is a logic formula that is built from logic connectives $(\wedge, \vee, \neg, \rightarrow, \leftrightarrow)$, vertex variables, edge variables, vertex set variables, edge set variables, the existential $(\exists)$ and universal $(\forall)$ quantifiers, the membership symbol $(\in)$, and the equality symbol. For example, bipartiteness can be expressed by the formula

$$\exists W \forall v \forall w \, (v, w) \in E \rightarrow ((v \in W \wedge \neg(w \in W)) \vee (w \in W \wedge \neg(v \in W))).$$

This result has been extended to show that for a property that is defined by a monadic second order logic formula, the maximum subgraph problem can be solved in linear time on graphs with bounded treewidth [4, 13].

The above algorithms are based on finding a tree decomposition of the input graph, and then applying a dynamic programming algorithm on that tree. The bottleneck in the time complexity was the tree decomposition, since at the time when the above papers were written, the best algorithm for finding a tree decomposition were super-linear. Later, Bodlaender gave a linear time decomposition algorithm [10]. A different kind of algorithm was given by Arnborg et al. [3] for the property identification problem: The algorithm is based on graph reductions, and thus, it does not require a tree decomposition.

For discussing parallel complexity, recall that an optimal-speedup algorithm is a parallel algorithm that has the same overall number of operations as the fastest sequential algorithm. The fastest optimal-speedup parallel algorithm for finding a tree decomposition (i.e., one that uses $O(n)$ operations on an n-vertex graph) takes $O(\log^2 n)$ time on an EREW machine [12]. The algorithm of [4] can be used to solve the maximum subgraph problem with the same parallel time complexity. Bodlaender and Hagerup [12] gave an optimal-speedup algorithm for the property identification problem on graphs with bounded treewidth, requiring $O(\log n)$ time on a CRCW machine. The algorithm is based on the reduction algorithm of Arnborg et al. [3]. This algorithm was extended to solve the maximum subgraph problem on graphs with bounded treewidth with the same time complexity [9, 11]. Bodlaender gave a dynamic algorithm for several optimization problems on graphs with treewidth 2 such that each update of the graph takes $O(\log n)$ time, and each query takes $O(1)$ or $O(\log n)$ time [8].

A major problem with the above algorithms is that the constants hidden in the time complexity are extremely large. In order to explicitly consider these constants, we will look at the time complexity as a function of the size of the input graph and the property P, or, in other words, we consider P to be part of the input. We therefore need to define how P is represented in the input. One possible way is to assume that P is represented by a monadic second order logic formula. An alternative approach is to assume that P is a hereditary property, and is represented by its obstruction set (we assume that the obstruction set is finite). Thus, the input has two parts: a query graph and an obstruction set which is a collection of forbidden graphs. In this case, the time complexity of the algorithm is measured as a function of the number of vertices in the query graph plus the total number of vertices in the forbidden graphs. We will concentrate on the case when all the input graphs are trees: Given a tree G and an obstruction set of trees $\mathcal{H}$, the goal is to find a subforest of G that does not contain a subtree isomorphic to any tree from $\mathcal{H}$ and has maximum number of edges. We call this problem the *maximum subforest problem* (MSP). We call the corresponding edge deletion problem the *tree deletion problem* (TDP). The maximum subforest problem is a generalization of the subtree isomorphism problem which was studied in Chapter **??**.

Using the approach of Takamizawa et al. [26], the maximum subforest problem

can be solved in $2^{2^{O(k)}}n$ time, where n is the number of vertices in G, and k is the total number of vertices in the trees in $\mathcal{H}$.

1.1 New results

We show that MSP is NP-hard. We also give a $2^{O(k^2/\log k)}n$-time algorithm for MSP, improving the result implied by the work of Takamizawa et al.

Next, we study the approximation problem. While MSP and TDP are equivalent if an exact solution is required, approximating the former is easier: We show that TDP is hard to approximate within $c \log k$ factor for some constant c, and in contrast, we give a polynomial time approximation scheme for MSP.

We note that some restrictions of MSP are tractable. In [27], we show, for example, that MSP is polynomial if each tree in $\mathcal{H}$ has diameter of at most 5. This result is best possible as it is shown in this thesis that MSP is NP-hard when $\mathcal{H}$ contains trees with diameter 6. In fact, we give stronger results that imply hardness of approximation for TDP when $\mathcal{H}$ contains trees with diameter 6.

The paper is organized as follows: In Section 2 we give a general framework for exact algorithms for solving MSP. Section 3 describes a combinatorial lemma. In Section 4 we analyze the complexity of the MSP algorithm using the combinatorial lemma. In Section 5 we use the lemma and the exact algorithm to give a polynomial time approximation scheme for MSP.

We conclude this section with some definitions. A *rooted forest* is a collection of trees with one distinguished vertex which is called the root. We will sometime write G^r to denote the rooted forest G with root r. Also, for an unrooted forest G, we denote by G^r the rooted forest formed by choosing the vertex r to be 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 J^r must have the same root as G^r). For a forest (rooted or unrooted) G and an unrooted tree H, we write $H \subseteq G$ if H is isomorphic to a subgraph of G. For a forest G and a set of trees $\mathcal{H}$ we write $\mathcal{H} \subseteq G$ if $H \subseteq G$ for some $H \in \mathcal{H}$. If $\mathcal{H} \not\subseteq G$ we say that G is *$\mathcal{H}$-free*.

For a vertex v in the graph $G = (V, E)$ define $N(v) = \{u : uv \in E\}$. For a set of vertices X, $N(X) = \cup_{x \in X} N(x)$.

For a graph G, $E(G)$ denotes the set of edges of G, and $e(G) = |E(G)|$. We denote the family of all rooted forests by $\mathcal{R}$.

An *l-ary rooted forest operator* is a function 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.

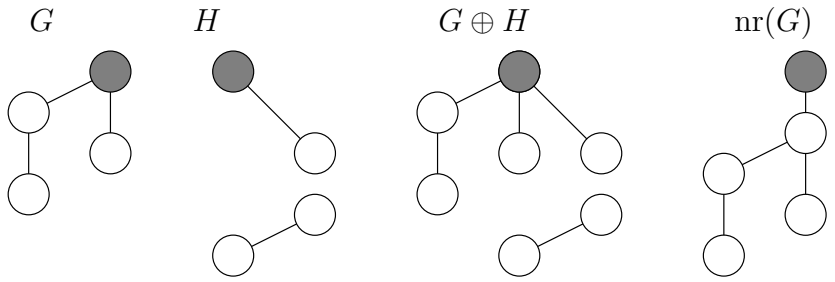

Figure 1: The $\oplus$ and nr operators.

2. Adding new vertices.

3. Adding new edges, where each endpoint of a new edge is either a root of some input forests or a new vertex.

Finally, the root of $f(G_1, \ldots, G_l)$ is either a root of some input forests or a new vertex. For an operator f, let $a(f)$ denote the number of forests on which f operates. For a set of operators Φ, let $a(\Phi) = \max_{f \in \Phi} a(f)$. An operator f is a *suboperator* of an operator f' if $a(f) = a(f')$ and for any $G_1, \ldots, G_{a(f)}$, $f(G_1, \ldots, G_{a(f)})$ is a subgraph of $f'(G_1, \ldots, G_{a(f)})$. For an operator f, let $\mathrm{sub}(f)$ denote the set of all suboperators of f. A set of operators Φ is called *closed* if $\mathrm{sub}(f) \subseteq \Phi$ for any $f \in \Phi$. A set of operators Φ is called *complete* if every rooted forest can be built from copies of the single-node rooted tree by a series of applications of the operators in Φ.

We define some rooted forests operators: Given two rooted forests G and H, $G \oplus H$ is the rooted forest formed by taking G and H and identifying their roots. The nr (new root) operator takes a rooted forest G, adds a vertex s and an edge between s and the root of G, and makes s the new root. The nir (new isolated root) operator takes a rooted forest, adds to it a disconnected vertex and makes it the new root. Note that $\mathrm{sub}(\mathrm{nr}) = \{\mathrm{nr}, \mathrm{nir}\}$. See Figure 1 for examples. The set $\{\oplus, \mathrm{nr}, \mathrm{nir}\}$ is closed and complete.

The tree $K_{1,l}$ is composed by taking l vertices, and a distinguished vertex called the *center* and connecting the center to all other vertices. 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.

Let G be a tree and P be a graph property. We define the predicate $P(G)$ to have value 1 if G has property P, and 0 otherwise.

For a set A, 2^A denotes the set of all subsets of A.

