FPT algorithms for packing k-safe spanning rooted sub(di)graphs

Stéphane Bessy, Florian Hörsch, Ana Karolinna Maia, Dieter Rautenbach, Ignasi Sau

▶ **To cite this version:**

Stéphane Bessy, Florian Hörsch, Ana Karolinna Maia, Dieter Rautenbach, Ignasi Sau. FPT algorithms for packing k-safe spanning rooted sub(di)graphs. Discrete Applied Mathematics, 2024, 346, pp.80-94. 10.1016/j.dam.2023.11.026 . lirmm-04352990

HAL Id: lirmm-04352990
https://hal-lirmm.ccsd.cnrs.fr/lirmm-04352990

Submitted on 19 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

FPT algorithms for packing k-safe spanning rooted sub(di)graphs

Stéphane Bessy
LIRMM, Université de Montpellier, CNRS, Montpellier, France
stephane.bessy@lirmm.fr

Florian Hörsch
Technisch Universität Ilmenau, Ilmenau, Germany
florian.hoersch@tu-ilmenau.de

Ana Karolinna Maia
Departamento de Computação, Universidade Federal do Ceará, Fortaleza, Brazil
karolmaia@ufc.br

Dieter Rautenbach
Institute of Optimization and Operations Research, Ulm University, Germany
dieter.rautenbach@uni-ulm.de

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, Montpellier, France
ignasi.sau@lirmm.fr

Abstract

We study three problems introduced by Bang-Jensen and Yeo [Theor. Comput. Sci. 2015] and by Bang-Jensen, Havet, and Yeo [Discret. Appl. Math. 2016] about finding disjoint "balanced" spanning rooted substructures in graphs and digraphs, which generalize classic packing problems such as detecting the existence of mutiple arc-disjoint spanning arborescences. Namely, given a positive integer k, a digraph $D = (V, A)$, and a root $r \in V$, we first consider the problem of finding two arc-disjoint k-safe spanning r-arborescences, meaning arborescences rooted at a vertex r such that deleting any arc rv and every vertex in the sub-arborescence rooted at v leaves at least k vertices. Then, we consider the problem of finding two arc-disjoint (r, k)-flow branchings meaning arc sets admitting a flow that distributes one unit from r to every other vertex while respecting a capacity limit of $n - k$ on every arc. We show that both these problems are FPT with parameter k, improving on existing XP algorithms. The latter of these results answers a question of Bang-Jensen, Havet, and Yeo [Discret. Appl. Math. 2016]. Further, given a positive integer k, a graph $G = (V, E)$, and $r \in V$, we consider the problem of finding two edge-disjoint (r, k)-safe spanning trees meaning spanning trees such that the component containing r has size at least k when deleting any vertex different from r. We show that this problem is also FPT with parameter k, again improving on a previous XP algorithm. Our main technical contribution is to prove that the existence of such spanning substructures is equivalent to the existence of substructures with size and maximum (out-)degree both bounded by a (linear or quadratic) function of k, which may be of independent interest.

2012 ACM Subject Classification Design and analysis of algorithms → Fixed parameter tractability.

Keywords and phrases Digraphs, packing problems, arborescences, branching flows, safe spanning trees, parameterized complexity, fixed-parameter tractability.

Funding *Stéphane Bessy*: DIGRAPHS (ANR-19-CE48-0013-02).
Ana Karolinna Maia: FUNCAP Pronem 4543945/2016 and CAPES/STIC-AmSud 88881.197438/2018-01.
Ignasi Sau: DEMOGRAPH (ANR-16-CE40-0028), ESIGMA (ANR-17-CE23-0010), ELIT (ANR-20-CE48-0008-01), and French-German Collaboration ANR/DFG Project UTMA (ANR-20-CE92-0027).

1 Introduction

This article deals with finding certain disjoint substructures in graphs and digraphs. Throughout the article, when given a graph or a digraph, we use n for its number of vertices.

All graphs and digraphs considered in this paper are loopless, but may have multiple edges or arcs. Given a graph $G = (V, E)$, we say that an edge $e = uv$ is *incident* to u and v. For some $X \subseteq V$, we denote by $d_G(X)$ the number of edges that are incident to exactly one vertex in X. For some $v \in V$, we use $N_G(v)$ for the set of vertices w such that there is an edge in E between v and w. Further, we use $\delta_G(v)$ for the set of edges incident to v and $d_G(v)$ for $|\delta_G(v)|$. A vertex $v \in V$ with $d_G(v) = 1$ is called a *leaf* of G. A *rooted graph* is a graph $G = (V + r, E)$ with a special vertex r called the *root*. Note that $r \notin V$ and $V(G) = V + r$. If $G = (r, \emptyset)$, we say that r is a *terminal* of G. Otherwise, the set of *terminals* in G is the set of leaves in V.

