The maximum subforest problem: Approximation and exact algorithms
(extended abstract)

*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 so that $G - O$ satisfies P.

A graph property P is *hereditary* if for any 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. Therefore, the maximum subgraph problem w.r.t. an hereditary property P (whose obstruction set is finite) is a special case of the following problem which we call *the maximum subgraph problem:* Given a graph $G = (V, E)$ and an obstruction set $\mathcal{H}$, find a maximum edge set $E', E' \subseteq E$ such that (V, E') does not contain an induced subgraph isomorphic to any graph H from $\mathcal{H}$.

In this paper we concentrate on the case where all the input graphs are trees: Given a tree G and a set of trees $\mathcal{H}$, find a subforest of G that does not contain a subtree isomorphic to any tree from $\mathcal{H}$ and has a maximum number of edges. This problem is called the *maximum subforest problem* (MSP). Even this restricted version of the maximum subgraph problem is NP-hard [7]. While the edge deletion and maximum subforest problems are equivalent if an exact solution is required, approximating the maximization problem is easier: The maximum edge deletion problem on trees was recently shown to be hard to approximate within $c \log n$ factor for some constant c [7]. In contrast, we give here a polynomial time approximation scheme for the maximum subforest problem.

Bern, Lawler and Wong [2] described a general approach based on decomposition for constructing linear-time algorithms for subgraph problems on certain families of graphs, that include the graphs with bounded treewidth. The set of problems solvable in linear time includes all problems that can be described using monadic second-order logic (with some extension) as shown by Courcelle [4] and others [1, 3], and in particular it includes the edge deletion problem on trees for every fixed $\mathcal{H}$. Using the approach of [2], the maximum subgraph problem can be solved in $2^{p^{O(k)}}n$ time when the graph G has treewidth p. (Throughout, n denotes the number of vertices in G, and k is the total number of vertices in the graphs of $\mathcal{H}$). In particular, if G is a tree then the time complexity is $2^{2^{O(k)}}n$. In this paper, we give an $2^{O(k^2/\log k)}n$-time algorithm for MSP, improving the result implied by the work of Bern et al. Both our approximation and exact algorithms use a simple combinatorial lemma which is interesting by itself.

The paper is organized as follows: In §2 we give a general description of the algorithm for MSP. §3 describes the combinatorial lemma. In §4 we use the lemma and the exact algorithm to give a polynomial time approximation scheme for MSP. In §5 we refine the analysis of the exact algorithm for MSP and reduce its time complexity to $2^{O(k^2/\log k)}n$.

A *rooted forest* is a collection of trees with one distinguished vertex which is called the root. A *rooted tree* is a connected rooted forest. We will sometime write

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

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 denote the family of all rooted forests by $\mathcal{R}$.

We say that two rooted forests G^r and $H^{r'}$ are *isomorphic* if there is an isomorphism between G and H which maps r to r'. Likewise, we write $H^{r'} \subseteq_R G^r$ if there is a rooted subforest J^r of G^r which is isomorphic to $H^{r'}$ (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, $N(X) = \cup_{x \in X} N(x)$.

2 A framework for solving MSP

In this section we describe the general method for solving the maximum subforest problem resulting from the work of Bern, Lawler and Wong [2]. We will complete its details and study its time complexity in section 5. Also, in section 4 we will use this algorithm in order to give a polynomial time algorithm for a special case of MSP. The latter algorithm will be used in the approximation scheme for MSP.

We begin by describing three operators on rooted forests. 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. See figure 1 for examples.