2 A framework for solving MSP

In this section we describe a general method for solving maximum subforest problems based on decomposition. We will use this method in Section 4 to give

an exact algorithm for MSP, and in Section 5 to give an approximation scheme for MSP. The general idea behind our framework is similar to the one used by Bern el al. [6] and others (e.g., [4, 13]), although some aspects are different, as will be explained later.

Before describing the framework, we give some definitions. Let P be some graph property. The *maximum subforest problem with respect to* P is to find, in a given tree $G = (V, E)$, a subforest of G with property P and with the maximum number of edges. In the following we will describe an algorithm for solving this problem under certain assumptions on P.

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 node 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 node in the tree is associated with a rooted forest: A leaf is associated with the forest $\hat{P}_1$ and an internal node is associated with the rooted forest formed by applying the node's operator on the forests associated with the children of the node. The forest associated with the root of the composition tree is isomorphic to G.

We now describe the basic idea of our algorithm. Let G be the input to the maximum subforest problem, and let G' be a forest that corresponds to some node in the decomposition tree of G. We want to create a collection 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 collection. As an example, consider the tree G' in Figure 2, and the property P of not containing a path of length 4. Out of the two subforests $G' - \{a\}$ and $G' - \{b\}$, it is better to take $G' - \{a\}$ into the collection: If there is an optimal solution G^* to MSP such that the subforest of G^* that is induced by the vertices of G' is $G' - \{b\}$, then $G^* \cup \{b\} - \{a\}$ is also an optimal solution. Hence, in this case, only $G' - \{a\}$ and $G' - \{a, b\}$ will be the candidates. The key to the efficiency of the approach is eliminating as many candidate subforests ("pieces") as possible.

We now give a formal description of this idea: We will define partial orders which are used to compare candidates for the collection. Let $\leq$ be some partial order on rooted forests. We say that $\leq$ is *preserved by* Φ if for any $f \in \Phi$, and any $G_1, \ldots, G_{a(f)}, G'_1, \ldots G'_{a(f)}$ such that $G_i \leq G'_i$ for $i = 1, \ldots, a(f)$, there is an operator $f' \in \mathrm{sub}(f)$ such that $f(G_1, \ldots, G_{a(f)}) \leq f'(G'_1, \ldots, G'_{a(f)})$. We say that $\leq$ *preserves* P if $G \leq G'$ implies $P(G) \leq P(G')$. We say that $\leq$ *preserves* P *with size* if $\leq$ preserves P, and additionally, $G \leq G'$ and $P(G) = 1$ implies that $e(G) \leq e(G')$. A (P, Φ)-*order* is a partial order that preserves P with size, and is preserved by Φ.

For example, let $\Phi = \{\oplus, \mathrm{nr}, \mathrm{nir}\}$ and let P be the property of not containing a path of length 4. For a rooted forest G, we denote by $h(G)$ the height of the

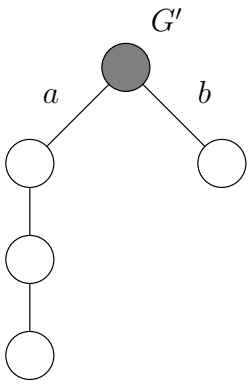

Figure 2: Example for the definition of complete collection. Let P be the property of not containing a path of length 4. The collection $\mathcal{C} = \{G' - \{a\}, G' - \{a, b\}\}$ is a complete collection of G' w.r.t. $\leq^3$.

forest G, namely the length of a longest path that starts in the root of G. We define a partial order $\leq^1$ by $G \leq^1 G'$ iff $h(G) \geq h(G')$. We have that $\leq^1$ is preserved by Φ: If $G_1 \leq^1 G_1'$ and $G_2 \leq^1 G_2'$ then

$$h(G_1 \oplus G_2) = \max(h(G_1), h(G_2)) \geq \max(h(G_1'), h(G_2')) = h(G_1' \oplus G_2'),$$

and therefore $G_1 \oplus G_2 \leq^1 G_1' \oplus G_2'$. Similarly, $G \leq^1 G'$ implies $\mathrm{nr}(G) \leq^1 \mathrm{nr}(G')$ (as $h(\mathrm{nr}(G)) = h(G) + 1 \geq h(G') + 1 = h(\mathrm{nr}(G'))$) and $\mathrm{nir}(G) \leq^1 \mathrm{nir}(G')$. The partial order $\leq^1$ does not preserve P, since $\hat{P}_2 \leq^1 \mathrm{nir}(\hat{P}_5)$, but $P(\hat{P}_2) = 1 > 0 = P(\mathrm{nir}(\hat{P}_5))$. The partial order $\leq^2$ defined by $G \leq^2 G'$ iff $h(G) \geq h(G')$ and $P(G) \leq P(G')$ preserves P, and is preserved by Φ. The partial order $\leq^3$ defined by $G \leq^3 G'$ iff $P(G) = 0$, or ($G \leq^2 G'$ and $e(G) \leq e(G')$) is a (P, Φ)-order.

For a collection $\mathcal{C}$ of rooted forests and a partial order $\leq$, a *complete collection of $\mathcal{C}$ (w.r.t. $\leq$)* is a collection $\mathcal{C}' \subseteq \mathcal{C}$ such that (1) there are no $G, G' \in \mathcal{C}'$ such that $G \leq G'$, and (2) for any $G \in \mathcal{C} - \mathcal{C}'$ there is some $G' \in \mathcal{C}'$ such that $G \leq G'$. A collection $\mathcal{C}$ of rooted subforests of a rooted forest G is called a complete collection of G (w.r.t. $\leq$) if $\mathcal{C}$ is a complete collection (w.r.t. $\leq$) of the set of all subforests of G. See Figure 2 for an example.

If $\leq$ is a (P, Φ)-order and we have a procedure to compute $\leq$, then the maximum subforest problem with respect to P can be solved by the algorithm MaxSubforest that is given in Figure 3.

Lemma 2.1. *Algorithm MaxSubforest solves the maximum subforest problem with respect to property P.*

Proof. To prove the correctness of this algorithm, we will show that for any tree G' corresponding to a node v in the composition tree, $\mathcal{C}(G')$ is a complete collection of G'. The proof is by induction on the structure of G'. The base of the induction is trivial. Suppose that $G' = f(G_1, \ldots, G_l)$, where $G_1, \ldots, G_l$ are the forests that

1 Arbitrarily choose a vertex r in G.
2 Compute a composition tree for G^r.
3 Scan the nodes of the composition tree in postorder.
For every node v **do**
4 Let G' be the forest that corresponds to v.
5 **If** v is a leaf **then** let $\mathcal{C}(G') = \{G'\}$.
else
6 Let f be the operator associated with v.
Let $G_1, \ldots, G_l$ be the trees that correspond to the children of v.
(i.e., $G' = f(G_1, \ldots, G_l)$)
7 Let $\mathcal{C}_0(G') = \{f'(H_1, \ldots, H_l) : f' \in \mathrm{sub}(f), H_1 \in \mathcal{C}(G_1), \ldots, H_l \in \mathcal{C}(G_l)\}$.
Let $\mathcal{C}(G') = \phi$.
8 **For** every $H \in \mathcal{C}_0(G')$ **do**
9 **If** there is least one forest $H' \in \mathcal{C}(G')$ such that $H' \leq H$ but $H \not\leq H'$ **then** remove all such forests from $\mathcal{C}(G')$ and add H to $\mathcal{C}(G')$.
else if there is no $H' \in \mathcal{C}(G')$ such that $H \leq H'$ **then** add H to $\mathcal{C}(G')$.
10 Check for each forest in $\mathcal{C}(G^r)$ if it has property P, and output a forest from $\mathcal{C}(G^r)$ that has property P and has maximum number of edges.

Figure 3: Algorithm MaxSubforest(G).

correspond to the children of v. By construction there are no $H, H' \in \mathcal{C}(G')$ such that $H \leq H'$. It remains to show that for any subforest H of G', there is a subforest $H' \in \mathcal{C}(G)$ such that $H \leq H'$. 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 the induction hypothesis, there are $H'_1 \in \mathcal{C}(G_1), \ldots, H'_l \in \mathcal{C}(G_l)$ such that $H_i \leq H'_i$ for $i = 1, \ldots, l$. Since $\leq$ is preserved by Φ, it follows that $H \leq f'(H'_1, \ldots, H'_l)$. Furthermore, it is easy to verify that $\mathcal{C}(G')$ is a complete collection of $\mathcal{C}_0(G')$. Thus, as $f'(H'_1, \ldots, H'_l) \in \mathcal{C}_0(G')$, there is a forest $H' \in \mathcal{C}(G)$ such that $f'(H'_1, \ldots, H'_l) \leq H'$. Therefore, $\mathcal{C}(G')$ is a complete collection of G'.