Given a digraph $D = (V, A)$ and some $X \subseteq V$, we use $\delta^-_D(X)$ (resp. $\delta^+_D(X)$) for the set of arcs entering (resp. leaving) X. We use $d^-_D(X)$ (resp. $d^+_D(X)$) for $|\delta^-_D(X)|$ (resp. $|\delta^+_D(X)|$). For a single vertex v, we abbreviate $\delta^+_D(\{v\})$ (resp. $\delta^-_D(\{v\}), d^+_D(\{v\}), d^-_D(\{v\})$) to $\delta^+_D(v)$ (resp. $\delta^-_D(v), d^+_D(v), d^-_D(v)$). We call $d^-_D(v)$ (resp. $d^+_D(v)$) the *in-degree* (resp. *out-degree)* of v. We use $N^+_D(v)$ for the set of vertices w such that there is an arc in A from v to w. A vertex $v \in V$ with $d^+(v) = 0$ is called a *sink* of D. Subscripts may be omitted when they are clear from the context. The *underlying graph* of D is obtained by replacing each arc of A by an edge between the same two vertices. A *rooted digraph* is a digraph $D = (V + r, A)$ with a special vertex r, called the *root*, whose in-degree is 0. As before, observe that $r \notin V$ and $V(D) = V + r$.

Packing k-safe spanning r-arborescences. The first objects we deal with are called arborescences. We remark here that the term *out-branching* is used in [3] to describe the same object. An *r-arborescence* is a directed graph $X = (V + r, A)$ such that the underlying graph of X is a tree, the in-degree of r is 0 in X, and the in-degree of all other vertices is 1 in X. We say that r is the root of X. Observe that every $v \in V$ is the root of a unique maximal subarborescence of X. We denote this subarborescence by B^v_X. Given a digraph D, an r-arborescence X that is a subdigraph of D is *spanning* in D if it has the same vertex set as D. The following is a fundamental result in digraph theory.

▶ **Theorem 1** (Edmonds [6]). *Let $D = (V + r, A)$ be a rooted digraph and k a positive integer. There exists a set of k arc-disjoint spanning r-arborescences in D if and only if $d^-_D(X) \geq k$ for every $\emptyset \neq X \subseteq V$.*

A number of alternative proofs of Theorem 1 have been found, several of which are algorithmic and yield polynomial-time algorithms to find the desired arc-disjoint spanning r-arborescences, if they exist [7, 11].

This naturally raises the question whether we also can efficiently find spanning arborescences satisfying some extra properties. This consideration, as well as practical applications concerning protection against arc failures, motivated Bang-Jensen and Yeo [4] to introduce the notion of k-safe spanning arborescences. For a positive integer k, an r-arborescence $X = (V + r, A)$ is called *k-safe* if $n - |V(B^v_X)| \geq k$ for every $v \in V$. Notice that it is enough that only the out-neighbours of r satisfy this latter condition for X to be k-safe. They proved a negative result showing that in general even the problem of finding a *single* k-safe spanning r-arborescence is hard. More specifically, they showed that deciding whether a rooted digraph $D = (V + r, A)$ has an $(n - k)$-safe spanning r-arborescence is NP-complete for any fixed $k \geq 3$.

In this light, a characterization in the shape of Theorem 1 clearly seems out of reach. It nevertheless remains interesting to investigate the possibility of finding arc-disjoint k-safe spanning r-arborescences for small values of k. This question has been dealt with by Bang-Jensen, Havet, and Yeo [3]. On the negative side, they implicitly proved the following result, which shows that there is little hope to algorithmically find arc-disjoint $k(n)$-safe spanning r-arborescences if k is a function that does not grow too slowly. While a polynomial-time algorithm for the problem they consider would not imply $\mathsf{P} = \mathsf{NP}$, it would imply the failure of the *Exponential Time Hypothesis* (ETH for short) of Impagliazzo and Paturi [9], stating that there is an $\varepsilon > 0$ such that there is no algorithm for solving a 3-SAT formula with ℓ variables and m clauses in time $2^{\varepsilon\ell} \cdot (\ell + m)^{O(1)}$.