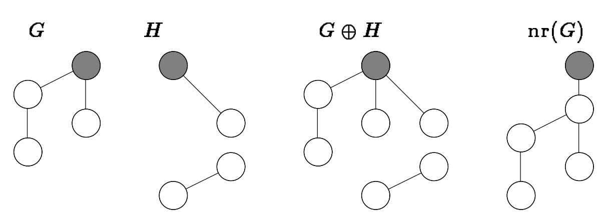

Figure 1: The $\oplus$ and nr operators

Let K_1 denote the rooted forest with one vertex. Every rooted forest can be built from copies of K_1 by a series of applications of the $\oplus$, nr and nir operators. A *composition tree* of a rooted forest G^r is a binary rooted tree such that its internal nodes of degree 1 are labeled by nr or nir, and its internal nodes of degree 2 are labeled by $\oplus$. Each node in the tree is associated with a rooted forest: A leaf is associated with the forest K_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^r.

Let P be a graph property and let D be some set. A mapping h from the family $\mathcal{R}$ of rooted forests onto D is called *homomorphism with respect to property* P if

1. For every rooted forests G and H, if $h(G) = h(H)$ then $P(G) = P(H)$.

2. There are operators $\oplus', \mathrm{nr}', \mathrm{nir}'$ defined on D such that $h(\mathrm{nr}(G)) = \mathrm{nr}'(h(G))$, $h(\mathrm{nir}(G)) = \mathrm{nir}'(h(G))$ and $h(G \oplus H) = h(G) \oplus' h(H)$ for any rooted forests G, H.

We call the operators $\oplus', \mathrm{nr}', \mathrm{nir}'$ the h*-operators*.

For example, consider the property $P =$ 'maximum degree < 3'. We build a homomorphism $h : \mathcal{R} \to \{0, 1, 2, 3\}$ with respect to P by

$$h(G^r) = \begin{cases} 3 & \text{if the max. degree of } G \geq 3 \\ \mathrm{degree}(r) & \text{otherwise} \end{cases}$$

and the h-operators are

$$\begin{aligned} c \oplus' d &= \min(3, c + d) \\ \mathrm{nr}'(d) &= \begin{cases} 3 & d \geq 2 \\ 1 & \text{otherwise} \end{cases} \\ \mathrm{nir}'(d) &= \begin{cases} 3 & d = 3 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In [2], a dynamic programming on the composition tree together with a utilization of the notion of homomorphism led to the following theorem:

THEOREM 2.1. *(Bern, Lawler and Wong [2]) Let $\mathcal{H}$ be a set of trees. Define property P by $P(G)$ iff $\mathcal{H} \not\subseteq G$. Given a homomorphism $h : \mathcal{R} \to D$ with respect to property P, and its h-operators, one can find a maximum $\mathcal{H}$-free subforest in any tree with n vertices in $O(|D|^2 n)$ time.*

In order to solve MSP, one still needs to describe how to construct an homomorphism h and its h-operators for a given obstruction set $\mathcal{H}$. We shall describe such constructions and analyze them in sections 4 and 5.

3 Counting families of matching subsets

We now turn to deal with a combinatorial problem that is of independent interest. 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.

Let T be some finite set, and let $\mathcal{S}$ be a multi-set whose elements are from T. We define the following mapping $R_{\mathcal{S}}$: Given a multi-set of sets $\mathcal{A} = \{A_1, A_2, \ldots, A_p\}$ such that each A_i is a subset of T, $R_{\mathcal{S}}(\mathcal{A})$ returns the set of all multi-sets $\mathcal{B}$ such that $\mathcal{B} \subseteq \mathcal{S}$ and there is an injective mapping $f_{\mathcal{B}} : \mathcal{B} \to \mathcal{A}$ 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 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 type of sets $\mathcal{S}$ contain identical elements and so have smaller images). Our goal is to evaluate (or bound) the maximum number of distinct matching subsets families as a function of n and k. We will show that $r(n,k) \leq \binom{n+2^k-1}{n}$.

Lemma 3.1. *For every multi-sets $\mathcal{S}$ and $\mathcal{A}$, there is a multi-set $\mathcal{A}' \subseteq \mathcal{A}$, 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., [5, p. 99]), there are a set $X \subseteq U$ with $|N(X)| < |X|$ and matching M that matches the vertices of $U - X$ into $V - N(X)$. We define $\mathcal{A}'$ to contain all the sets corresponding to vertices in $N(X)$ to $\mathcal{A}'$, and all the sets corresponding to vertices of $V - N(X)$ which are matched in M.