Let H^* be an optimal solution for the maximum subforest problem. From the fact that $\mathcal{C}(G^r)$ is a complete collection, it follows that there is a subforest $H \in \mathcal{C}(G^r)$ such that $H^* \leq H$, and from the fact that $\leq$ preserves P with size, it follows that H is an optimal solution. The algorithm returns a forest H' from $\mathcal{C}(G^r)$ that has property P and has maximum number of edges, and in particular, $e(H) \leq e(H')$. Therefore, H' is an optimal solution. ■

We now compute the time complexity of the algorithm. Let $|\leq|$ denote the size of the largest complete collection of $\mathcal{R}$ w.r.t. $\leq$ (we assume that $|\leq|$ is finite). Suppose that checking whether a forest G with at most n vertices has property P takes at most $T_1(n)$ time, and checking whether $G \leq G'$ for two forests with at most n vertices each takes at most $T_2(n)$ time. Suppose that v is some node in the decomposition tree, G' is the forest that corresponds to v, and $G_1, \ldots, G_l$ are

the forests that correspond to the children of v. The time complexity of building the collection $\mathcal{C}(G')$ is $O(|\mathcal{C}_0(G')| \cdot |\leq| \cdot T_2(n))$. We have that

$$|\mathcal{C}_0(G')| \leq \max_{f\in\Phi} |\mathrm{sub}(f)| \cdot \prod_{i=1}^{l} |\mathcal{C}(G_i)| \leq \max_{f\in\Phi} |\mathrm{sub}(f)| \cdot |\leq|^{a(\Phi)}.$$

Therefore, we have:

Theorem 2.2. *The time complexity of algorithm MaxSubforest is*

$$O(\max_{f\in\Phi} |\mathrm{sub}(f)| \cdot |\leq|^{a(\Phi)+1} \cdot T_2(n) \cdot n + |\leq| \cdot T_1(n)).$$

Naturally, given P and Φ, our goal will be to find a (P, Φ)-order $\leq$ such that $|\leq|$ is as small as possible.

We now give an example for the framework. For the rest of the section, let $\Phi_1 = \{\oplus, \mathrm{nr}, \mathrm{nir}\}$, and assume that P is some hereditary property. We define two partial orders on rooted forests: For two rooted forests G and G',

$$G \leq_P G' \iff \forall \text{rooted forest } J,\ P(G \oplus J) \leq P(G' \oplus J),$$

and

$$G \leq'_P G' \iff P(G) = 0 \text{ or } (G \leq_P G' \text{ and } e(G) \leq e(G')).$$

It is clear that $\leq_P$ is partial order. To show that $\leq'_P$ is a partial order, we need to show that $\leq'_P$ is transitive. Suppose that $G \leq'_P G' \leq'_P G''$. If $P(G) = 0$ then we are done. Otherwise, we have $1 = P(G) = P(G \oplus \hat{P}_1) \leq P(G' \oplus \hat{P}_1) = P(G')$, so we have that $G \leq_P G' \leq_P G''$ and $e(G) \leq e(G') \leq e(G'')$. Therefore, $G \leq_P G''$ and $e(G) \leq e(G'')$, and it follows that $G \leq'_P G'$.

Lemma 2.3. *For any rooted forest operator f (not necessarily in Φ_1), and for any $G_1, \ldots, G_{a(f)}, G'_1, \ldots, G'_{a(f)}$, if $G_i \leq_P G'_i$ for $i = 1, \ldots, a(f)$ then*

$$f(G_1, \ldots, G_{a(f)}) \leq_P f(G'_1, \ldots, G'_{a(f)}).$$

In particular, $\leq_P$ is preserved by Φ_1.

Proof. Let J be some rooted forest. Let J' be the subforest of $f(G'_1, G_2, \ldots, G_{a(f)}) \oplus J$ induced by the vertices of $G_2, \ldots, G_{a(f)}$, and J. The forest $f(G'_1, G_2, \ldots, G_{a(f)}) \oplus J$ is isomorphic to $G'_1 \oplus J'$. Therefore,

$$P(f(G_1, \ldots, G_{a(f)}) \oplus J) = P(G_1 \oplus J') \leq P(G'_1 \oplus J') = P(f(G'_1, G_2, \ldots, G_{a(f)}) \oplus J).$$

Repeating this argument gives that

$$\begin{aligned} P(f(G_1, \ldots, G_{a(f)}) \oplus J) &\leq P(f(G'_1, G_2, \ldots, G_{a(f)}) \oplus J) \\ &\leq P(f(G'_1, G'_2, G_3, \ldots, G_{a(f)}) \oplus J) \\ &\vdots \\ &\leq P(f(G'_1, \ldots, G'_{a(f)}) \oplus J). \end{aligned}$$

■

Lemma 2.4. *$\leq'_P$ is a (P, Φ_1)-order.*

Proof. To show that $\leq'_P$ is preserved by Φ_1, let G_1, G_2, G'_1, and G'_2 be rooted forests such that $G_1 \leq'_P G'_1$ and $G_2 \leq'_P G'_2$. If either $P(G_1) = 0$ or $P(G_2) = 0$ then we have $P(G_1 \oplus G_2) = 0$ as P is hereditary, and therefore $G_1 \oplus G_2 \leq'_P G'_1 \oplus G'_2$. Otherwise, $G_1 \leq_P G'_1$ and $G_2 \leq_P G'_2$, so Lemma 2.3 implies that $G_1 \oplus G_2 \leq_P G'_1 \oplus G'_2$. Furthermore, $e(G_1) \leq e(G'_1)$ and $e(G_2) \leq e(G'_2)$, so $e(G_1 \oplus G_2) = e(G_1) + e(G_2) \leq e(G'_1) + e(G'_2) = e(G'_1 \oplus G'_2)$. Therefore, $G_1 \oplus G_2 \leq'_P G'_1 \oplus G'_2$. Similarly, we have that if $G \leq'_P G'$ then $\mathrm{nr}(G) \leq'_P \mathrm{nr}(G')$ and $\mathrm{nir}(G) \leq'_P \mathrm{nir}(G')$.

To show that $\leq'_P$ preserves P with size, suppose that $G \leq'_P G'$. If $P(G) = 0$ then clearly $P(G) \leq P(G')$. Otherwise, as $G \leq_P G'$, we have that $P(G) = P(G \oplus \hat{P}_1) \leq P(G' \oplus \hat{P}_1) = P(G')$. Furthermore, if $P(G) = 1$ then by the definition of $\leq'_P$, $e(G) \leq e(G')$. ∎

We will use the partial order $\leq'_P$ in Section 4 and Section 5. We note that it is not hard to show that for any (P, Φ_1)-order $\leq$, $G \leq G'$ implies $G \leq'_P G'$ (Furthermore, for any partial order $\leq$ that preserves P and is preserved by Φ_1, $G \leq G'$ implies $G \leq_P G'$). Therefore $|\leq'_P| \leq |\leq|$, or in other words, $\leq'_P$ is an "optimal" (P, Φ_1)-order.

Let $=_P$ be the following equivalence relation: $G =_P G'$ iff $G \leq_P G'$ and $G' \leq_P G$. Let $|P|$ denotes the number of equivalence classes of $=_P$. Clearly, if $G =_P G'$ then either $G \leq'_P G'$ or $G' \leq'_P G$. Therefore, a complete collection cannot contain forests from the same equivalence class of $=_P$. It follows that $|\leq'_P| \leq |P|$. Since $a(\Phi_1) = 2$ and $\max_{f \in \Phi_1} |\mathrm{sub}(f)| = 2$, we conclude:

Theorem 2.5. *The time complexity of the algorithm in the above case is $O(|P|^3 \cdot T_2(n) \cdot n + |P| \cdot T_1(n))$.*

The difference between the approach we described in this section and the approach of Bern el al. [6], is that in the latter, an equivalence relation is used instead of a partial order, 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 $|\leq|$ for the appropriate partial order is small. This generalized setting is used in [27] in order to give polynomial algorithms to restrictions of MSP.

3 Counting families of matching subsets

In this section we describe and analyze a combinatorial problem. Its solution will be needed for the approximation scheme that we shall develop in the next section and for improving the time complexity of the exact algorithm. The combinatorial problem is interesting in its own right and may have other applications.

