Graph decomposition of slim graphs

Yair Caro [*] Raphael Yuster [†]

Abstract

A Graph $G = (V, E)$ is called *k-slim* if for every subgraph $S = (V_S, E_S)$ of G with $s = |V_S| \geq k$ there exists $K \subset V_S$, $|K| = k$, such that the vertices of $V_S \setminus K$ can be partitioned into two subsets, A and B, such that $|A| \leq \frac{2}{3}s$ and $|B| \leq \frac{2}{3}s$ and no edge of E_S connects a vertex from A and a vertex from B. k-slim graphs contain, in particular, the graphs with *tree-width* k. In this paper we give an algorithm solving the H-decomposition problem for a large family of graphs H which contains, among others, the stars, the complete graphs, and the complete r-partite graphs where $r \geq 3$. The algorithm runs in polynomial time in case the input graph is k-slim, where k is fixed. In particular, our algorithm runs in polynomial time when the input graph has bounded tree width k. Our results supply the first polynomial time algorithm for H-decomposition of connected graphs H having at least 3 edges, in graphs with bounded tree-width.

1 Introduction

All graphs considered here are finite, undirected and simple. For the standard graph-theoretic notations the reader is referred to [5]. Let $G = (V, E)$ be a graph, and let $0.5 < \alpha < 1$. A partition of V into three subsets K, A, B is called a (k, α)-*separator* if $|K| = k$ and no edge of E connects a vertex from A and a vertex from B. Furthermore, $|A| \leq \alpha|V|$ and $|B| \leq \alpha|V|$. Small separators of graphs have been studied extensively, see e.g. [11, 2].

We define the class of (k, α)-*slim* graphs, which are graphs G that have the property that *every* subgraph of G with more than k vertices has a (k, α)-separator. It is easy to see that every (k, α)-*slim* graph is also a $(dk, \frac{2}{3})$-slim graph where $d = \max\{1, \frac{\log 2/3}{\log \alpha}\}$. Since in our applications k and α will be fixed, we only consider $(k, \frac{2}{3})$-slim graphs, and call them k-slim. By a *k-separator* we always mean a $(k, 2/3)$-separator. Note also that a k-slim graph does not have a $3k$-connected subgraph, since such a subgraph must contain at least $3k + 1$ vertices, and hence cannot have a k-separator.

[*]Department of Mathematics, University of Haifa-ORANIM, Tivon 36006, Israel. e-mail: zeac603@uvm.haifa.ac.il

[†]Department of Mathematics, University of Haifa-ORANIM, Tivon 36006, Israel. e-mail: raphy@math.tau.ac.il

The motivation behind the notion of slim graphs is the following. Given a k-slim graph $G = (V, E)$ with $|V| \geq k$, one can always find a k-separator K, A, B. The induced subgraph on $A \cup K$ either contains less than, say, $6k$ vertices, or otherwise can be further separated into two appropriate smaller parts. The same holds for $B \cup K$. Thus the graph G can be recursively separated in a balanced manner until one obtains parts with, say, at most $6k$ vertices. This recursive k-separation makes slim graphs suitable for dynamic programming techniques, assuming the solution to the problem at hand for the input graph can be efficiently derived from solutions to the problem on the smaller parts.

There are many well-known non-trivial families of graphs which are k-slim, for some fixed values of k. For example, trees are 1-slim. This follows from the fact that in every tree T on n vertices there is a vertex v such that every connected component of $T \setminus \{v\}$ has size at most $2n/3$. Generalizing the concept of a tree, Robertson and Seymour, in their series of works about graph minors, introduced the concept of the *tree-width* of a graph [13]. The graphs with tree-width 1 are the forests and the graphs with tree-width 2 contain the series-parallel graphs. Graphs with tree-width k are also called *partial k-trees* as they can be viewed as subgraphs of chordal graphs with clique size $k + 1$. It is shown in [14] that graphs with tree-width k have a $(k, \frac{2}{3})$-separator. Hence, since every subgraph of a graph with tree-width k also has tree-width k, it follows that graphs with tree-width k are k-slim. Thus, for fixed k, any polynomial algorithm that applies to k-slim graphs also applies to graphs with tree-width k. We currently have no proof that a k-slim graph also has tree-width $f(k)$ for some function f that depends only on k, but we believe this to be true.

Many graph-theoretic problems which are hard for general graphs can be solved in polynomial time for graphs with bounded tree-width. See, e.g. [13] and [7] for examples of such problems. These algorithms require that the input graph be given together with its *tree-decomposition*. Given the tree-decomposition, these algorithms usually apply some divide-and-conquer technique where the division uses the fact that the solution for the problem can be derived by solving constantly many subproblems on each of the subgraphs induced by the members of subtrees at a given node. Extending a result of Courcelle [6], it is shown by Arnborg et al. in [3] that each graph-theoretic problem that is expressible by an Extended Monadic Second-Order formula, can be solved in linear time on graphs with bounded tree-width. Such problems include, for example, the Hamiltonian Circuit problem, the graph k-colorability problem (where k is fixed) and the H-factor problem (which is to determine if the input graph has a spanning subgraph whose connected components are isomorphic to some fixed graph H). Many other problems fall into this category. It is therefore interesting to devise algorithms for graph-theoretic problems that are not expressible in Extended

Monadic Second-Order logic, which run in polynomial time in case the input graph has bounded tree-width. In this paper we give such an algorithm for many cases of the following well-studied graph-theoretic problem:

The H-decomposition problem: Let H be a fixed graph. Given a graph G, is there a set L of edge-disjoint subgraphs of G such that each member of L is isomorphic to H, and every edge of G appears in a member of L. If such a set L exists, we say that G has an H-decomposition.