▶ **Theorem 2** (Bang-Jensen, Havet, and Yeo [3])**.** *Suppose that the* ETH *holds, let $\varepsilon > 0$ be arbitrary and let $k : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ be a function such that $(\log(n))^{1+\varepsilon} \leq k(n) \leq \frac{n}{2}$ for every $n > 0$. Further, suppose that there exists a constant C^* such that for every integer $c \geq C^*$ there exists an n such that $k(n) = c$. Then there is no algorithm running in time $n^{O(1)}$ for deciding whether a given rooted digraph $D = (V + r, A)$ has two arc-disjoint $k(n)$-safe spanning r-arborescences.*

In particular, it is possible to obtain the following lower bound for the running time of any algorithm deciding whether a given rooted digraph $D = (V + r, A)$ has two arc-disjoint $k(n)$-safe spanning r-arborescences.

▶ **Corollary 3.** *Unless the* ETH *fails, there is no algorithm deciding whether a given rooted digraph $D = (V + r, A)$ has two arc-disjoint k-safe spanning r-arborescences that runs in $2^{c \cdot k^{1-\varepsilon}} \cdot n^{O(1)}$ time for some $\varepsilon > 0$ and $c > 0$.*

Indeed, if an algorithm as above existed, for $k(n) := (\log(n))^{1+\varepsilon}$, we would obtain an algorithm running in time $2^{c \cdot (\log(n))^{(1+\varepsilon)(1-\varepsilon)}} \cdot n^{O(1)} = n^{O(1)}$, contradicting Theorem 2.

On the positive side, they show that the problem becomes tractable when fixing the value of k. While the main results considered in this article hold for finding an *arbitrary number* of disjoint objects, we only present the case where we want to find just *two* of them, in order to avoid technicalities. In Section 7 we discuss the generalization to more than two objects.

▶ **Theorem 4** (Bang-Jensen, Havet, and Yeo [3])**.** *Deciding whether a given rooted digraph $D = (V + r, A)$ contains two arc-disjoint k-safe spanning r-arborescences is* XP *with parameter k.*

The definition of the classes XP and FPT can be found in Subsection 2.5. Our first contribution is to improve Theorem 4 by showing that the problem is *fixed-parameter tractable* (FPT).

▶ **Theorem 5.** *Deciding whether a given rooted digraph $D = (V + r, A)$ contains two arc-disjoint k-safe spanning r-arborescences is* FPT *with parameter k. More precisely, it can be solved in time $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. Further, if they exist, the two arc-disjoint k-safe spanning r-arborescences can be computed within the same running time.*

Packing (r, k)-flow branchings. The second structure we consider, which was introduced by Bang-Jensen and Bessy [1], builds a connection between the theory of finding arc-disjoint spanning arborescences and flow problems. A *flow* in a rooted digraph $X = (V + r, A)$ is a function $\mathbf{z} : A \to \mathbb{Z}_{\geq 0}$. For some $A' \subseteq A$, we abbreviate $\sum_{a \in A'} \mathbf{z}(a)$ to $\mathbf{z}(A')$. Recall that $n = |V| +$. Given a capacity function $\mathbf{c} : A \to \mathbb{Z}_{\geq 0}$, an $(r, n - \mathbf{c})$-*branching flow* is a flow $\mathbf{z} : A \to \mathbb{Z}_{\geq 0}$ such that $\mathbf{z}(a) \leq \mathbf{c}(a)$ for every $a \in A$, $\mathbf{z}(\delta^+(r)) - \mathbf{z}(\delta^-(r)) = n - 1$ and

$\mathbf{z}(\delta^-(v)) - \mathbf{z}(\delta^+(v)) = 1$ for every $v \in V$. In other words, the idea is that $\mathbf{z}$ reaches all vertices of X from r and each vertex other than r retains one unit of flow. If X admits an $(r, n - \mathbf{c})$-branching flow, we say that X is an $(r, n - \mathbf{c})$-*flow branching*. If for some positive integer k, we have $\mathbf{c}(a) = n - k$ for all $a \in A$, we speak of an (r, k)-*branching flow* and an (r, k)-*flow branching*. Given a digraph D, an (r, k)-flow branching X that is a subdigraph of D is *spanning* in D if it has the same vertex set as D. If D admits a k-safe spanning r-arborescence, then it is easy to see that D is an (r, k)-flow branching, but the converse is not necessarily true, as pointed out in [3]. In particular, observe that the underlying graph of an arc-minimal (r, k)-flow branching is not necessarily acyclic.