Let T be some finite set, and let $\mathcal{S}$ be a multiset whose elements are from T. We define the following mapping $R_{\mathcal{S}}$: Given a multiset of sets $\mathcal{A} = \{A_1, \ldots, A_p\}$

such that each A_i is a subset of T, $R_{\mathcal{S}}(\mathcal{A})$ returns the set of all multisets $\mathcal{B}$ such that $\mathcal{B} \subseteq \mathcal{S}$ and there is an injective mapping $f_{\mathcal{B}}\colon \mathcal{B} \to \mathcal{A}$ such that $b \in f_{\mathcal{B}}(b)$ for every $b \in \mathcal{B}$. In words, $R_{\mathcal{S}}(\mathcal{A})$ is the set of all sub-multisets of $\mathcal{S}$ that can be constructed by taking at most one element from each set A_i. Hence, each sub-multiset is obtained by matching to each of its elements a different set A_i containing that element. For this reason we call $R_{\mathcal{S}}(\mathcal{A})$ a *matching subsets family*. For example,

$$R_{\{1,2,3\}}(\{\{1,2\},\{1,3\}\}) = \{\phi,\{1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\}\}$$

and

$$R_{\{1,1,2,2,3\}}(\{\{1,2\},\{1,3\}\}) = \{\phi,\{1\},\{1,1\},\{2\},\{3\},\{1,2\},\{1,3\},\{2,3\}\}.$$

For integers n and k, $k \leq n$ we define $r(n,k)$ to be the maximum of $|\text{Image}(R_{\mathcal{S}})|$ taken over all $\mathcal{S}$ such that $|\mathcal{S}| = n$ and $|T| = k$. For example, $r(2,2) = 5$: Take $\mathcal{S} = T = \{1,2\}$ and the five matching subsets families in $\text{Image}(R_{\mathcal{S}})$ are $\{\phi\},\{\phi,\{1\}\},\{\phi,\{2\}\},\{\phi,\{1\},\{2\}\}$ and $\{\phi,\{1\},\{2\},\{1,2\}\}$ (these families are the images of $\phi,\{\{1\}\},\{\{2\}\},\{\{1,2\}\}$ and $\{\{1\},\{2\}\}$, respectively. The other types of sets $\mathcal{S}$ contain identical elements and so have smaller images). Our goal is to obtain upper and lower bounds for the maximum number of distinct matching subsets families as a function of n and k. In particular, we will show that $r(n,k) \leq \binom{n+2^k-1}{n}$.

Lemma 3.1. *For every multisets $\mathcal{S}$ and $\mathcal{A}$, there is a multiset $\mathcal{A}' \subseteq \mathcal{A}$, such that $|\mathcal{A}'| \leq |\mathcal{S}|$ and $R_{\mathcal{S}}(\mathcal{A}) = R_{\mathcal{S}}(\mathcal{A}')$.*

Proof. Build a bipartite graph $G = (U,V,E)$, where U is a set of $|\mathcal{S}|$ vertices labeled by the elements of $\mathcal{S}$, V is a set of $|\mathcal{A}|$ vertices labeled by the sets of $\mathcal{A}$, and for a vertex $u \in U$ whose label is i and a vertex $v \in V$ whose label is A, there is an edge $uv \in E$ iff $i \in A$. By a theorem of Dulmage and Mendelsohn (see, e.g., [19, p. 99]), there are a set $X \subseteq U$ with $|N(X)| \leq |X|$ (if $X \neq \phi$ then $|N(X)| < |X|$) and a matching M that matches all the vertices of $U - X$ to vertices in $V - N(X)$. We define $\mathcal{A}'$ to contain all the sets corresponding to vertices in $N(X)$, and all the sets corresponding to vertices of $V - N(X)$ which are matched in M. Clearly, $|\mathcal{A}'| \leq |\mathcal{S}|$ and since $\mathcal{A}' \subseteq \mathcal{A}$, it follows that $R_{\mathcal{S}}(\mathcal{A}') \subseteq R_{\mathcal{S}}(\mathcal{A})$.

To prove that $R_{\mathcal{S}}(\mathcal{A}') \supseteq R_{\mathcal{S}}(\mathcal{A})$, let $\mathcal{B}$ be any multiset in $R_{\mathcal{S}}(\mathcal{A})$. Let $f_{\mathcal{B}}\colon \mathcal{B} \to \mathcal{A}$ be the injection from the definition of R. We define $f'_{\mathcal{B}}\colon \mathcal{B} \to \mathcal{A}'$ in the following way: For every $b \in \mathcal{B}$, if the vertex u corresponding to b is in X, then let $f'_{\mathcal{B}}(b) = f_{\mathcal{B}}(b)$. Otherwise, let $f'_{\mathcal{B}}(b) = A$, where A is the set corresponding to the vertex that is matched to u in M. Since $f'_{\mathcal{B}}$ is an injection, we have that $\mathcal{B} \in R_{\mathcal{S}}(\mathcal{A}')$, and therefore, $R_{\mathcal{S}}(\mathcal{A}') \supseteq R_{\mathcal{S}}(\mathcal{A})$, as required. ∎

Corollary 3.2. $r(n,k) \leq \binom{n+2^k-1}{n}$.

Proof. The number of multisets of size at most a whose elements are from a set of size b is equal to the number of ways to partition a balls into $b+1$ cells, which is $\binom{a+b}{a}$. Therefore, as $|T| = k$, the number of multisets of size at most n whose elements are nonempty subsets of T is $\binom{n+2^k-1}{n}$, and the corollary follows from Lemma 3.1. ∎

We note that $\binom{n+2^k-1}{n} \leq \min(2^{O(kn)}, \left(\frac{en}{2^k-1} + e\right)^{2^k-1})$. We also give lower bounds on $r(n,k)$:

Lemma 3.3. $r(n,k) \geq \left(\lfloor \frac{n}{k2^{k-1}} \rfloor + 1\right)^{2^k-1}$. *Furthermore,*

$$r(n,k) \geq \begin{cases} 2^{\Omega(\frac{n}{\log_k n})} & \text{if } k = \log^{1+\Omega(1)} n \\ 2^{\Omega(k \log n)} & \text{if } k = \Omega(\frac{n}{\log n}) \end{cases}$$

Proof. Let $T = \{1, \ldots, k\}$ and let $\mathcal{S}$ be a some multiset containing each element of T at least $\lfloor n/k \rfloor$ times. We will give lower bound on $r(n,k)$ by giving a lower bound on the size of Image$(R_{\mathcal{S}})$. We call a mapping $g \colon (2^T - \{\phi\}) \to \mathbb{N}$ *bounded* if for all i, $\sum_{X: i \in X} g(X) \leq \lfloor n/k \rfloor$. For a bounded mapping g, we define $\mathcal{A}_g$ to be the multiset of sets obtained by taking each nonempty set $X \subseteq T$ exactly $g(X)$ times.

Claim 3.4. *If $g \neq h$ are two bounded mappings, then $R_{\mathcal{S}}(\mathcal{A}_g) \neq R_{\mathcal{S}}(\mathcal{A}_h)$.*

Proof. Let X be a set of minimal size for which $g(X) \neq h(X)$ and w.l.o.g. $g(X) > h(X)$. Build a multiset $\mathcal{B}$ in the following way: Begin with $\mathcal{B} = \phi$. For each $Y \subseteq X$, choose arbitrarily $i \in Y$, and add $g(Y)$ copies of i to $\mathcal{B}$. For each $Y \not\subseteq X$, choose arbitrarily $i \in Y - X$, and add $g(Y)$ copies of i to $\mathcal{B}$. Since g is bounded, we have that the multiplicity of each number i in $\mathcal{B}$ is at most $\lfloor n/k \rfloor$, and therefore $\mathcal{B} \subseteq \mathcal{S}$. Also, it is easy to see that $\mathcal{B} \in R_{\mathcal{S}}(\mathcal{A}_g)$. Let $\mathcal{B}'$ be the multiset obtained from $\mathcal{B}$ by removing all its elements which are in X. As $\mathcal{B}' \subseteq \mathcal{B}$ we have $\mathcal{B}' \in R_{\mathcal{S}}(\mathcal{A}_g)$.

If $\mathcal{B}' \notin R_{\mathcal{S}}(\mathcal{A}_h)$ then we are done. Otherwise, from the definition of $R_{\mathcal{S}}$ there is an injection $f \colon \mathcal{B}' \to \mathcal{A}_h$ such that $i \in f(i)$ for any $i \in \mathcal{B}'$. Clearly, $\sum_{Y: Y \not\subseteq X} g(Y) = |\mathcal{B}'|$ and $\sum_{Y: Y \not\subseteq X} h(Y) \geq |\mathcal{B}'|$ ($\sum_{Y: Y \not\subseteq X} h(Y) - |\mathcal{B}'|$ is the number of sets $Y \in \mathcal{A}_h$ which are not in the image of f and $Y \not\subseteq X$). If $\sum_{Y: Y \not\subseteq X} h(Y) > |\mathcal{B}'|$, select a set $Z \in \mathcal{A}_h$ which is not in the image of f and $Z \not\subseteq X$, and build a multiset $\mathcal{B}''$ by taking $\mathcal{B}'$ and adding a number $i \in Z - X$ chosen arbitrarily. Clearly, $\mathcal{B}'' \in R_{\mathcal{S}}(\mathcal{A}_h)$. However, the number of sets in $\mathcal{A}_g$ which contain at least one element from $\mathcal{B}''$ (i.e., an element from $Y - X$) is exactly $\sum_{Y: Y \not\subseteq X} g(Y) = |\mathcal{B}'|$, which is one less than the size of $\mathcal{B}''$. Hence $\mathcal{B}'' \notin R_{\mathcal{S}}(\mathcal{A}_g)$ so we are done again.