To prove that $R_{\mathcal{S}}(\mathcal{A}') \supseteq R_{\mathcal{S}}(\mathcal{A})$, let $\mathcal{B}$ be any element of $R_{\mathcal{S}}(\mathcal{A})$. Let $f_{\mathcal{B}} : \mathcal{B} \to \mathcal{A}$ be the injection from the definition of R. We define $f'_{\mathcal{B}} : \mathcal{B} \to \mathcal{A}'$ 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 3.1. $r(n,k) \leq \binom{n+2^k-1}{n}$.

Proof. If $\mathcal{S}$ is a multiset of size n and $|T| = k$, then from lemma 3.1 the image of $R_{\mathcal{S}}$ on all multi-sets of sets is equal to the image of $R_{\mathcal{S}}$ on all the multi-sets of at most n sets, where each set is a nonempty subset of a set of size k. The number of such distinct multi-sets is $\binom{n+2^k-1}{n}$. ■

We also give lower bounds on the value of $r(n,k)$:

Lemma 3.2. $r(n,k) \geq \left(\lfloor \frac{n}{k\,2^{k-1}} \rfloor + 1\right)^{2^k-1}$. *Furthermore,*

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

We omit the proof in this abstract. Note that for fixed k our bounds are optimal up to constants (in this case $r(n,k) = \Theta(n^{2^k-1})$).

4 A polynomial time approximation scheme

In this section we give a polynomial time approximation scheme for MSP.

A vertex x in a tree H is called a *l-separator* if every connected component in $H - x$ has at most l vertices. A tree H is called *l-separable* if it has a 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 $l+1$, mark the edge between v and its parent and also mark v and its descendents.

3. Output the unmarked edges of G'.

LEMMA 4.1. *Algorithm A_l returns an $\mathcal{H}$-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 removed in step 2, there are at least l edges which were not removed (the edges in the subtree under the removed 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, $e(G') \geq \mathrm{OPT}$. Thus, $e(G_A) \geq \frac{l}{l+1}\mathrm{OPT}$. ∎

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.

Denote by d the maximum degree of a vertex in a tree in $\mathcal{H}$. Let B be the set of all rooted trees H^r such that the degree of r is at most d, and r is a l-separator. Let C be the set of all rooted trees with at most l vertices. Recall that for a set A, 2^A denotes the set of all subsets of A. Let $D = 2^B \times 2^C$, and denote $\beta = (B, C)$. We now define a mapping h from rooted forests to D:

$$h(G^r) = \begin{cases} \beta & \text{if } \mathcal{H}_l \subseteq G \\ (\{H \in B : H \subseteq_R G^r\}, & \\ \{H \in C : \mathcal{H}_l \subseteq G \oplus H\}) & \text{otherwise} \end{cases}$$

We write $h(G^r)_1, h(G^r)_2$ for the first and second coordinates of $h(G^r)$. For an example of the definition of h, see figure 2. We note that the somewhat complicated definitions of D and h seem essential. It is possible to define a simpler homomorphism for the property $\mathcal{H}_l \not\subseteq G$ (namely, we can define $h(G^r) = \{H \in B' : H \subseteq_R G^r\}$ where B' is the set of all rooted l-separable trees with maximum degree at most d) but then algorithm A_l would not be polynomial.