It was shown by Dor and Tarsi in [9] that the H-decomposition problem is NP-Complete for every graph H having a connected component with at least three edges. In particular, it is NP-Complete if H is a connected graph with at least three edges. Thus, even if H is a star on $h \geq 4$ vertices, the problem is NP-Complete. Furthermore, the H-decomposition problem is not expressible in Extended Monadic Second-Order logic. This follows from the even stronger fact, pointed to us by M. Fellows, that the H-decomposition problem is not *finite state* even when H is a tree by using the methods shown in [1] and in [10]. We can, however, show the following:

Theorem 1.1 *Let k be a fixed integer, and let H be a star. There exists a polynomial time algorithm that, given a k-slim graph G, finds an H-decomposition of G if one exists.*

We can also solve the H-decomposition problem for a much-wider class of graphs H: Let H be a connected graph such that if S is any nonempty set of vertices of H whose deletion disconnects H (or leaves H with one vertex), then S is not an independent set. We call such graphs *robust*. Thus, complete graphs with 3 or more vertices are robust. Also, complete r-partite graphs with $r \geq 3$ are robust. A wheel on $n \geq 6$ vertices and $n-1$ spokes is robust, although it is not complete r-partite for any r. Bipartite graphs are non-robust, as one may take S to be an entire vertex-class; in particular, stars are non-robust. The property of being robust is monotone-increasing with respect to edge addition.

Theorem 1.2 *Let k be a fixed integer, and let H be a fixed robust graph. There exists a polynomial time algorithm that, given a k-slim graph G, finds an H-decomposition of G if one exists.*

Note that Theorem 1.1 and 1.2 apply, in particular, to graphs with tree-width k. In Section 2 we prove Theorem 1.1 and in Section 3 we prove Theorem 1.2. Concluding remarks and open problems are presented in the final section.

2 Star decomposition in k-slim graphs

In this section we prove Theorem 1.1. Let H be a fixed star on $h+1 \geq 3$ vertices, and h leaves. Note that a graph G has an H-decomposition iff one can orient the edges of G such that the out-

degree of every vertex is 0 mod h. It will be convenient to view the H-decomposition problem in a more general setting, which is the following. A graph G is called *partially-oriented* if some of its edges are oriented. An *h-orientation* of G is an orientation of the non-oriented edges of G such that the out-degree of every vertex in the resulting directed graph is 0 mod h. We shall present an algorithm that, given a partially-oriented graph, finds an h-orientation of it, if one exists. Clearly, this algorithm also solves the H-decomposition problem, as the undirected graph G can be viewed as a partially-oriented graph. Our algorithm will run in polynomial time in case the input graph is k-slim, where k is fixed.

The key idea of our algorithm is the notion of a (K, A, B)-extension. Let $G = (V, E)$ be a partially-oriented k-slim graph, and let K, A, B be a k-separator of G. A (K, A, B)-extension of G is a pair of two partially-oriented graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ for which the following hold:

1. $V_A = A \cup K \cup D_A$, $V_B = B \cup K \cup D_B$ where D_A is disjoint from $A \cup K$ and D_B is disjoint from $B \cup K$.

2. Every edge e adjacent to a vertex of A in G, appears in G_A. The orientation of e in G_A is the same as its orientation in G. If e was non-oriented in G it remains non-oriented in G_A. Similarly, every edge adjacent to a vertex of B in G, appears in G_B, and with the same orientation.

3. Every edge e connecting two vertices of K in G appears in both of G_A and G_B. If e is oriented in G, it keeps the same orientation in both G_A and G_B. If e is not oriented in G, it becomes oriented in G_A and in G_B, and the new orientation is the same in both G_A and G_B.

4. Every vertex v in D_A is connected only to vertices of K, and the edges adjacent to v are all oriented toward v. Similarly, every vertex v in D_B is connected only to vertices of K, and the edges adjacent to v are all oriented toward v. (Note that, in particular, D_A and D_B are independent sets).

5. Let $v \in K$. Let $R_K(v)$ be the number of oriented edges that emanate from v toward a vertex of K, in G_A (or in G_B, it is the same). Similarly put $R_A(v)$, $R_B(v)$, $R_{D_A}(v)$ and $R_{D_B}(v)$. Note that, according to the previous paragraph, $R_{D_A}(v)$ is, in fact, the number of neighbors of v in D_A and, similarly, $R_{D_B}(v)$ is the number of neighbors of v in D_B. Then, it is required that $R_{D_A}(v) < h$ and $R_{D_B}(v) < h$ and that $R_K(v) + R_{D_A}(v) + R_{D_B}(v) = 0 \bmod h$. Thus, $(R_K(v) \bmod h) + R_{D_A}(v) + R_{D_B}(v) \in \{0, h, 2h\}$. In particular, it follows that D_A and D_B have at most $k(h-1)$ non-isolated vertices each, and since isolated vertices play no role

in our h-orientation, we may assume that D_A and D_B contain no isolated vertices. Thus $|D_A|, |D_B| \leq k(h-1)$.

Note that a (K, A, B)-extension is established by two sets of decisions. We first need to decide how to orient the non-oriented edges of G, with both endpoints in K. We then need to define D_A and D_B, namely their sizes and their adjacencies to K, such that item 5 in the list or requirements stated above in satisfied.

Lemma 2.1 *Let G be a partially-oriented k-slim graph and let K, A, B be a k-separator in G. Then G has an h-orientation iff there exists a (K, A, B)-extension such that both G_A and G_B have an h-orientation, and both G_A and G_B are k-slim.*