Now, suppose that $\sum_{Y: Y \not\subseteq X} h(Y) = |\mathcal{B}'|$, so $\sum_{Y: Y \not\subseteq X} g(Y) = \sum_{Y: Y \not\subseteq X} h(Y)$. From the minimality of X we have $g(Y) = h(Y)$ for all $Y \subsetneq X$. As $g(X) > h(X)$, we have $|\mathcal{B}| = \sum_Y g(Y) > \sum_Y h(Y)$ which implies that $\mathcal{B} \notin R_{\mathcal{S}}(\mathcal{A}_h)$. ∎

By the above claim, the number of distinct bounded mappings is a lower bound on the size of Image($R_{\mathcal{S}}$). We will therefore give a lower bound on the number of bounded mappings. If $g(X) \leq \lfloor \frac{n}{k2^{k-1}} \rfloor$ for all X, then g is bounded (as for all i, $\sum_{X:i\in X} g(X) \leq |X : i \in X| \lfloor \frac{n}{k2^{k-1}} \rfloor = 2^{k-1} \lfloor \frac{n}{k2^{k-1}} \rfloor \leq \lfloor \frac{n}{k} \rfloor$). Hence, the number of bounded mappings is at least $\left(\lfloor \frac{n}{k2^{k-1}} \rfloor + 1\right)^{2^k-1}$.

We now give better lower bounds for large values of k. Let $\mathcal{F}$ be a set of nonempty subsets of T such that

$$\text{for all } i, \text{ the number of sets in } \mathcal{F} \text{ that contain } i \text{ is at most } \lfloor n/k \rfloor \qquad (1)$$

Then any g that satisfies $g(X) \leq 1$ for $X \in \mathcal{F}$ and $g(X) = 0$ otherwise, is a bounded mapping, so the number of bounded mappings is at least $2^{|\mathcal{F}|}$. We now show how to build a set $\mathcal{F}$ that satisfies (1) with large size: Let l be the maximum integer such that $\binom{k}{l} \geq \lfloor \frac{n}{k} \rfloor$. If $k = \log^{1+\Omega(1)} n$ then $l = \Theta(\log_k \frac{n}{k})$. First, we assume that $\binom{k}{l} = \lfloor \frac{n}{k} \rfloor$. We take $\mathcal{F}$ to be all the subsets of T of size $l+1$ ($\mathcal{F}$ satisfies (1) as for each i, the number of sets in $\mathcal{F}$ that contain i is exactly $\binom{k}{l} = \lfloor \frac{n}{k} \rfloor$). Then

$$|\mathcal{F}| = \binom{k}{l+1} = \frac{k-l}{l+1}\binom{k}{l} = \Omega(\frac{n}{l}) = \Omega(\frac{n}{\log_k n})$$

which gives us the desired lower bound. If $\binom{k}{l} > \lfloor \frac{n}{k} \rfloor$ then let $p = 0.5\lfloor \frac{n}{k} \rfloor / \binom{k}{l}$ and build the set $\mathcal{F}$ by randomly adding each subset of T of size $l+1$ to $\mathcal{F}$ with probability p. Let X_i be the number of sets in $\mathcal{F}$ that contain i. Suppose that $k \leq n/100 \log n$. Using Chernoff bounds [14], we have that with high probability, $X_i \leq 2\mathrm{E}[X_i] = 2p\binom{k}{l} = \lfloor \frac{n}{k} \rfloor$ for all i, so $\mathcal{F}$ satisfies (1). Furthermore, with high probability,

$$|F| \geq \frac{1}{2}\mathrm{E}[|F|] = \frac{1}{2}p\binom{k}{l+1} = \frac{1}{2}p\frac{k-l}{l+1}\binom{k}{l} = \Omega(\frac{k}{l}) = \Omega(\frac{n}{\log_k n}).$$

Finally, we prove the third lower bound. Let $T_1, \ldots, T_l$ be a partition of T into disjoint subsets, and define $g(X) = 1$ if $X = T_i$ for some i, and $g(X) = 0$ otherwise. Since such a mapping g is bounded, we conclude that the number of bounded mappings is at least as the number of ways to partition T into disjoint subsets. This number is $2^{\Theta(k \log k)}$ (see, e.g., [21]). Therefore, for $k = n^{\Omega(1)}$ (and in particular, for $k = \Omega(n/\log n)$), the number of bounded mappings is $2^{\Omega(k \log n)}$. ■

Note that for fixed k our bounds are optimal up to constants: In this case, $r(n,k) = \Theta(n^{2^k-1})$. Another interesting case is when $k = n$. In this case we have $2^{\Omega(n \log n)} \leq r(n,n) \leq 2^{O(n^2)}$.

For Section 5 we will need a variation of the above problem. For a set T and an integer n, we define a mappings $R_{T,n}$ as follows: Given a multiset $\mathcal{A}$ of subsets of T, $R_{T,n}(\mathcal{A})$ is the set of all multisets of size at most n that can be

constructed by taking at most one element from each set in $\mathcal{A}$. In other words, $R_{T,n}(\mathcal{A}) = R_{\mathcal{S}}(\mathcal{A}) \cap U$, where $\mathcal{S}$ is a multiset containing each element of T exactly n times, and U is the set of all multisets of size at most n with elements from T. Therefore, we have $|\text{Image}(R_{T,n})| \leq r(nk, k)$, where $k = |T|$. As for a lower bound on $|\text{Image}(R_{T,n})|$, clearly, $\text{Image}(R_{T,n}) \supseteq \text{Image}(R_{\mathcal{S}})$ for any multiset $\mathcal{S}$ of size n with elements from T. Therefore, $|\text{Image}(R_{T,n})| \geq r(n, k)$.

4 An exact algorithm

We now turn to the problem of finding an exact solution to MSP.

Given an instance $(G, \mathcal{H})$ for MSP, we define a graph property P by $P(G) = 1$ iff $\mathcal{H} \not\subseteq G$. We then use algorithm MaxSubforest.

Let H be a tree and v be a vertex in H. We say that a tree J is *v-subtree* of H if J is a rooted subtree of H that is induced by v and one or more connected components of $H - \{v\}$, and whose root is v. J is called a *simple v-subtree* of H if it contains exactly one connected component of $H - \{v\}$. A tree J is called a *cropped v-subtree* if $\text{nr}(J)$ is a simple v-subtree. Let $u_1, \ldots, u_k$ denote all the vertices in all the trees in $\mathcal{H}$. We define for every $i \leq k$ the set F_i to be the set of all distinct (i.e., non-isomorphic) u_i-subtrees of H, where H is the tree from $\mathcal{H}$ that contains u_i. Also, let F_i' be the set of all distinct cropped u_i-subtrees of H, and let F_i'' be the multi-set of all cropped u_i-subtrees of H. Let $d_i = |F_i''|$ (i.e., d_i is the degree of u_i), $D_i = |F_i'|$, and $F = \cup_{i=1}^{k} F_i$. Note that given a set $\mathcal{H}$, the set F can be built in $2^{O(k)}$ time (This is done by building all the u_i-subtrees for every u_i, and then for every two resulting trees checking if they are isomorphic. The time complexity is $O(\sum_{i=1}^{k}(2^{d_i})^2 \cdot k) = 2^{O(k)}$.)