Bang-Jensen and Bessy [1] showed that the problem of deciding if a rooted digraph has two arc-disjoint spanning flow branchings is NP-complete if every arc has capacity at most two. This result has been strengthened in [3] to the case of capacities all being equal to some fixed $k \geq 2$. The following result is proven in [3]. It shows that there is little hope to algorithmically find arc-disjoint spanning $(r, k(n))$-flow branchings if k is a function that does not grow too slowly. It can be viewed as an analogue of Theorem 2 for (r, k)-flow branchings.

▶ **Theorem 6** (Bang-Jensen, Havet, and Yeo [3]). *Suppose that* ETH *holds, let $\varepsilon > 0$ be arbitrary and let $k : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ be a function such that $(\log(n))^{1+\varepsilon} \leq k(n) \leq \frac{n}{2}$ for every $n > 0$. Further, suppose that there exists a constant C^* such that for every integer $c \geq C^*$ there exists an n such that $k(n) = c$. Then there is no algorithm running in time $n^{O(1)}$ for deciding whether a given rooted digraph $D = (V + r, A)$ has two arc-disjoint spanning $(r, k(n))$-flow branchings.*

In a similar way as Corollary 3 can be concluded from Theorem 2, the following result is obtained from Theorem 6.

▶ **Corollary 7.** *Unless the* ETH *fails, there is no algorithm deciding whether a given rooted digraph $D = (V + r, A)$ has two arc-disjoint spanning (r, k)-flow branchings that runs in $2^{c \cdot k^{1-\varepsilon}} \cdot n^{O(1)}$ time for some $\varepsilon > 0$ and $c > 0$.*

On the other hand, the problem of finding arc-disjoint spanning (r, k)-flow branchings for some small values of k turns out to be more tractable. The study of such spanning flow branchings has surprisingly many similarities with the study of k-safe spanning arborescences. Bang-Jensen and Bessy [1] showed that the case $k = 1$ can be solved in polynomial time. This result was again generalized by Bang-Jensen, Havet, and Yeo [3], who proved that the problem can be solved in polynomial time for every fixed value of $k \geq 1$. The following result can be viewed as an analogue of Theorem 4 for (r, k)-flow branchings.

▶ **Theorem 8** (Bang-Jensen, Havet, and Yeo [3]). *Deciding whether a given rooted digraph $D = (V + r, A)$ contains two arc-disjoint spanning (r, k)-flow branchings is* XP *with parameter k.*

The authors of [3] ask whether the above problem is FPT. Our second contribution is an affirmative answer to this question.

▶ **Theorem 9.** *Deciding whether a given rooted digraph $D = (V + r, A)$ contains two arc-disjoint spanning (r, k)-flow branchings is* FPT *with parameter k. More precisely, it can be solved in time $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. Further, if they exist, the two arc-disjoint spanning (r, k)-flow branchings can be computed within the same running time.*

Packing (r, k)-safe spanning trees. Finally, we consider a similar problem in undirected graphs that has also been introduced in [3]. Given a graph $G = (V, E)$, a *spanning tree* is a subgraph T of G that is a tree with $V(T) = V$. The theory of finding edge-disjoint spanning trees is also pretty rich. The most fundamental result is the following one.

▶ **Theorem 10** (Tutte [12]). *Let $G = (V, E)$ be a graph and p a positive integer. Then G has p edge-disjoint spanning trees if and only if $\sum_{i=1}^{q} d_G(V_i) \geq 2p(q-1)$ for every partition $\{V_1, \ldots, V_q\}$ of V.*

An algorithmic proof of Theorem 10, yielding a polynomial-time algorithm for finding the spanning trees, if they exist, can be found as the proof of Theorem 10.5.1 in [8]. Again, we may wish to also find edge-disjoint spanning trees satisfying certain extra properties. Given a rooted tree $T = (V + r, E)$ and some $v \in V$, we use C_T^v for the subgraph of $T - v$ that arises from deleting the component of $T - v$ containing r. Observe that $v \notin V(C_T^v)$. We say that T is (r, k)-*safe* if for every $v \in V$, we have $|V - V(C_T^v)| \geq k$.

▶ **Theorem 11** (Bang-Jensen, Havet, and Yeo [3]). *Deciding whether a given rooted graph $G = (V + r, E)$ contains two edge-disjoint (r, k)-safe spanning trees is* XP *with parameter k.*

Again, we improve Theorem 11 as follows.