Proof Assume first that G has an h-orientation, and denote it by G'. Thus, G' is a directed graph such that the outdegree of every vertex is 0 mod h. We must create a (K, A, B)-extension such that both G_A and G_B are k-slim and both have an h-orientation. As noted in the paragraph preceding the lemma, we must first decide on the orientation in G_A and in G_B of an edge e connecting two vertices of K. Our decision will simply be the orientation of e in G'. We must now create D_A and D_B and their adjacencies to K. For $v \in K$, let $R_A(v)$ denote the number of edges, modulo h, of G', directed from v toward a vertex of A. Similarly define $R_B(v)$ and $R_K(v)$. Clearly, $R_A(v) + R_B(v) + R_K(v) \in \{0, h, 2h\}$. Let $X_B(v)$ be a set of $R_B(v)$ vertices of B which are adjacent to v in G' via an edge directed from v. Similarly, define $X_A(v)$. Now put $D_A = \cup_{v \in K} X_B(v)$ and $D_B = \cup_{v \in K} X_A(v)$. Every $v \in K$ is adjacent in G_A to every vertex of $X_B(v)$ via a directed edge emanating from v. Likewise, every $v \in K$ is adjacent in G_B to every vertex of $X_A(v)$ via a directed edge emanating from v. Note that the graphs G_A and G_B are subgraphs of G (although some non-oriented edges of G may be oriented in G_A or in G_B). Thus, G_A and G_B are both k-slim. In fact, the orientation of each oriented edge of G_A and G_B is the same as its orientation in G'. Let G'_A (G'_B) be the directed graphs resulting from G_A (G_B) by orienting each non-oriented edge e of G_A (G_B) in the same way as e is oriented in G'. We now show that the outdegree of every vertex of G'_A (G'_B) is 0 mod h, thus showing that G_A and G_B have an h-orientation. Indeed, let $v \in G'_A$. If $v \in A$ we have no problem since its outdegree in G'_A is identical to its outdegree in G'. If $v \in D_A$ then its outdegree is 0. If $v \in K$ then, by the fact that $R_B(v)$ is equal modulo h to the number of edges emanating from v toward a vertex of B in G', we have that the outdegree of v in G'_A is equal modulo h to the outdegree of v in G', which is 0 mod h. Similar arguments hold when $v \in G'_B$.

We now assume that there exists a (K, A, B)-extension such that both G_A and G_B have an h-orientation. We must show that G also has an h-orientation. Let G'_A and G'_B be h-orientations of G_A and G_B respectively. We create an h-orientation G' of G as follows. Let e be an undirected

edge in G. If e has an endpoint in A then e also appears in G_A. The orientation of e in G' will be the same as its orientation in G'_A. Similarly, if e has an endpoint in B then its orientation in G' will be the same as its orientation in G'_B. If both endpoints of e belong to K then e appears in both G_A and G_B, and it is oriented in the same way in both G_A and G_B, and this orientation will be the orientation of e in G'. We must now show that the outdegree of every vertex of G' is 0 mod h. If $v \in A$ then its outdegree in G' is the same as its outdegree in G'_A, which is 0 mod h. A similar argument holds if $v \in B$. If $v \in K$, let $R_A(v)$ be the number of edges emanating from v toward A in G'_A. Analogously, define $R_B(v)$, $R_K(v)$, $R_{D_A}(v)$ and $R_{D_B}(v)$. Note that $R_K(v)$, $R_{D_A}(v)$ and $R_{D_B}(v)$ are also the number of edges emanating from v toward the respective classes in G_A and G_B, while $R_A(v)$ and $R_B(v)$ only apply to G'_A and G'_B. Now, since G_A and G_B are a (K, A, B)-extension, we know that $R_K(v) + R_{D_A}(v) + R_{D_B}(v) = 0 \bmod h$. Also, since G'_A and G'_B are h-orientations we have $R_A(v) + R_K(v) + R_{D_A}(v) = 0 \bmod h$, and $R_B(v) + R_K(v) + R_{D_B}(v) = 0 \bmod h$. It follows from these three equalities that $R_A(v) + R_B(v) + R_K(v) = 0 \bmod h$. But $R_A(v) + R_B(v) + R_K(v)$ is exactly the outdegree of v in G'. □

Lemma 2.2 *Let k be a fixed positive integer. Let $G = (V, E)$ be a partially-oriented graph on n vertices, where $n > 12kh$. A k-separator K, A, B of G, if it exists, can be found in $O(n)$ time, or otherwise we can decide in $O(n)$ time that G is not k-slim. Furthermore, if K, A, B is a k-separator of G then there are at most $(kh)^{2kh}2^{\binom{k}{2}}$ (K, A, B)-extensions, and they can all be generated in constant time.*

Proof If G is k-slim, a k-separator K, A, B of G exists. Such a K can be found in linear $O(V + E)$ time (under our assumption that k is fixed), by applying an algorithm that uses flow techniques [14]. Once K is found, the connected components of $G \setminus K$ can be computed in linear time, using Breadth First Search. Each connected component can then be assigned to either A or B in a greedy manner, while maintaining that $|A|, |B| \leq (2/3)n$. The fact that $O(V + E) = O(n)$ follows from the result of Mader in [12], which states that a graph on n vertices with no k-connected subgraph has $O(nk)$ edges, and from the fact, mentioned in the introduction, that k-slim graphs have no $3k$-connected subgraphs. If our algorithm failed to find K within the required running time, then we may halt it and output that G is not k-slim.

Given the K, A, B-separator, we now show how all (K, A, B)-extensions can be constructed in constant time. Let P be the set of all the non-oriented edges that have both of their endpoints in K. In each (K, A, B)-extension, all the edges of P must be oriented. There are exactly $2^{|P|}$ ways to orient P, and they can all be generated in $O(2^{|P|})$ time. Since $|P| \leq \binom{k}{2}$, this is $O(2^{\binom{k}{2}})$. We now let P' be a particular orientation of P, and show how to create all (K, A, B)-extensions in which the

edges of P have the orientation P'. This is done as follows. We create two sets of $k(h-1)$ vertices each, which we call D_A and D_B, and which are initially isolated. For each $v \in K$ we can compute in constant time the value $R_K(v)$ which is the number of edges emanating from v toward a vertex of K, taken modulo h. (In fact, we could have computed the $R_K(v)$'s when we generated each orientation of P). We must now decide upon the values $R_{D_A}(v)$ and $R_{D_B}(v)$ for each $v \in K$. Each of these values is at most $h-1$, and we must also have that $R_{D_A}(v) + R_{D_B}(v) + R_K(v) = 0 \bmod h$. Thus there are exactly h ways to select the mutual values of $R_{D_A}(v)$ and $R_{D_B}(v)$. Thus, there are at most h^k choices for the set of $2k$ values of $R_{D_A}(v)$ and $R_{D_B}(v)$ for all $v \in K$ simultaneously. Clearly these sets of numbers can be computed in constant time. Fixing such a set, we must select $R_{D_A}(v)$ vertices from D_A and $R_{D_B}(v)$ vertices from D_B for each $v \in K$ and connect them with v via an edge emanating from v. Once this is done for all $v \in K$, we delete from D_A and D_B the vertices that are still isolated. Thus, we have obtained a (K, A, B)-extension. There are exactly