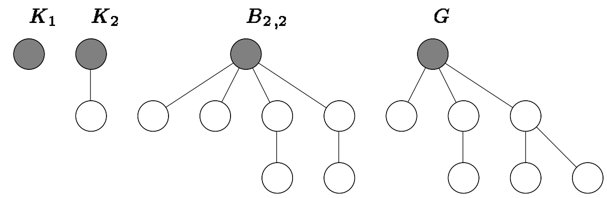

Figure 2: Example of the definition of h. Here $l = 2$, $d = 10$. In this case the set C consists of the two rooted trees K_1 and K_2. For any pair x, y of positive integers where $x + y \leq 10$ we match the rooted tree $B_{x,y}$ from B which is obtained by taking x copies of K_1 and y copies of K_2, adding a new vertex which is connected 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 $h(G) = (\{B_{x,y} : 0 \leq y \leq 2, 0 \leq x \leq 3 - y\}, \{K_2\})$.

LEMMA 4.2. *h is a homomorphism with respect to the property $\mathcal{H}_l \not\subseteq G$.*

Proof. It is clear that the first condition of homomorphism is true. To show the second condition, we define a mapping $\alpha : B \times C \to D$ by

$$\alpha(d_1, d_2) = \begin{cases} \beta & \text{if } K_1 \in d_2 \\ (d_1, d_2) & \text{otherwise} \end{cases}$$

We now define the h-operators $\oplus', \mathrm{nr}', \mathrm{nir}'$ in the following way: $\mathrm{nr}'(\beta) = \mathrm{nir}'(\beta) = \beta$, and for every $d \in D$, $d \oplus' \beta = \beta \oplus' d = \beta$. For $c, d \neq \beta$ define:

$$\begin{aligned} c \oplus'_1 d &= \{H_1 \oplus H_2 : H_1 \in c_1, H_2 \in d_1\} \cap B \\ c \oplus'_2 d &= \{H \in C : \exists H_1 \in c_1, H_2 \in d_1, H' \subseteq_R H, \text{ s.t.} \\ &\quad (H' \oplus H_2 \in c_2) \vee (H' \oplus H_1 \in d_2) \vee \\ &\quad (H_1 \oplus H_2 \oplus H' \in \mathcal{H}_l)\} \\ \mathrm{nr}'_1(d) &= \{\mathrm{nr}(H) : H \in d_1\} \cap B \\ \mathrm{nr}'_2(d) &= \{H \in C : (\exists H' \subseteq_R H \text{ s.t. } \mathrm{nr}(H') \in d_2) \vee \\ &\quad (\exists H' \in d_1 \text{ s.t. } \mathrm{nr}(H') \oplus H \in \mathcal{H}_l)\} \end{aligned}$$

and then

$$\begin{aligned} c \oplus' d &= \alpha(c \oplus'_1 d, c \oplus'_2 d) \\ \mathrm{nr}'(d) &= \alpha(\mathrm{nr}'_1(d), \mathrm{nr}'_2(d)) \\ \mathrm{nir}'(d) &= (\{K_1\}, \{H \in C : \mathcal{H}_l \subseteq H\}) \end{aligned}$$

We need to show that $\oplus', \mathrm{nr}', \mathrm{nir}'$ are indeed h-operators. We will only show that $\mathrm{nr}'(h(G)) = h(\mathrm{nr}(G))$, as the proof for the other cases is similar.

CLAIM 4.1. *If $\mathcal{H}_l \not\subseteq nr(G)$ then $h(nr(G))_1 = nr'_1(h(G))$*

Proof. $\mathcal{H}_l \not\subseteq \mathrm{nr}(G)$ implies that $\mathcal{H}_l \not\subseteq G$. By definitions, for any $H \in B$ we have

$$\begin{aligned} H \in h(\mathrm{nr}(G))_1 &\iff H \subseteq_R \mathrm{nr}(G) \\ &\iff \exists H' \in B, H = \mathrm{nr}(H'), H' \subseteq_R G \\ &\iff \exists H' \in h(G)_1, H = \mathrm{nr}(H') \\ &\iff H \in \mathrm{nr}'_1(h(G)). \end{aligned}$$