Now, we define a mapping h from rooted forests to 2^F as follows:

$$h(G) = \{H \in F : \mathcal{H} \subseteq G \oplus H\}.$$

Claim 4.1. *$G \leq_P G'$ iff $h(G) \supseteq h(G')$.*

Proof. Suppose that $h(G) \supseteq h(G')$ for two rooted forests G, G'. We need to show that for any rooted forest J, $P(G \oplus J) \leq P(G' \oplus J)$. It suffices to show that $P(G' \oplus J) = 0$ implies that $P(G \oplus J) = 0$, or in other words, $\mathcal{H} \subseteq G' \oplus J$ implies $\mathcal{H} \subseteq G \oplus J$. Let J be some rooted forest for which $\mathcal{H} \subseteq G' \oplus J$. There is a subgraph H of $G' \oplus J$ which is isomorphic to a tree from $\mathcal{H}$. Let J_H be the rooted subtree of J which is induced by the vertices of H. We have that $H \subseteq G' \oplus J_H$ and therefore $\mathcal{H} \subseteq G' \oplus J_H$. Furthermore, $J_H \in F$, thus $J_H \in h(G')$. It follows that $J_H \in h(G)$, so we have that $\mathcal{H} \subseteq G \oplus J_H$. Therefore $\mathcal{H} \subseteq G \oplus J$

The second direction is straightforward: If $G \leq_P G'$, then for every rooted forest J, $P(G' \oplus J) = 0$ implies that $P(G \oplus J) = 0$, namely $\mathcal{H} \subseteq G' \oplus J$ implies $\mathcal{H} \subseteq G \oplus J$. In particular, this is satisfied for every $J \in F$, and therefore $h(G) \supseteq h(G')$. ∎

Claim 4.1 gives an efficient procedure for checking whether $G \leq_P G'$ and therefore we can use algorithm MaxSubforest. In order to make the algorithm more efficient, we need to be able to compute $h(G')$ efficiently for every forest G' that is built during the execution of the algorithm. Each such forest is of the form $G_1 \oplus G_2$, $\mathrm{nr}(G_1)$ or $\mathrm{nir}(G_1)$. Suppose that $G' = G_1 \oplus G_2$. The sets $h(G_1)$ and $h(G_2)$ were computed previously by the algorithm. Clearly, $h(G') = \{H_1 \oplus H_2 \in F : H_1 \in h(G_1), H_2 \in h(G_2)\}$. Therefore, we can compute $h(G')$ by building all the forests of the form $H_1 \oplus H_2$ and then for each forest checking if there is a forest in F that is isomorphic to it. The time complexity of this procedure is $O(|\mathcal{C}(G_1)| \cdot |\mathcal{C}(G_2)| \cdot |F|k) = O(|F|^3 k) = 2^{O(k)}$. Similarly, the value of $h(\mathrm{nr}(G_1))$ or $h(\mathrm{nir}(G_1))$ can be computed from $h(G_1)$ in $2^{O(k)}$ time. Checking whether a forest with at most n vertices has property P takes $O((k^{1.5}/\log k)n)$ time [25]. Therefore, the time complexity of the MSP algorithm is $O(|P|^2 \cdot 2^{O(k)} \cdot n)$. Clearly $|P| \leq |2^F| = 2^{2^{O(k)}}$.

In the rest of this section, we give a better upper bound on $|P|$, using the results of Section 3. For $i = 1, \ldots, k$, define a mapping $h_i \colon \mathcal{R} \to 2^{F_i}$ as follows:

$$h_i(G) = \{H \in F_i : H \subseteq_R G\}.$$

Furthermore, define $\hat{h}(G) = (h_1(G), \ldots, h_k(G))$.

Lemma 4.2. *$\hat{h}(G) = \hat{h}(G')$ implies $G =_P G'$.*

Proof. Suppose that $\hat{h}(G) = \hat{h}(G')$ for two rooted forests G, G'. By symmetry, it suffices to show that $\mathcal{H} \subseteq G \oplus J$ for some rooted forest J implies $\mathcal{H} \subseteq G' \oplus J$. Let J be some rooted tree for which $\mathcal{H} \subseteq G \oplus J$, and let H be a subgraph of $G \oplus J$ which is isomorphic to a tree from $\mathcal{H}$. Let G_H be the rooted subtree of G which is induced by the vertices of H. We have that $G_H \subseteq_R G$ and $G_H \in F_i$ for some i, so $G_H \in h_i(G)$. As $\hat{h}(G) = \hat{h}(G')$, we have that $G_H \in h_i(G')$. Therefore $G_H \subseteq_R G'$ and it follows that $H \subseteq G' \oplus J$. ∎

By Lemma 4.2, we have that $|P| \leq |\mathrm{Image}(\hat{h})| \leq \prod_{i=1}^{k} |\mathrm{Image}(h_i)|$. We now give an upper bound on $|\mathrm{Image}(h_i)|$ for some fixed i. Define a mapping R' in the following way: Given a multiset of sets $\mathcal{A} = \{A_1, \ldots, A_k\}$, where each A_j is a subset of F_i', let

$$R'(\mathcal{A}) = \{\oplus_{H \in \mathcal{B}} \mathrm{nr}(H) : \mathcal{B} \in R_{F_i''}(\mathcal{A}), \mathcal{B} \neq \phi\} \cup \{\hat{P}_1\}$$

i.e., for every nonempty multiset $\mathcal{B}$ from $R_{F_i''}(\mathcal{A})$, $R'(\mathcal{A})$ contains the rooted tree formed by taking the trees in $\mathcal{B}$ and connecting their roots to a new vertex that becomes the new root. By definition, $|\mathrm{Image}(R')| = |\mathrm{Image}(R_{F_i''})| \leq r(d_i, D_i)$.

Let G be some rooted forest and let $J \in F_i$. We decompose G by $G = \mathrm{nr}(G_1) \oplus \cdots \oplus \mathrm{nr}(G_c)$, where c is the number of the children of the root of G. Also, we decompose J by $J = \mathrm{nr}(J_1) \oplus \cdots \oplus \mathrm{nr}(J_d)$, where each J_j is in F_i'. For each $j \leq c$, let $A_j = \{H \in F_i' : H \subseteq_R G_j\}$. Then, $J \subseteq_R G$ iff for each $j \leq d$,

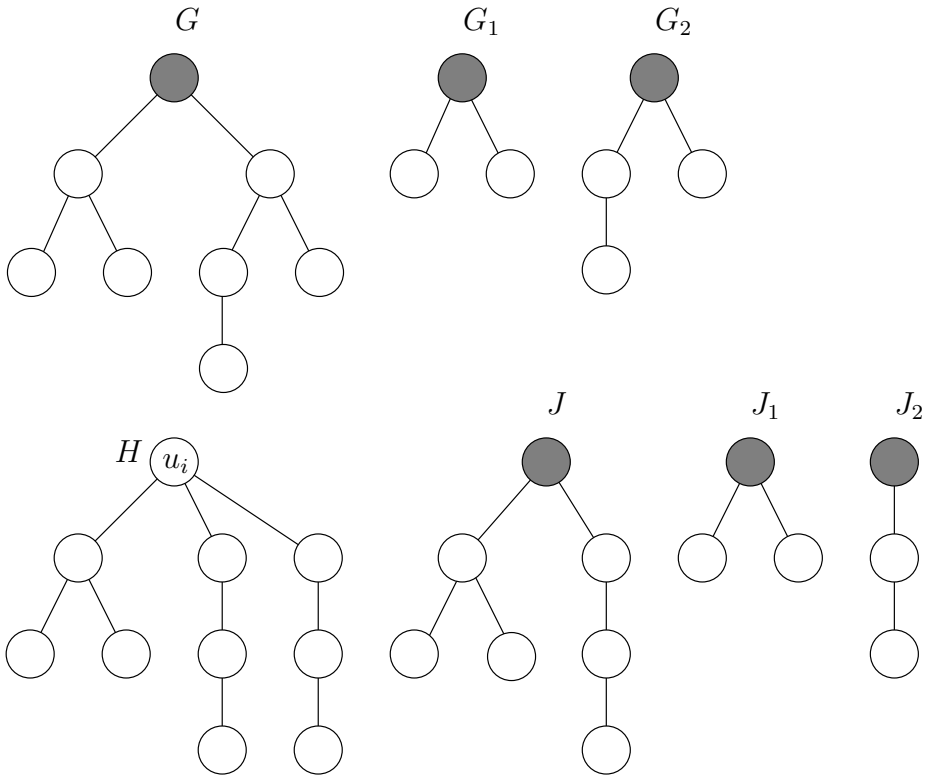

Figure 4: The connection between MSP and the matching subsets problem. Suppose that $\mathcal{H}$ consists of the tree H. We have $F_i' = \{J_1, J_2\}$ and $F_i'' = \{J_1, J_2, J_2\}$. Consider the the rooted forest G and the rooted tree J from F_i. We decompose G by $G = \mathrm{nr}(G_1) \oplus \mathrm{nr}(G_2)$, and decompose J by $J = \mathrm{nr}(J_1) \oplus \mathrm{nr}(J_2)$. We have $A_1 = \{J_1\}$ and $A_2 = \{J_1, J_2\}$, so $\{J_1, J_2\} \in R_{F_i''}(\{A_1, A_2\}) = \{\phi, \{J_1\}, \{J_2\}, \{J_1, J_2\}\}$. Therefore, $J \subseteq_R G$. More generally, we have that $h_i(G) = R'(\{A_1, A_2\}) = \{\hat{P}_1, \mathrm{nr}(J_1), \mathrm{nr}(J_2), J\}$.