$$\prod_{v \in K} \binom{k(h-1)}{R_{D_A}(v)} \binom{k(h-1)}{R_{D_B}(v)} < \binom{kh}{h}^{2k}$$

ways to select the adjacent vertices in D_A and D_B for all $v \in K$ simultaneously, although some of the resulting extensions may be isomorphic. Once again, since k and h are constants, these extensions can be generated in constant time. Summing up, we have that there are at most $\binom{kh}{h}^{2k} h^k 2^{\binom{k}{2}}$ extensions which agree with P', and thus there are at most

$$\binom{kh}{h}^{2k} h^k 2^{\binom{k}{2}} \leq (k(kh)^{h-1})^{2k} h^k 2^{\binom{k}{2}} \leq (kh)^{2kh} 2^{\binom{k}{2}}$$

(K, A, B)-extensions. □

Proof of Theorem 1.1: Our algorithm receives a partially-oriented n-vertex graph $G = (V, E)$ as input, and returns with one of the following three possible results:

1. An h-orientation of G.
2. An announcement 'G does not have an h-orientation'.
3. An announcement 'G is not k-slim'.

Furthermore, our algorithm runs in time which is polynomial in n. The algorithm proceeds as follows. If $n \leq 12kh$ we compute an h-orientation of G, if it exists, with brute force. If it does not exist we output that G does not have an h-orientation. If $n > 12kh$, we either find a k-separator K, A, B of G, or decide that G is not k-slim. This can be done in $O(n)$ time by Lemma 2.2. If

we have managed to compute a separator, we continue computing, in constant time, all (K, A, B)-extensions. Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be the pair of graphs of a given extension. Then,

$$|V_A| = |A| + k + |D_A| \leq (2/3)n + k + k(h-1) \leq (3/4)n.$$

Similarly, $|V_B| \leq (3/4)n$. We recursively apply our h-orientation algorithm on G_A and G_B, and do this for all possible extensions. According to Lemma 2.1 we know that if at least one extension in which both G_A and G_B are k-slim satisfies that both G_A and G_B have an h-orientation, then G also has an h-orientation. If no extension in which both G_A and G_B are k-slim satisfies this, then G does not have an h-orientation. Note that some of the extensions that we generate may result in non-slim graphs G_A or G_B, even if G is k-slim. Let $T(n)$ be the overall running time of the algorithm. $T(n)$ measures the number of instructions that the algorithm performs on an input of size n, under any valid computation model (say, the RAM model). Let C_1 be a positive integer such that $T(n) \leq C_1$ whenever $n \leq 12kh$. According to Lemma 2.2 and our recursive implementation we have that $T(n) \leq C_2 n + 2C_3 T(0.75n)$ where C_2 and C_3 are absolute constants. $C_2 n$ represents the running time of computing the separator. C_3 represents the time to compute all possible extensions. $2T(0.75n)$ represents the running time over the two subproblems G_A and G_B that correspond to a given extension. It is shown in [8] (there called the Master Theorem) that such functions like $T(n)$ are polynomial in n, where the degree of the polynomial is $O(\log C_3)$, which, in our case, is $O(kh \log kh)$. □

3 Robust decompositions of k-slim graphs

In this section we prove Theorem 1.2. From here on we let H be a fixed robust graph on h vertices. Let G be a k-slim graph, and let K, A, B be a k-separator of G.

The algorithm for solving the H-decomposition problem is, in fact, a special case of a more general algorithm. In order to describe the general algorithm we need several definitions. Denote the edge-set of H by $E_H = \{1, \ldots, p\}$. Thus, $|E_H| = p$. A graph $G = (V, E)$ is called *partially-labeled* if every edge $e \in E$ is associated with a label $l(e) = l_G(e)$ where either $l(e) = \emptyset$ (a non-labeled edge) or $l(e) = (c, i)$ where c is the *color* of e, chosen from a finite set of possible colors, and $i \in \{1, \ldots, p\}$ is the *role* of e. Furthermore, if $l(e) = (c, i)$ and $l(e') = (c, j)$ then $i \neq j$. In other words, no two labeled edges have the same label. A *labeled H-decomposition* of a partially-labeled graph G is an H-decomposition whose members possess the following two additional properties:

1. If $l(e) = (c, i)$ then e plays the role of the edge i of H in the H-decomposition.

2. If $l(e) = (c, i)$ and $l(e') = (c, j)$ then e and e' belong to the same member of the H-decomposition.

Every member of a labeled H-decomposition of G is called *label-isomorphic* to H, since labeled edges are mapped to their roles under the isomorphism. Our algorithm will solve the labeled H-decomposition problem. Note that such a solution implies a solution to the (unlabeled) H-decomposition problem if we initially define $l_G(e) = \emptyset$ for all $e \in E$.

As in the previous section, the key ingredient in the algorithm is the notion of a (K, A, B)-*extension* of a partially-labeled graph. This is defined as follows. Let $G = (V, E)$ be partially-labeled, and let K, A, B be a k-separator of G. Let $\mathcal{F}$ be the set of colors used in the labels of the edges of G. Let $\mathcal{F}^* \subset \mathcal{F}$ be the set of colors that are used in a label of an edge which has an endpoint in A and also in a label of an edge which has an endpoint in B. A (K, A, B)-extension of G is a set of two partially labeled graphs, $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ for which the following six requirements hold:

1. $V_A = A \cup K \cup D_A$, $V_B = B \cup K \cup D_B$ where D_A is disjoint from $A \cup K$ and D_B is disjoint from $B \cup K$.