▶ **Theorem 12.** *Deciding whether a given rooted graph $G = (V + r, E)$ contains two edge-disjoint (r, k)-safe spanning trees is* FPT *with parameter k. More precisely, it can be solved in time $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. Further, if they exist, the two edge-disjoint (r, k)-safe spanning trees can be computed within the same running time.*

Since a hardness result in the spirit of Theorem 2 and Theorem 6 was not provided in [3], we fill this gap and prove the following theorem, whose proof is inspired by the one of [3, Theorem 5.2]. It shows that the problem is hard even if we want to find one single (r, k)-safe spanning tree.

▶ **Theorem 13.** *Let $p \geq 1$ be a fixed positive integer. Deciding whether a given rooted graph $G = (V + r, E)$ has p edge-disjoint (r, k)-safe spanning trees is* NP*-complete. Moreover, let $\varepsilon > 0$ be arbitrary and let $k : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ be a function such that $(\log(n))^{2+\varepsilon} \leq k(n) \leq \frac{n}{2}$ for all $n > 0$. Further, suppose that there exists a constant C^* such that for every integer $c \geq C^*$ there exists an n such that $k(n) = c$. Then, assuming the* ETH*, there is no algorithm running in time $n^{O(1)}$ for deciding whether a given rooted graph has p edge-disjoint $(r, k(n))$-safe spanning trees.*

Again, in a similar way as Corollary 3 can be concluded from Theorem 2, the following result is obtained from Theorem 13.

▶ **Corollary 14.** *Unless the* ETH *fails, there is no algorithm deciding whether a given rooted digraph $D = (V + r, A)$ has two edge-disjoint spanning (r, k)-safe spanning trees that runs in $2^{c \cdot k^{\frac{1}{2}-\varepsilon}} \cdot n^{O(1)}$ time for some $\varepsilon > 0$ and $c > 0$.*

Our techniques. In order to obtain the FPT algorithms for the three considered problems, we follow a common strategy. In a nutshell, the main ideas used in the XP algorithms of [3], and how we manage to improve them to FPT algorithms, can be summarized as follows. The algorithms of Bang-Jensen, Havet, and Yeo [3] are all based on proving the following general property for each of the considered problems, where all the substructures are rooted:

> A given (di)graph contains the required substructure $\mathcal{X}$ (i.e., a pair of disjoint k-safe spanning arborescences, flow branchings, or spanning trees) if and only if it contains another type of substructure $\mathcal{X}'$ (i.e., a pair of appropriately defined disjoint objects) of *size bounded by a function of k* and that can be extended to the substructure $\mathcal{X}$ in *polynomial time*.

We would like to stress that the above property is deliberately abstract in order to capture the essential common property that can be used for all the considered problems. More informally, it states that, instead of finding the required substructure $\mathcal{X}$, which may have arbitrary size (in particular, not depending only on k), we can safely focus on finding another substructure $\mathcal{X}'$, with the guarantee that it can *always* be extended to $\mathcal{X}$ in polynomial time, and with the following crucial property that is exploited in the corresponding algorithms, as discussed below: the size of $\mathcal{X}'$ is bounded by a function depending only on the parameter k.

Once the above property is proved, an XP algorithm follows naturally: generate all candidate substructures $\mathcal{X}'$ in time $n^{f(k)}$ and, for each of them, try to extend it to a substructure $\mathcal{X}$ in polynomial time. Our main contribution is to prove that the above general property is still true if we replace $\mathcal{X}'$ with another type of substructure $\mathcal{X}''$ having the crucial property that the candidate substructures $\mathcal{X}''$ can be enumerated in time $f(k) \cdot n^{O(1)}$, hence yielding an FPT algorithm. In order to achieve this, we prove that we can restrict ourselves to objects $\mathcal{X}''$ whose "non-sink" vertices (i.e., those with positive (out-)degree in $\mathcal{X}''$) have *(out-)degree, in the original (di)graph, bounded by some function of* k, namely $O(k)$ or $O(k^2)$. This is possible because, given a pair $\mathcal{X}' = \{X_1, X_2\}$ containing a vertex v of large (out-)degree (as a function of k) in, say, X_1, we can safely prune the "branch" of X_1 hanging from v, with the guarantee that it will always be possible to extend the pruned substructure to another substructure of the original type. Note that since the substructures $\mathcal{X}''$ have size *and* maximum (out-)degree bounded by a function of k, we can indeed generate all candidate substructures in time $f(k) \cdot n^{O(1)}$, as required. We now make this informal explanation more concrete. For all technical definitions used in the next paragraphs, see Section 2.