there is a distinct j' such that $J_j \in A_{j'}$. This is a case of the matching subsets problem:

$$\begin{aligned} J \subseteq_R G &\iff \{J_1, \ldots, J_d\} \in R_{F_i''}(\{A_1, \ldots, A_c\}) \\ &\iff J \in R'(\{A_1, \ldots, A_c\}). \end{aligned}$$

(See Figure 4 for an example). Since every tree in $R'(\{A_1, \ldots, A_c\})$ is in F_i, we have that $h_i(G) = R'(\{A_1, \ldots, A_c\})$. Hence,

$$|\mathrm{Image}(h_i)| \leq |\mathrm{Image}(R')| \leq r(d_i, D_i).$$

Using Corollary 3.2 we have

$$|P| \leq \prod_{i=1}^{k} r(d_i, D_i) \leq \prod_{i=1}^{k} 2^{O(d_i D_i)} \leq 2^{O(\sum_{i=1}^{k} d_i D_i)} = 2^{O(k^2/\log k)},$$

where the last equality follows from an analysis similar to the one in [25]. We therefore have the following theorem:

Theorem 4.3. *The maximum subforest problem can be solved in* $2^{O(k^2/\log k)}n$ *time.*

We also note that the same approach can be used to solve the maximum subgraph problem for the case when G is a sparse graph: If G is a connected graph with n vertices and $n+p$ edges then we can find a maximum $\mathcal{H}$-free subgraph of G in $2^{O(k^5)}n + 2^{O(k^5 p)}$ time.

5 A polynomial time approximation scheme

In this section we give a polynomial time approximation scheme for MSP. The basic idea behind the algorithm is as follows: We partition the trees in $\mathcal{H}$ into two kinds: "small" trees and "big" trees. We first find an *optimal* subforest G' of G that does not contain a subforest that is isomorphic to a "small" tree. Due to the "smallness" of the forbidden trees, this stage will be done in polynomial time. In the second stage, we need to remove edges from G' such that at least one edge is removed from each subforest that is isomorphic to a "big" tree. Since each of these trees is "big", we can perform this task by deleting at most a constant fraction of the edges in G'.

A vertex x in a tree H is called an *l-separator* if every connected component in $H - \{x\}$ has at most l vertices. A tree H is called *l-separable* if it has an l-separator. We now describe a family $\{A_l\}_{l\geq 1}$ of approximation algorithms for MSP. Algorithm A_l is as follows:

1. Let $\mathcal{H}_l = \{H \in \mathcal{H} : H \text{ is } l\text{-separable}\}$. Find a maximum $\mathcal{H}_l$-free subforest of G, and call it G'.

2. Go over all connected components of G'. For each component T, select a root r and scan its vertices in postorder. When reaching a vertex $v \neq r$ such that the number of unmarked descendent vertices of v (including v) is at least $l+1$, tag the edge between v and its parent and also mark v and its descendents.

3. Output the untagged edges of G'.

Lemma 5.1. *Algorithm A_l returns a $\mathcal{H}$-free subforest of G with a number of edges that is at least $l/(l+1)$ times the number of edges in a maximum $\mathcal{H}$-free subforest of G.*

Proof. We denote by G_A the solution returned by the algorithm. We first show that G_A is $\mathcal{H}$-free. Suppose that $H \subseteq G_A$ for some $H \in \mathcal{H}$. Since H is connected,

we have that H is isomorphic to a subtree of some connected component S of G_A. Step 2 of the algorithm ensures that in G_A every connected component is l-separable. Since H is isomorphic to a subtree of S, we have that H is also l-separable, hence $H \in \mathcal{H}_l$. But as $H \subseteq G_A$, we conclude that $H \subseteq G'$, a contradiction to the fact that G' is $\mathcal{H}_l$-free. Therefore, $\mathcal{H} \not\subseteq G_A$.

We now prove the stated approximation factor. Note that for each edge of G' that was tagged in step 2, there are at least l edges which were not tagged (the edges in the subtree under the tagged edge). Hence, $e(G_A) \geq \frac{l}{l+1} e(G')$.

Let OPT denote the number of edges in a maximum $\mathcal{H}$-free subforest of G. Since $\mathcal{H}_l$ is a subset of $\mathcal{H}$, we have that the number of edges in a maximum $\mathcal{H}_l$-free subforest of G is at least OPT, namely, $e(G') \geq \text{OPT}$. Thus, $e(G_A) \geq \frac{l}{l+1}\text{OPT}$. ■

To show that the algorithms family $\{A_l\}_{l \geq 1}$ is a polynomial time approximation scheme, we need to show that for every fixed l, finding the maximum $\mathcal{H}_l$-free subforest of a tree G is polynomial in n and k. We now proceed to give an algorithm for this problem, based on the framework described in Section 2.

Define a property P by by $P(G) = 1$ iff $\mathcal{H}_l \not\subseteq G$. From the previous section, the time for finding the maximum $\mathcal{H}_l$-free subforest of a tree G is $O(|P|^2 \cdot |F|^3 \cdot kn)$. The bounds on $|P|$ and $|F|$ which were given in the previous section do not suffice here. We will give better bounds using the fact that the trees in $\mathcal{H}_l$ are l-separable.

We denote by d the maximum degree of a vertex in a tree in $\mathcal{H}$. Let B_0 be the set of all l-separable rooted trees with degree at most d, and let B be the set of all distinct (i.e., non isomorphic) rooted trees in B_0 in which the root is an l-separator. Since $F \subseteq B_0$, we have that $|F| \leq |B_0|$. Any tree in B_0 can be obtained by taking a tree in B, and choosing a new root. A tree in B has at most $dl + 1$ vertices, hence $|B_0| \leq (dl + 1)|B|$.

Let C be the set of all rooted trees with at most l vertices and degree at most d, and let $q = |C|$. It is known that $q = 2^{\Theta(l)}$ (see, e.g., [21, p. 1197]). Any tree in B can be obtained by choosing at most d trees from C, with repetitions, and connecting their roots to a new vertex, and conversely, any tree that is built using this process is in B. Therefore $|B| = \binom{d+q}{q} = O(d^q)$ (the last equality follows from the fact that l is constant, so q is also constant), so $|F| \leq (dl + 1)|B| = d^{2^{O(l)}}$.

Define mappings $h_1, h_2, \hat{h}$ as follows:

$$h_1(G) = \{H \in B : H \subseteq_R G\}$$
$$h_2(G) = \{H \in C : \mathcal{H}_l \subseteq G \oplus H\}$$
$$\hat{h}(G) = (h_1(G), h_2(G)).$$

For an example of these definitions, see Figure 5.

Lemma 5.2. *$\hat{h}(G) = \hat{h}(G')$ implies $G =_P G'$.*

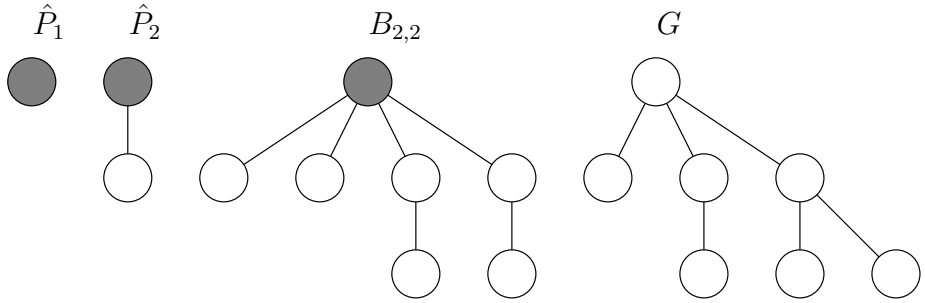

Figure 5: Example of the definition of h. Here $l = 2$, $d = 10$. In this case the set C consists of the two rooted trees $\hat{P}_1$ and $\hat{P}_2$. For any pair x, y of positive integers with $x + y \leq 10$ we match the rooted tree $B_{x,y}$ from B which is obtained by taking x copies of $\hat{P}_1$ 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. If $\mathcal{H}_l$ consists of an unrooted tree isomorphic to $B_{2,2}$, then for the graph G above, $h_1(G) = \{B_{x,y} : 0 \leq y \leq 2, 0 \leq x \leq 3 - y\}$ and $h_2(G) = \{\hat{P}_2\}$.

Proof. Suppose that $\hat{h}(G) = \hat{h}(G')$ for two rooted forest G, G'. We need to show that $\mathcal{H}_l \subseteq G \oplus J$ for some rooted forest J implies $\mathcal{H}_l \subseteq G' \oplus J$.