2. Every edge e adjacent to a vertex of A in G, appears in G_A. Also, $l_G(e) = l_{G_A}(e)$ (i.e. the edge e keeps its label). Similarly, every edge e adjacent to a vertex of B in G, appears in G_B, and keeps the same label.

3. Every edge e connecting two vertices of K appears in at least one of G_A and G_B, and it may appear in both. Let E_K denote the edges with both endpoints in K, which appear in both G_A and G_B. If $e \notin E_K$ (i.e. e appears only in G_A or only in G_B), then e keeps its label. If $e \in E_K$ and $l_G(e) \neq \emptyset$, then e keeps its label. If $e \in E_K$ and $l_G(e) = \emptyset$ then e becomes labeled in G_A and G_B and $l_{G_A}(e) = l_{G_B}(e)$. Note that in any case, every edge of E_K must be labeled in G_A and G_B, and with the same label in both.

4. Every vertex of D_A may only be adjacent to other vertices of D_A or to vertices of K. Similarly, vertices of D_B may only be adjacent to vertices of $D_B \cup K$. No vertex of D_A or D_B may be isolated.

5. Let $\mathcal{F}_K = \{c \mid e \in E_K,\ l_{G_A}(e) = (c, i)\}$ be the set of colors used by the labels of the edges of E_K. Every edge adjacent to $D_A \cup D_B$ must be labeled with a color from $\mathcal{F}_K$. Furthermore, it is required that $\mathcal{F}^* \subset \mathcal{F}_K$, and also that $\mathcal{F}_K \cap (\mathcal{F} \setminus \mathcal{F}^*) = \emptyset$. Hence, if a color c appears (in G) in an edge adjacent to A and also in an edge adjacent to B, then there must be at least one edge of E_K that has the color c (in G_A and G_B). If c does not have this property

(e.g. if c does not appear in an edge adjacent to A) then c does not appear in E_K. Note that $\mathcal{F}_K \setminus \mathcal{F}^*$ are new colors that do not appear in G.

6. Let $c \in \mathcal{F}_K$. Let A_c (B_c) be the subgraph of G_A (G_B) which is induced by the edges whose color is c. Note that A_c and B_c may share a few edges, namely those edges of E_k which are colored with c. Let A_c^* (B_c^*) be the subgraph of A_c (B_c) induced by the edges of E_k and the edges adjacent to D_A (D_B). Let H_c be the graph union of A_c^* and B_c^*. Then, it is required that H_c be label-isomorphic to H. Furthermore, it is also required that A_c (B_c) be label-isomorphic to the subgraph of H induced by the roles of the edges of A_c (B_c).

Corollary 3.1 *In any (K, A, B)-extension, $|D_A|, |D_B| \leq \binom{k}{2}h$.*

Proof According to requirement 6 in the definition of a (K, A, B)-extension, every edge which is adjacent to D_A belongs to some graph H_c, for $c \in \mathcal{F}_K$. Thus, there are at most h vertices of D_A which have adjacent edges colored with c, for $c \in \mathcal{F}_K$. Since $|\mathcal{F}_K| \leq |E_K| \leq \binom{k}{2}$, and since no edge of D_A is isolated, we have $|D_A| \leq \binom{k}{2}h$. Similar arguments hold for D_B. □

Note that a (K, A, B)-extension is established by the following decisions. We first need to decide, for each edge with both endpoints in K, if it belongs to G_A or G_B or both. The common edges form the set E_K, and we must then label those edges of E_K which were not labeled in G. When assigning these labels care must be taken to preserve a proper labeling, and that each color of $\mathcal{F}^*$ will indeed appear in E_K (if these conditions cannot be satisfied then our choice of E_K is not valid, since no (K, A, B)-extension has this choice as the set of common edges of G_A and G_B). We must then define D_A and D_B by creating these sets, and joining them (via properly labeled edges whose colors are taken from $\mathcal{F}_k$) among themselves or to vertices of K.

Lemma 3.2 *Let $G = (V, E)$ be a partially-labeled k-slim graph and let K, A, B be a k-separator in G. Then G has a labeled H-decomposition iff there exists a (K, A, B)-extension such that both G_A and G_B have a labeled H-decomposition, and both G_A and G_B are k-slim.*

Proof Assume first that G has a labeled H-decomposition. Let $G_1, \ldots G_{|E|/p}$ be the members of such a decomposition. Let $\mathcal{F}$ be the set of colors used in the partial labeling of G. We may assume that $\mathcal{F} = \{1, \ldots, t\}$ where $t \leq |E|/p$. We may thus assume that G_c contains all the edges whose color is c (G_c may also contain unlabeled edges), for $c = 1, \ldots, t$. G_c has all its edges unlabeled for $c = t+1, \ldots, |E|/p$. We now define a (K, A, B)-extension of G such that both G_A and G_B have a labeled H-decomposition. For $c = 1, \ldots, |E|/p$ let A_c and B_c be the subgraphs of G_c induced by the edges adjacent to A and B respectively. Let K_c be the subgraph of G_c induced by the edges

both of whose endpoints belong to K. We create a set of $|B|$ vertices which we call D'_A, where each vertex of D'_A is *assigned* to a vertex of B. Hence, for each $x \in B$ there is a unique vertex $b(x) \in D'_A$. Similarly, we create $|A|$ vertices which we call D'_B where each is assigned to a vertex of $|A|$. Hence, for each $x \in A$ there is a unique vertex $a(x) \in D'_B$. The following process will connect some of the vertices of D'_A to vertices of $D'_A \cup K$ and some of the vertices of D'_B to vertices of $D'_B \cup K$. The vertices of D'_A and D'_B that remain disconnected at the end of the process will be discarded and the remaining vertices of D'_A and D'_B will form D_A and D_B respectively. We now define our process of creation of G_A and G_B, which is done for each $c = 1, \ldots, |E|/p$.