Let us first focus on the problem of finding two arc-disjoint k-safe spanning r-arborescences. In this case, the substructure $\mathcal{X}'$ is an *extendable pair of arc-disjoint classic* (r,k)*-kernels*[1]. In Lemma 20, we restate a result from [3] that shows that the existence of this substructure is sufficient for the existence of two arc-disjoint k-safe spanning r-arborescences. We then introduce compact (r,k)-kernels. An *extendable pair of arc-disjoint compact* (r,k)*-kernels* corresponds to the substructure $\mathcal{X}''$. In Lemma 28, we show that the existence of an extendable pair of arc-disjoint compact (r,k)-kernels is also sufficient for the existence of two arc-disjoint k-safe spanning r-arborescences. The proof of Lemma 28 is our main technical contribution. Having Lemma 28 at hand, the proof of Theorem 5 is easy.

As for packing (r,k)-flow branchings, the substructure $\mathcal{X}'$ defined in [3] is an *extendable pair of arc-disjoint classic* (r,k)*-cores*. In Lemma 24, we restate a result from [3] that shows that the existence of this substructure is sufficient for the existence of two arc-disjoint (r,k)-flow branchings. We then introduce compact (r,k)-cores. An *extendable pair of arc-disjoint compact* (r,k)*-cores* corresponds to the substructure $\mathcal{X}''$. In Lemma 30, we show that the existence of an extendable pair of arc-disjoint compact (r,k)-cores is also sufficient for the existence of two arc-disjoint (r,k)-flow branchings. Again, the proof of Lemma 30 is our main technical contribution and is similar to the one of Lemma 28. Again, having Lemma 30 at hand, the proof of Theorem 9 is easy.

Finally, for packing (r,k)-safe spanning trees, the substructure $\mathcal{X}'$ defined in [3] is a *completable pair of edge-disjoint classic* (r,k)*-certificates*. In Lemma 26, we restate a result from [3] that shows that the existence of this substructure is sufficient for the existence of two edge-disjoint (r,k)-safe spanning trees. We then introduce compact (r,k)-certificates. A *completable pair of edge-disjoint compact* (r,k)*-certificates* corresponds to the substructure

[1] This notion should *not* be confused with the standard term '*kernel*' from parameterized complexity, as defined in Subsection 2.5.

$\mathcal{X}''$. In Lemma 32, we show that the existence of a completable pair of edge-disjoint compact (r,k)-certificates is also sufficient for the existence of two edge-disjoint (r,k)-safe spanning trees. Again, the proof of Lemma 32 is our main technical contribution and is similar to the ones of Lemma 28 and Lemma 30. However, while the proofs of Lemma 28 and Lemma 30 use a property on extendable structures proven in [3], the proof of Lemma 32 relies on a spanning tree reconfiguration argument. Again, having Lemma 32 at hand, the proof of Theorem 12 is easy.

Organization. In Section 2 we review some more technical results we need from [3], prove several preliminary results, and provide the basic definitions about parameterized complexity. In Section 3, Section 4, Section 5, and Section 6 we give the proof of Theorem 5, Theorem 9, Theorem 12, and Theorem 13, respectively. Finally, we conclude our work in Section 7.

2 Preliminaries

In this section we collect some more technical previous results and prove several preliminary statements. We first give some general results on graphs and digraphs and then some which are more specific to each of the particular applications. We also provide some basic definitions about parameterized complexity.

2.1 General preliminaries

The following is a well-known submodularity property of digraphs that can be found, for instance, in [8, Proposition 1.2.1].

▶ **Proposition 15.** *Let $D=(V,A)$ be a digraph and $S_1,S_2\subseteq V$. Then $d^-_D(S_1)+d^-_D(S_2)\geq d^-_D(S_1\cup S_2)+d^-_D(S_1\cap S_2)$.*

A rooted digraph $D=(V+r,A)$ is called *ℓ-root-connected* if $d^-_D(X)\geq\ell$ for every $X\subseteq V$. We use *root-connected* for 1-root-connected. In a root-connected rooted digraph $D=(V+r,A)$, an arc $a\in A$ is called *critical* if $D-a$ is not root-connected anymore. We now use Proposition 15 to obtain a result that will be useful when dealing with both k-safe spanning r-arborescences and spanning (r,k)-flow branchings.