■

CLAIM 4.2. *If $\mathcal{H}_l \not\subseteq G$ then $h(nr(G))_2 = nr'_2(h(G))$*

Proof. Suppose that $H \in \mathrm{nr}'_2(h(G))$ for some $H \in C$, namely either there is $H' \subseteq_R H$ such that $\mathrm{nr}(H') \in h(G)_2$ or there is $H' \in h(G)_1$ such that $\mathrm{nr}(H') \oplus H \in \mathcal{H}_l$. In the first case we have that $\mathcal{H}_l \subseteq G \oplus \mathrm{nr}(H') \subseteq G \oplus \mathrm{nr}(H) = \mathrm{nr}(G) \oplus H$ and therefore $\mathcal{H}_l \subseteq \mathrm{nr}(G) \oplus H$ implying $H \in h(\mathrm{nr}(G))_2$. In the second case, as $H' \subseteq_R G$, we have that $\mathcal{H}_l \subseteq \mathrm{nr}(H') \oplus H \subseteq \mathrm{nr}(G) \oplus H$ and again $H \in h(\mathrm{nr}(G))_2$.

To prove the opposite direction, assume that $H \in h(\mathrm{nr}(G))_2$. Then there is a subtree J of $\mathrm{nr}(G) \oplus H$ which is isomorphic to some tree in $\mathcal{H}_l$ (if $\mathcal{H}_l \subseteq \mathrm{nr}(G)$ then we can assume that J is a subtree of $\mathrm{nr}(G)$). Let r, s be the roots of G and H respectively. Let G' be the rooted subtree of G induced by the vertices of J, and whose root is r, and similarly H' is the rooted subtree of H induced by the vertices of J, and whose root is s. Clearly, $J = \mathrm{nr}(G') \oplus H' = G' \oplus \mathrm{nr}(H')$. If H' has at most $l-1$ vertices then $\mathrm{nr}(H') \in C$. Since $\mathcal{H}_l \subseteq J = G' \oplus \mathrm{nr}(H') \subseteq G \oplus \mathrm{nr}(H')$, then $\mathrm{nr}(H') \in h(G)_2$. Thus $H \in \mathrm{nr}'_2(h(G))$. We now consider the case when H' has at least l vertices. As $H \in C$, H can have at most l vertices, and therefore $H' = H$. We know that $J = \mathrm{nr}(G') \oplus H \in \mathcal{H}_l$ and that $G' \subseteq_R G$. To finish the proof we need to show that G' is in B, because then it follows that $G' \in h(G)_1$ and therefore $H \in \mathrm{nr}'_2(h(G))$.

First, the degree of r in J is at most d as $J \in \mathcal{H}_l$. Thus, the degree of r in G' is at most $d-1$. Denote by S the vertices of J which are vertices of H, plus the vertex r. Since $J \in \mathcal{H}_l$, then J is l-separable, and let x be some l-separator of J. Since S induces a connected subgraph of J of size $l+1$, it follows that $x \in S$. Therefore, any connected component of $G' - r$ is contained in some connected component of $J - x$ and therefore its size is at most l. This implies that r is a l-separator of G', so $G' \in B$. ■

We now look at three cases:

1. $\mathcal{H}_l \subseteq G$. Immediately, we have $\mathrm{nr}'(h(G)) = \mathrm{nr}'(\beta) = \beta$. Since $\mathcal{H}_l \subseteq G$ implies that $\mathcal{H}_l \subseteq \mathrm{nr}(G)$, we also have $h(\mathrm{nr}(G)) = \beta$.
2. $\mathcal{H}_l \not\subseteq G$, $\mathcal{H}_l \subseteq \mathrm{nr}(G)$. By claim 4.2 we have $\mathrm{nr}'_2(h(G)) = h(\mathrm{nr}(G))_2$. Since $h(\mathrm{nr}(G)) = \beta$, then $\mathrm{nr}'_2(h(G)) = h(\mathrm{nr}(G))_2 = C$, and in particular $K_1 \in \mathrm{nr}'_2(h(G))$. Therefore, by the definition of α we have $\mathrm{nr}'(h(G)) = \beta$.
3. $\mathcal{H}_l \not\subseteq G$, $\mathcal{H}_l \not\subseteq \mathrm{nr}(G)$. By claims 4.1 and 4.2 we have $(\mathrm{nr}'_1(h(G)), \mathrm{nr}'_2(h(G))) = h(\mathrm{nr}(G))$. As $\mathcal{H}_l \not\subseteq \mathrm{nr}(G)$, we get that $K_1 \notin h(\mathrm{nr}(G))_2 = \mathrm{nr}'_2(h(G))$. Hence, $\alpha(\mathrm{nr}'_1(h(G)), \mathrm{nr}'_2(h(G))) = (\mathrm{nr}'_1(h(G)), \mathrm{nr}'_2(h(G)))$ and therefore, $\mathrm{nr}'(h(G)) = h(\mathrm{nr}(G))$.

■

Denote $q = |C|$, namely, q is the number of rooted trees with at most l vertices. Clearly, $|B| = \binom{d+q}{q} < (d+q)^q$. It is known that $q = 2^{\Theta(l)}$ (see, e.g., [6, p. 1197]), and therefore, $|D| < 2^{(d+q)^q+q} = 2^{d^{2^{O(l)}}}$ (in fact, $|D| = 2^{d^{2^{\Theta(l)}}}$). This implies that if $d = 2^{o(\log\log(k+n))}$, then algorithm A_l is polynomial for every l. In order to remove the degree requirement, we will show (using Corollary 3.1) that the cardinality of the image of h is much smaller than $2^{d^{2^{O(l)}}}$.

For a set T of pairwise non-isomorphic trees, and a multi-set $\mathcal{S}$ of elements from T, we define a mapping $R'_{\mathcal{S}}$ in the following way: Given a multi-set of sets $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ where each A_i is a subset of T, let

$$R'_{\mathcal{S}}(\mathcal{A}) = \{\oplus_{H \in \mathcal{B}} H : \mathcal{B} \in R_{\mathcal{S}}(\mathcal{A})\}$$

i.e., for every multi-set $\mathcal{B}$ from $R_{\mathcal{S}}(\mathcal{A})$, $R'_{\mathcal{S}}(\mathcal{A})$ contains the rooted tree formed by joining the roots of all the trees in $\mathcal{B}$. Clearly, $|\mathrm{Image}(R'_{\mathcal{S}})| \le |\mathrm{Image}(R_{\mathcal{S}})|$.

Let G^r be any rooted tree and let $J \in B$. We decompose G by $G = \mathrm{nr}(G_1) \oplus \cdots \oplus \mathrm{nr}(G_c)$. Also, we decompose J by $J = \mathrm{nr}(J_1) \oplus \cdots \oplus \mathrm{nr}(J_{d'})$ where each J_j is in C and $d' \le d$. For each $j \le c$, let $A_j = \{H \in C : H \subseteq_R G_j\}$. Then, $J \subseteq_R G$ iff for each $j \le d'$, there is a distinct j' such that $J_j \in A_{j'}$. This is a case of the matching subsets problem: Define $\mathcal{S}$ to be a multi-set over C, containing each element of C exactly d times. Then

$$J \subseteq_R G \iff \{J_1, \ldots, J_{d'}\} \in R_{\mathcal{S}}(\{A_1, \ldots, A_c\})$$

or,

$$J \subseteq_R G \iff J \in R'_{\mathcal{S}}(\{A_1, \ldots, A_c\}).$$

Therefore, for every G, the first coordinate of h(G) (the set $\{J \in B : J \subseteq_R G\}$) must be in the image of $R'_{\mathcal{S}}$. Hence, by Corollary 3.1,

$$|\mathrm{Image}(h)| \le |\mathrm{Image}(R'_{\mathcal{S}})| \cdot |2^C| \le r(qd, q)2^q = d^{2^{2^{O(l)}}}$$