1. The edges of A_c belong only to G_A and the edges of B_c belong only to G_B.
2. If A_c is empty then the edges of K_c (if there are any) belong only to G_B.
3. Otherwise, if B_c is empty then the edges of K_c (if there are any) belong only to G_A.
4. Otherwise, both A_c and B_c are non-empty. This means that $c \in \mathcal{F}^*$. In this case the edges of K_c belong to both G_A and G_B, i.e. they belong to E_K. Note that, crucially, K_c is non-empty since H is robust, and thus G_c, being isomorphic to H, is also robust. Now consider an edge $e = (x, y) \in G_c$, and suppose e is mapped to the edge i of H under the labeled isomorphism between G_c and H. Exactly one of the following six possibilities applies to e:
   (a) If $e \in K_c$ and e is labeled, we do nothing.
   (b) If $e \in K_c$ and e is unlabeled, we must now label it because $e \in E_K$. We thus put $l_{G_A}(e) = l_{G_B}(e) = (c, i)$.
   (c) If $e \in A_c$ and $x \in A$ and $y \in A$ we create the edge $(a(x), a(y))$ (this is an edge connecting two vertices of D'_B in G_B), and we label it (c, i).
   (d) If $e \in A_c$ and $x \in A$ and $y \in K$ we create the edge $(a(x), y)$ in G_B, and label it (c, i).
   (e) If $e \in B_c$ and $x \in B$ and $y \in B$ we create the edge $(b(x), b(y))$ (this is an edge connecting two vertices of D'_A in G_A), and we label it (c, i).
   (f) If $e \in B_c$ and $x \in B$ and $y \in K$ we create the edge $(b(x), y)$ in G_A, and label it (c, i).

   Note, in particular, that all the edges of K_c are labeled and their color is c. This fact shows that every color in $\mathcal{F}^*$ appears in E_K, which is one of our requirements from an extension.

Clearly, the graph G_A that we have constructed is isomorphic to a subgraph of G, and thus G_A is k-slim. The same holds for G_B. Our construction also shows that G_A and G_B form a (K, A, B)-extension. Note also that each c such that $A_c \neq \emptyset$ defines a unique subgraph isomorphic to G_c

in G_A, which is also label isomorphic to H, and that the union of these subgraphs coincides with G_A. Thus G_A has a labeled H-decomposition. Similar arguments show that G_B has a labeled H-decomposition.

For the other direction, we assume that we have a (K, A, B)-extension such that both G_A and G_B have a labeled H-decomposition. We construct a labeled H-decomposition L of G as follows. Consider a member G' of the H-decomposition of G_A that contains no edge from E_K. We claim that G' does not contain any edge which is adjacent to D_A. To see this, note that if G_A contained some edge e which is adjacent to some vertex of D_A, then e must be labeled. In particular, e is colored by some color c. But our requirement from an extension is that $c \in \mathcal{F}_k$. Thus there is some edge $e' \in E_k$ which has the color c. Since $e' \notin G'$ this means that there is another member G'' of the labeled H-decomposition containing e'. Hence two distinct members contain edges having the same color c. This cannot happen in a labeled H-decomposition. We have thus shown that G' is entirely a subgraph of G, and we make G' a member of L. Note also that the labels of G' in G are identical to its labels is G_A, thus G' is label-isomorphic to H. Similarly, a member of the H-decomposition of G_B that contains no edge from E_K will also be a member of L.
Finally, let G' be a member of the H-decomposition of G_A that contains an edge $e \in E_K$. Since e must be labeled in G_A, we assume $l(e) = (c, i)$. Thus, all labeled edges of G' are colored with c. Let G'' be the member of the H-decomposition of G_B that contains e. Such a member must exist since edges of E_K appear in both G_A and G_B. Since the label of e in G'' is also (c, i), we have that G'' contains all the edges of G_B that are colored with c. Let A_c (B_c) be the edges of G' (G'') which are adjacent to vertices of $A \cup K$ ($B \cup K$), and which are not in E_K. Let K_c be the common edges of G' and G'' (namely the edges belonging to E_K and whose color is c). Let D_{Ac} and D_{Bc} be the edges of G' and G'' which are adjacent to vertices of D_A and D_B respectively. Clearly, $A_c \; cup K_c \cup D_{Ac}$ are all the edges of G' and $B_c \cup K_c \cup D_{Bc}$ are all the edges of G''. We know from item 6 in the list of requirements from an extension that the graph H_c induced by $K_c \cup D_{Ac} \cup D_{Bc}$ is label-isomorphic to H, and every edge in it is labeled. Let G^* be the subgraph of G induced by $A_c \cup B_c \cup K_c$. Clearly, every labeled edge of G^* is colored with c. It remains to show that G^* is label-isomorphic to H, since we can then add G^* to L. We first show how to map every edge $e \in G^*$. If $e \in A_c \cup K_c$, let i be the edge of E_H that e is mapped to under the labeled isomorphism between G' and H. If $e \in B_c \cup K_c$ let i be the edge of E_H that e is mapped to under the labeled isomorphism between G'' and H. (Note that if $e \in K_c$ there is no conflict in the definition of i, since $e \in E_K$ in this case and it is thus labeled in G' and G'', and with the same label in both). In any case, we map e to i in the labeled isomorphism between G^* and H that we are constructing. We must now show that our mapping is one-to-one. Suppose e and f are two distinct edges of G^*,

where e is mapped to i and f is mapped to j. If $\{e, f\} \subset A_c \cup K_c$ or $\{e, f\} \subset B_c \cup K_c$ then, clearly, $i \neq j$. If $e \in A_c$ and $f \in B_c$ then i cannot be a role of an edge of $D_{Ac} \cup K_c$. Thus, i must be a role of some edge of D_{Bc} (here we use the fact that H_c is label-isomorphic to H, and all p distinct roles appear in H_c). Similarly, j must be a role of some edge of D_{Ac}. Thus, once again, $i \neq j$. We have shown that our mapping is one-to-one. The fact that this mapping is a graph isomorphism between G^* and H is a straightforward consequence of the fact that H_c, G' and G'' are all label-isomorphic to H. $\square$

The next lemma is similar to Lemma 2.2 in the previous section.