▶ **Lemma 16.** *Let $D=(V+r,A)$ be a 2-root-connected rooted digraph and let $D'=(V+r,A')$ be a root-connected rooted digraph that is obtained from D by deleting α arcs of A. Then for any $v\in V+r$, there are at most α arcs which are critical in D' and whose tail is v.*

Proof. We proceed by induction on α. The statement is trivial for $\alpha=0$. We suppose that it holds for all integers up to some α and show that it also holds for $\alpha+1$. Let $v\in V+r$ and let $\{a_1,\ldots,a_{\alpha+1}\}$ be a set of arcs in A such that $D_2=D-\{a_1,\ldots,a_{\alpha+1}\}$ is root-connected. By the inductive hypothesis, v is the tail of at most α critical arcs in $D_1=D-\{a_1,\ldots,a_\alpha\}$. Suppose, for the sake of a contradiction, that there are two arcs vw_1,vw_2 which are critical in D_2, but not in D_1. It follows that there are sets $X_1,X_2\subseteq V$ such that $d^-_{D_2}(X_1),d^-_{D_2}(X_2)=1$, $d^-_{D_1}(X_1),d^-_{D_1}(X_2)=2$, and vw_i enters X_i. Since $D_2=D_1-a_{\alpha+1}$, it follows that $a_{\alpha+1}$ enters $X_1\cap X_2$, so $X_1\cap X_2\neq\emptyset$. As D_2 is root-connected, we have $d^-_{D_2}(X_1\cap X_2)\geq 1$. As vw_i enters X_i, we have $v\in V+r-(X_1\cup X_2)$ and so both vw_1 and vw_2 enter $X_1\cup X_2$. This yields $d^-_{D_2}(X_1)+d^-_{D_2}(X_2)=1+1<1+2\leq d^-_{D_2}(X_1\cap X_2)+d^-_{D_2}(X_1\cup X_2)$, a contradiction to Proposition 15. ◀

Given a 2-root-connected rooted digraph $D=(V+r,A)$, a pair of subdigraphs $(X_1=(V_1+r,A_1),X_2=(V_2+r,A_2))$ of D is called *extendable* if both $D-A_1$ and $D-A_2$ are

root-connected. The following is an immediate consequence of the fact that checking whether a digraph is root-connected can clearly be done in polynomial time using a breadth-first search algorithm.

▶ **Lemma 17.** *Given a rooted digraph $D = (V + r, A)$ and a pair of two subdigraphs $(X_1 = (V_1 + r, A_1), X_2 = (V_2 + r, A_2))$, we can decide in polynomial time whether (X_1, X_2) is extendable.*

We now switch to undirected graphs. The following is an easy known result, whose proof is provided for the sake of completeness.

▶ **Lemma 18.** *Let $G = (V, E)$ be a graph and let T be a spanning tree of G. Let $e = uv \in E - E(T)$. Then there is some $f \in E(T)$ that is incident to u such that $T + e - f$ is a spanning tree of G.*

Proof. The graph $T + e$ contains a unique cycle C such that $uv \in E(C)$ and the deletion of an arbitrary edge of $E(C)$ yields a spanning tree of G. As C is is a cycle, $E(C)$ contains an edge f different from e that is incident to u. This edge satisfies the condition. ◀

Given two spanning trees T_1, T_2 of a graph G, we say that a function $\sigma : E(T_1) \to E(T_2)$ is a *tree-mapping* function if for every $e \in E(T_1)$, both $T_1 - e + \sigma(e)$ and $T_2 - \sigma(e) + e$ are spanning trees of G. It is well-known that a tree-mapping function exists for any two spanning trees T_1, T_2 of a graph G, see for example [8, Theorem 5.3.3].

▶ **Lemma 19.** *Let $G = (V, E)$ be a graph, let T_1, T_2 spanning trees of G, $\sigma : E(T_1) \to E(T_2)$ a tree-mapping function from T_1 to T_2 and $F \subseteq \delta_{T_1}(v)$ for some $v \in V$. Then $\{\sigma(e) : e \in F\}$ contains at least $\lceil \frac{|F|}{2} \rceil$ distinct elements.*

Proof. It suffices to prove that there are no three distinct edges $e_1, e_2, e_3 \in F$ with $\sigma(e_1) = \sigma(e_2) = \sigma(e_3)$. Let $e_1, e_2, e_3 \in F$. As T_1 is a spanning tree, $T_1 - \{e_1, e_2, e_3\}$ contains three components C_1, C_2, C_3 none of which contains v such that e_i is incident to a vertex in $V(C_i)$ for $i = 1, 2, 3$. As $T_1 - e_i + \sigma(e_i)$ is a spanning tree, we obtain that $\sigma(e_i)$ is incident to a vertex in $V(C_i)$ for $i = 1, 2, 3$. As $V(C_1), V(C_2)$ and $V(C_3)$ are pairwise disjoint, the statement follows. ◀

2.2 Preliminaries on k-safe spanning r-arborescences

Given a rooted digraph $D = (V + r, A)$ and a positive integer k, a *classic (r, k)-kernel* is an r-subarborescence $X = (V' + r, A')$ of D such that X is k-safe and $|V'| = 2k - 2$. The XP algorithm of Theorem 4 is based on the following result, which we reformulate here using our terminology.