Let J be some rooted tree for which $\mathcal{H} \subseteq G \oplus J$, and let H be a subgraph of $G \oplus J$ which is isomorphic to a tree from $\mathcal{H}$. The tree H is l-separable, and let x be an l-separator of J. Let G_H and J_H be the rooted subforests of G and J, respectively, which are induced by the vertices of H. We consider two cases.

If x is the root of G_H (and J_H), then the root of G_H is an l-separator of G_H. If x is a vertex from J_H (except the root), then G_H is a subgraph of some connected component of $H - \{x\}$ and therefore the number of vertices in G_H is at most l. In particular, the root of G_H is an l-separator of G_H. In both cases it follows that $G_H \in B$ and since $G_H \subseteq_R G$ we have that $G_H \in h_1(G) = h_1(G')$. Therefore, $G_H \subseteq_R G'$ so $H = G_H \oplus J_H \subseteq G' \oplus J$. Hence, $\mathcal{H}_l \subseteq G' \oplus J$.

Now suppose that x in a vertex from G_H which is not the root. J_H is a subgraph of some connected component of $H - \{x\}$ and therefore the number of vertices in J_H is at most l. Thus, we have $J_H \in C$. As $H \subseteq G \oplus J_H$, we have $\mathcal{H}_l \subseteq G \oplus J_H$ implying that $J_H \in h_2(G) = h_2(G')$. Thus, $\mathcal{H}_l \subseteq G' \oplus J_H$ and we conclude that $\mathcal{H}_l \subseteq G' \oplus J$. ■

By Lemma 5.2 we have that

$$|P| \leq |\text{Image}(\hat{h})| \leq |\text{Image}(h_1)| \cdot |\text{Image}(h_2)|.$$

Clearly, $|\text{Image}(h_2)| \leq |2^C| = 2^q$. Define a mapping R' in the following way: Given a multiset of sets $\mathcal{A} = \{A_1, \ldots, A_k\}$, where each A_i is a subset of C, let

$$R'(\mathcal{A}) = \{\oplus_{H \in \mathcal{B}} \text{nr}(H) : \mathcal{B} \in R_{C,d}(\mathcal{A})\}.$$

Input: A tree G and a tree $H = K_{1,d}$.
Output: A maximum H-free subforest G' of G.

Select a vertex r of G to be the root of G.
For all vertices v in a postorder of G^r **do**
  If $v \neq r$ and the degree of v is at least d
    then remove the edge between v and its parent.
  While the degree of v is at least d **do**
    remove an edge between v and some child of v.
end for
return G.

Figure 6: Algorithm for the maximum subforest problem when the obstruction set consists of a single 1-separable tree.

Similarly to Section 4, we have that for any rooted tree G, $h_1(G) = R'(\mathcal{A}) \cap B$ for some $\mathcal{A}$. Hence,

$$|\text{Image}(h_1)| \leq |\text{Image}(R')| = |\text{Image}(R_{C,d})| \leq r(qd, d) \leq (qd)^{2^q - 1},$$

and therefore $|P| = d^{2^{2^{O(l)}}}$. We therefore proved the following theorem:

Theorem 5.3. *The algorithms family $\{A_l\}_{l \geq 1}$ is a polynomial time approximation scheme for the maximum subforest problem. In particular, algorithm A_l gives an $l/(l+1)$-approximation in $d^{2^{2^{O(l)}}} n$ time.*

The above theorem is of theoretical interest, but practically, the time complexity of the above approximation scheme is too large even for modest values of l. The bottleneck in the complexity is finding a maximum $\mathcal{H}$-free subforest of a tree G, where all trees in $\mathcal{H}$ are l-separable. However, we can give faster implementations of A_1 and A_2. To improve the running time of A_2 we need to describe how to find a maximum $\mathcal{H}$-free subforest of a tree G, for a set of 2-separable trees $\mathcal{H}$. Each 2-separable tree has diameter of at most 4, and by a result from [27], finding such subforest can be done in $O(|\mathcal{H}|n)$ time. Hence, A_2 can be implemented in $O(pn)$ time, where p is the number of 2-separable trees in $\mathcal{H}$ (note that $p \leq \binom{d+2}{2} = O(d^2)$).

The same algorithm can also find a $\mathcal{H}$-free subforest of a tree when all trees in $\mathcal{H}$ are 1-separable in $O(n)$ time, because in this case we can assume that $\mathcal{H}$ contains a single tree H (if it is not the case, we can remove from $\mathcal{H}$ all its trees but the smallest one). In fact, for this case we can give a much simpler algorithm. The algorithm is given in Figure 6. Hence, A_1 can be implemented in $O(n)$ time.

6 Acknowledgments

We thank Noga Alon for helpful discussions.

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability. *BIT*, 25:2–33, 1985.

[2] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. on Algebraic and Discrete Methods*, 8:277–284, 1987.

[3] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. *J. Assoc. Comput. Mach.*, 40:1134–1164, 1993.

[4] S. Arnborg, J. Lagergren, and D. Seese. Problems easy for tree-decomposable graphs. *J. of Algorithms*, 12:308–340, 1991.

[5] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems on graphs embedded in k-trees. *Discrete Applied Math*, 23:11–24, 1989.

[6] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear-time computation of optimal subgraphs of decomposable graphs. *J. of Algorithms*, 8(2):216–235, 1987.

[7] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Proc. 15th Int. Colloq. Automata, Languages and Programming*, LNCS 317, pages 105–118. Springer-Verlag, 1988.

[8] H. L. Bodlaender. Dynamic algorithms for graphs with treewidth 2. In *Proc. 19th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG 93)*, pages 112–124, 1993.

[9] H. L. Bodlaender. On reduction algorithms for graphs with small treewidth. In *Proc. 19th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG 93)*, pages 45–56, 1993.

[10] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Computing*, 25:1305–1317, 1996.

[11] H. L. Bodlaender and B. de Fluiter. Reduction algorithms for constructing solutions of graphs with small treewidth. In *Proc. 2nd Int. Conf. on Computing and Combinatorics (COCOON 96)*, LNCS 1090, pages 199–208. Springer-Verlag, 1996.

[12] H. L. Bodlaender and T. Hagerup. Parallel algorithms with optimal speedup for bounded treewidth. In *Proc. 22nd Int. Colloq. on Automata, Languages and Programming*, LNCS 944, pages 268–279. Springer-Verlag, 1995.

[13] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursively constructed graphs. *Algorithmica*, 7:555–581, 1992.

[14] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statis.*, 23:493–507, 1952.

[15] E. Cockayne, S. Goodman, and S. Hedetniemi. A linear algorithm for the domination number of a tree. *Information Processing Letters*, 4(2):41–44, 1975.

[16] B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.

[17] T. Kikuno, N. Yoshida, and Y. Kakuda. A linear algorithm for the domination number of series-parallel graphs. *Discrete Applied Math*, 37:299–311, 1983.

[18] R. Laskar, J. Pfaff, S. M. Hedetniemi, and S. T. Hedetniemi. On the algorithmic complexity of total domination. *SIAM J. on Algebraic and Discrete Methods*, 5(3):420–425, 1984.

[19] L. Lovasz and M. D. Plummer. *Matching Theory*. North-Holland, Amsterdam, 1986.

[20] S. Mahajan and J. G. Peters. Algorithms for regular properties in recursive graphs. In *Proc. 25th Allerton Conf. on Communications, Control and Computing*, pages 14–23, 1987.

[21] A. M. Odlyzko. Asymptotic enumeration methods. In R. L. Graham, M. Grotschel, and L. Lovasz, editors, *Handbook of Combinatorics*, volume 2, pages 1063–1229. Elsevier and the MIT press, 1995.

[22] P. Scheffler. Linear-time algorithms for NP-complete problems restricted to partial k-trees. Technical Report R-Math-03/87, Karl-Weierstrass-Inst. für Mathematik, Berlin, 1987.

[23] P. Scheffler and D. Seese. A combinatorial and logical approach to linear-time computability. In *Proc. European Conference on Computer Algebra*, LNCS 378, pages 379–380. Springer-Verlag, 1989.

[24] D. Seese. Tree-partite graphs and the complexity of algorithms. Technical Report P-MATH-08/86, Akademie der Wissenschaften der DDR, Karl-Weierstrass-Inst. für Mathematik, Berlin, 1986.

[25] R. Shamir and D. Tsur. Faster subtree isomorphism. *J. of Algorithms*, 33:267–280, 1999.

[26] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *J. Assoc. Comput. Mach.*, 29:623–641, 1982.

[27] D. Tsur. Tree deletion problems with bounded-diameter obstruction sets. Manuscript, 2004.