Lemma 3.3 *Let k be a fixed positive integer. Let $G = (V, E)$ be a partially-labeled graph on n vertices, where $n > 12k^2h$. A k-separator K, A, B of G, if it exists, can be found in $O(n)$ time, or otherwise we can decide in $O(n)$ time that G is not k-slim. Furthermore, if K, A, B is a k-separator of G then:*

1. *Let $\mathcal{F}^*$ be the set colors that appear in a label of an edge adjacent to A and also in a label of an edge adjacent to B. Then $\mathcal{F}^*$ can be computed in $O(n)$ time.*
2. *If $|\mathcal{F}^*| > \binom{k}{2}$ then G does not have a labeled H-decomposition.*
3. *If $|\mathcal{F}^*| \leq \binom{k}{2}$ then all the (K, A, B)-extensions can be computed in constant time.*

Proof The first part of the lemma is identical to the first part of Lemma 2.2, and so we continue with the second part.

1. We scan the edges adjacent to A and create a list of all colors used in these edges. Similarly, we create a list of all colors used by edges adjacent to B. These operations take time which is proportional to the number of edges adjacent to A and B, which is $O(n)$. Since there are $O(n)$ elements in each of these two lists, the lists can be sorted in $O(n)$ time with bucket sort, and then their intersection can be computed in $O(n)$ time, producing $\mathcal{F}^*$.

2. In every (K, A, B)-extension, every member of $\mathcal{F}^*$ must appear in a color of a label of some edge of E_K (recall that E_K are the edges which are shared between G_A and G_B and have both of their endpoints in K). Since, clearly, $|E_K| \leq \binom{k}{2}$, it follows that if $|\mathcal{F}^*| > \binom{k}{2}$ then there is no (K, A, B)-extension. Thus, according to Lemma 3.2, G does not have a labeled H-decomposition.

3. We now assume that $|\mathcal{F}^*| \leq \binom{k}{2}$ and show that the number of (K, A, B)-extensions can be bounded by a constant, and how We must first decide, for each edge with both endpoints in K, if it will belong to G_A or G_B or both. There are at most $3^{\binom{k}{2}}$ such choices. Fixing

such a choice, we denote by E_K the common edges of G_A and G_B, and we must label each non-labeled edge of E_K. The colors in these labels may only be taken from $\mathcal{F}^*$ or may be new colors that are not used in G. In a proper extension, each color in $\mathcal{F}^*$ must appear in an edge of E_K. Thus, there are at most $|E_K| - |\mathcal{F}^*|$ new colors. Altogether, each non-labeled edge of E_K may be assigned a color from a set of at most $|E_K| \leq \binom{k}{2}$ possible colors. We must also assign a role for each non-labeled edge of E_K. There are p possible roles. Thus, the number of ways to label all non-labeled edges of E_K is at most $(\binom{k}{2}p)^{|E_K|}$. Once a labeling is determined we must check its validity. If a color of $\mathcal{F}^*$ does not appear in E_K, the labeling is not valid. Also, if the same label appears in an edge of E_K, and in another edge of G, (there are at most $\binom{k}{2}p$ edges of G which may have a label that appears also in a vertex of E_K), the labeling is not valid. Since k and p are fixed, and since $\mathcal{F}^*$ has already been computed, these validity checks can be done in constant time; in fact, in $O(k^2p)$ time. Consider a labeling of E_K which passed these validity tests. Let $\mathcal{F}_K$ be the set of colors used in the labels of the edges of E_K. We must now define the sets D_A and D_B, and their adjacent edges. Note that, according to item 6 in the list of requirements of a K, A, B-extension, the graph induced by E_K and the adjacent edges to D_A and D_B (all these edges are labeled in the extension), is simply an edge-disjoint union of graphs H_c which are label-isomorphic to H, for all $c \in \mathcal{F}_K$. We must therefore consider all the possible ways to generate such an edge-disjoint union. This is done as follows: Let K_c be the set of edges of E_K which are colored by c. Let A_c and B_c be the edges adjacent to A and B respectively, which are colored by c (if c is a new color then A_c and B_c are empty). Let R_c be the set of roles of K_c. R_c is non-empty since K_c is non-empty. Let X'_c be the set of roles in A_c and let Y'_c be the set of roles in B_c. We compute all possible triples R_c, X_c, Y_c such that $X'_c \subset X_c$ and $Y'_c \subset Y_c$, and $R_c \cup X_c \cup Y_c = \{1, \ldots, p\} = E_H$, and X_c and Y_c are non-empty. For every such triple, we check whether the subgraph of H induced by R_c separates H such that the subgraph of H induced by X_c is separated from the subgraph induced by Y_c. Such a triple is called *good*. For each $c \in \mathcal{F}_K$ let R_c, X_c, Y_c be a good triple. Such a set of triples naturally defines D_A and D_B and their adjacencies, since for each c, a good triple R_c, X_c, Y_c defines H_c. Checking that a set of $|\mathcal{F}_K|$ good triples is edge-disjoint can also be done in constant time, since these are graphs of constant size, and there is a constant number of them. For each such valid set, we must also verify whether the subgraph of G induced by $K_c \cup A_c \cup B_c$ contains a spanning subgraph which is label-isomorphic to the subgraph of H induced by the roles of $K_c \cup A_c \cup B_c$. For each $c \in \mathcal{F}_K$ there are at most 2^p triples R_c, X_c, Y_c, as for each role that does not appear in $R_c \cup X'_c \cup Y'_c$ we must decide if it appears in X_c or Y_c. Thus, there are at most $2^{p|\mathcal{F}_K|}$ ways to obtain a valid extension, after

deciding upon E_K and its labeling. Altogether, there are at most

$$3^{\binom{k}{2}} \cdot (\binom{k}{2}p)^{|E_K|} \cdot 2^{p|\mathcal{F}_K|} \leq 3^{k^2}(k^2p)^{k^2}(2^p)^{k^2} = (3k^2p2^p)^{k^2}.$$