Now, in order to implement algorithm A_l in $d^{2^{2^{O(l)}}} n$ time, we need to show how to build the set

Image($R'_{\mathcal{S}}$) and the h-operators in $d^{2^{2^{O(l)}}}$ time. To build Image($R'_{\mathcal{S}}$), we first build C and $\mathcal{S}$, and then we build the image of $R_{\mathcal{S}}$ by first computing the value of $R_{\mathcal{S}}$ on every multi-set with $|\mathcal{S}|$ elements (each element is a subset of C). This creates $d^{2^{2^{O(l)}}}$ sets where some of them are identical. By comparing every two sets we created we can remove multiple copies of the same set and obtain Image($R_{\mathcal{S}}$). Afterwards, we can easily build Image($R'_{\mathcal{S}}$). Constructing the h-operators is straightforward. We therefore proved the following theorem:

THEOREM 4.1. *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.*

(More precisely, the running time of A_l is $O(d^{2(2^{q_l}-1)}n)$ where q_l is the number of distinct rooted trees with at most l vertices).

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. We note that using techniques developed in [7], we can solve this problem faster for l=1,2: Algorithm A_1 can be implemented in $O(n)$ time, and algorithm A_2 (that gives a guaranteed approximation ratio of at least 2/3) in $O(d^2 n)$.

5 An exact algorithm

We now turn to the problem of finding an exact solution to MSP. 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 whose root is v and is induced by v and one or more connected components of $H-v$. J is called a *simple v-subtree* of H if it contains exactly one connected component of $H-v$. Let $\mathcal{H}$ be a set of trees. Denote by $u_1,\ldots,u_k$ all the vertices in all the trees in $\mathcal{H}$. We define for every $i\leq k$ the set $\mathcal{F}_i$ to be the set of all u_i-subtrees of H, where H is the tree from $\mathcal{H}$ that contains u_i. Also, let $\mathcal{F}'_i$ be the set of all simple u_i-subtrees of H. We define the set $\mathcal{F}=\mathcal{F}_1\cup\cdots\cup\mathcal{F}_k$.

We define a mapping h from rooted forests to $D=2^{\mathcal{F}}$ in the following way (we denote $\beta=\mathcal{F}$):

$$h(G^r)=\begin{cases}\beta & \text{if } \mathcal{H}\subseteq G\\ \{H\in\mathcal{F}: H\subseteq_R G^r\} & \text{otherwise}\end{cases}$$

LEMMA 5.1. *h is a homomorphism with respect to the property $\mathcal{H}\not\subseteq G$.*

The proof of Lemma 5.1 is similar to the proof of Lemma 4.2, and thus is omitted. We have that $|D|=2^{2^{\Theta(k)}}$, but like before we can bound the size of Image(h). Denote by d_i the degree of u_i (i.e., $d_i=|\mathcal{F}'_i|$) and by D_i the number of distinct (i.e., non-isomorphic) trees in $\mathcal{F}'_i$. Using Corollary 3.1 we have that the size of Image(h) is at most

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

where the last equality follows from an analysis similar to the one in [8]. We therefore have

THEOREM 5.1. *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 edge deletion 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 $2^{O(k^5)}n+2^{O(k^5p)}$ time.

6 Acknowledgments

We thank Noga Alon for helpful discussions.

References

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

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

[3] 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.

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

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

[6] A. M. Odlyzko. Asymptotic enumeration methods. In *Handbook of Combinatorics, edited by R. L. Graham, M. Grotschel, and L. Lovasz, Elsevier and The MIT Press, 1995*, volume 2. 1995.

[7] R. Shamir and D. Tsur. The tractability of the tree deletion problem. Manuscript.

[8] R. Shamir and D. Tsur. Faster subtree isomorphism. In *Proc. of the fifth Israel Symposium on the Theory of Computing and Systems, (ISTCS 97)*, pages 126–131, 1997.