□

Having proved Lemmas 3.3 and 3.2, the proof of Theorem 1.2 is almost identical to the proof of Theorem 1.1. Let $G = (V, E)$ be an n-vertex graph which is k-slim. Our algorithm proceeds as follows. If $n \leq 12k^2h$ we compute an H-decomposition in G, if it exists, with brute force. Otherwise, we find a k-separator K, A, B of G in $O(n)$ time. This can be done by Lemma 3.3. We then compute, in constant time, all (K, A, B)-extensions. Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be the pair of graphs of a given extension. Then,

$$|V_A| = |A| + k + |D_A| \leq (2/3)n + k + \binom{k}{2}h \leq (2/3)n + k^2h \leq (3/4)n.$$

Similarly, $|V_B| \leq (3/4)n$. We recursively apply the labeled H-decomposition algorithm on G_A and G_B, and do this for all possible extensions. According to Lemma 3.2 we know that if at least one extension in which both G_A and G_B are k-slim satisfies that both G_A and G_B have a labeled H-decomposition, then G also has a labeled H-decomposition. If no extension in which both G_A and G_B are k-slim satisfies this, then G does not have a labeled H-decomposition. Let $T(n)$ be the overall running time of the algorithm. $T(n)$ measures the number of instructions that the algorithm performs on an input of size n, under any valid computation model (say, the RAM model). Let C_1 be a positive integer such that $T(n) \leq C_1$ whenever $n \leq 12k^2h$. According to Lemma 3.3 and our recursive implementation we have that $T(n) \leq C_2n + 2C_3T(0.75n)$ where C_2 and C_3 are absolute constants. C_2n represents the running time of computing the separator and the set $\mathcal{F}^*$. C_3 represents the time to compute all possible extensions. $2T(0.75n)$ represents the running time over the two subproblems G_A and G_B that correspond to a given extension. As in Theorem 1.1, we use the result in [8] to obtain that $T(n)$ is polynomial in n, where the degree of the polynomial is $O(\log C_3)$, which, in our case, is $O(k^2p\log(k^2p))$. □

4 Concluding remarks and open problems

The class of graphs H for which we have a polynomial time algorithm solving the H-decomposition problem in k-slim graphs, consists of the robust graphs and the stars. It is an interesting open problem if this can be extended to all fixed graphs H. In particular, is the problem polynomial in case $H = P_4$, the path on 4 vertices. Note that P_4 is not a star and is non-robust as one can take the second vertex of the path as an independent separator. A relaxation of this open problem

(but not necessarily a simpler one) is to assume the stricter requirement that the input graph has tree-width k.

In case the input graph G is assumed to be with bounded tree-width, and also with bounded maximum degree, the H-decomposition problem can be expressed with a monadic second order formula, and thus the problem can be solved in linear time in this case. However, this is not very exciting as this family of graphs is very limited.

Arnborg, Lagergren and Seese note, and demonstrate, in [3] that extended monadic second order logic captures many natural graph-theoretic problems. Our result shows that there are natural graph theoretic problems that are not known to be expressible in this logic, and are still efficiently solvable in graphs with bounded tree-width k, and even in the larger family of k-slim graphs.

Finally, we note that almost every graph is robust, where by "almost every" we mean that a random graph on h vertices is robust with probability tending to one as h tends to infinity (we assume the $G(n, 1/2)$ model). This is because the maximal independent set is almost always $O(\log h)$, while the vertex-connectivity of the random graph is almost always linear in h (cf. [4]). Thus, say, for every two positive integers k and h, the family of h-vertex graphs for which edge-decomposition can be solved in k-slim graphs in polynomial time contains, say, 99 percent of all h-vertex graphs, provided $h > h_0$ where h_0 is an absolute constant.

5 Acknowledgment

The authors thank the referee for important comments and suggestions, and for pointing out to us that the H-decomposition problems which we handle are not finite-state.

References

[1] K. Abrahamson and M. Fellows, *Finite automata, bounded treewidth and well-quasiordering*, In: Graph Structure Theory, N. Robertson and P. Seymour eds. AMS Contemporary Math. 147 (1993), 539-564.

[2] N. Alon, P. D. Seymour and R. Thomas, *A separator theorem for non-planar graphs*, Journal of the American Mathematical Society 3 (1990), 801-808.

[3] S. Arnborg, J. Lagergren and D. Seese, *Easy problems for tree-decomposable graphs*, Journal of Algorithms 12 (1991), 308-340.

[4] B. Bollobás, *Random Graphs*, Academic Press, 1985.

[5] J. A. Bondy and U.S. R. Murty, *Graph Theory with Applications*, Macmillan Press, London, 1976.

[6] B. Courcelle, *The monadic second order logic of graphs III: Treewidth, forbidden minors and complexity issues*, Report 8552, University Bordeaux 1, 1988.

[7] C.J. Colbourn and E.S. El-Mallah, *Partial k-tree algorithms*, Congressus Numerantium 64 (1988) 105-119.

[8] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to algorithms*, The MIT Press, 1990.

[9] D. Dor and M. Tarsi, *Graph decomposition is NPC - A complete proof of Holyer's conjecture*, Proceedings of the 20^{th} ACM Symposium on the Theory of Computing, ACM Press (1992), 252-263.

[10] M. Fellows, M. Hallett and H. Wareham, *DNA Physical Mapping: three ways difficult*, Proceedings of the 1^{st} Annual European Symposium on Algorithms, Springer Verlag LNCS vol. 726 (1993), 157-168.

[11] R. Lipton and R. E. Tarjan, *Applications of a planar separator theorem*, SIAM Journal of Computing 36 (1979), 177-189.

[12] W. Mader, *Connectivity and edge connectivity in finite graphs*, In: Surveys in Combinatorics, B. Bollobás, Ed. Cambridge University Press, London, 1979.

[13] N. Robertson and P. Seymour, *Graph minors II. Algorithmic aspects of tree-width*, Journal of Algorithms 7 (1986), 309-322.

[14] N. Robertson and P. Seymour, *Graph minors XIII. The disjoint paths problem*, Manuscript, 1986.