▶ **Lemma 20** (Bang-Jensen, Havet, and Yeo [3])**.** *Let k be a positive integer and $D = (V + r, A)$ a rooted digraph with $|V| \geq 2k - 2$. Then D contains two arc-disjoint k-safe spanning r-arborescences if and only if D contains an extendable pair of arc-disjoint classic (r, k)-kernels. Further, the two arc-disjoint k-safe spanning r-arborescences can be constructed from the extendable pair of classic (r, k)-kernels in polynomial time.*

2.3 Preliminaries on spanning (r, k)-flow branchings

We first need the following result that allows to recognize (r, k)-flow branchings.

▶ **Lemma 21** (Bang-Jensen, Havet, and Yeo [3])**.** *Given a rooted digraph $D = (V + r, A)$ and a positive integer k, we can decide in polynomial time whether D is an (r, k)-flow branching.*

Given a digraph $D = (V, A)$ and two vertices $u, v \in V$, a *uv-path flow* is a flow $\mathbf{z}$ such that $\mathbf{z}(a) = 1$ for all arcs $a \in A(P)$ and $\mathbf{z}(a) = 0$ for every $a \in A - A(P)$ for some uv-path P. Similarly, a *cycle flow* is a flow $\mathbf{z}$ such that $\mathbf{z}(a) = 1$ of all $a \in A(C)$ and $\mathbf{z}(a) = 0$ for every $a \in A - A(C)$ for some cycle C. We need the following result on flows which is proven in a more general form in [2].

▶ **Lemma 22.** *Let $X = (V + r, A)$ be a flow branching and $\mathbf{z} : A \to \mathbb{Z}_{\geq 0}$ be a branching flow in X. Then there is an rv-path flow $\mathbf{z}_v$ for every $v \in V$ and a set of cycle flows $\{\mathbf{z}_C : C \in \mathcal{C}\}$ for some set of cycles $\mathcal{C}$ such that $\mathbf{z} = \sum_{v \in V} \mathbf{z}_v + \sum_{C \in \mathcal{C}} \mathbf{z}_C$.*

We now use Lemma 22 to prove an important property of arc-minimal (r, k)-flow branchings. A rooted digraph $X = (V + r, A)$ is called *triple-free* if it does not contain more than two arcs in the same direction between the same two vertices.

▶ **Lemma 23.** *Let k be a positive integer and $X = (V + r, A)$ an arc-minimal (r, k)-flow branching with $|V| \geq 2k - 1$. Then X is triple-free.*

Proof. Suppose that X contains three arcs a_1, a_2, a_3 whose tail is u and whose head is v for some $u, v \in V + r$. Further, let $\mathbf{z} : A \to \mathbb{Z}_{\geq 0}$ be an (r, k)-branching flow. By Lemma 22, we obtain that $\mathbf{z} = \sum_{v \in V} \mathbf{z}_v + \sum_{C \in \mathcal{C}} \mathbf{z}_C$ where $\mathbf{z}_v$ is an rv-path flow for every $v \in V$ and $\mathbf{z}_C$ is a cycle flow for every $C \in \mathcal{C}$ for some set of cycles $\mathcal{C}$. Let $\mathbf{z}' = \sum_{v \in V} \mathbf{z}_v$. Observe that $\mathbf{z}'$ is an (r, k)-branching flow. As all of the $\mathbf{z}_v$ are path flows, we obtain $\mathbf{z}'(a_1) + \mathbf{z}'(a_2) + \mathbf{z}'(a_3) \leq |V| < 2(n - k)$. It follows that we can define a flow $\mathbf{z}'' : A \to \mathbb{Z}_{\geq 0}$ such that $\mathbf{z}''(a_1) + \mathbf{z}''(a_2) = \mathbf{z}'(a_1) + \mathbf{z}'(a_2) + \mathbf{z}'(a_3)$, $\mathbf{z}''(a_3) = 0$, $\min\{\mathbf{z}''(a_1), \mathbf{z}''(a_2)\} \leq n - k$ and $\mathbf{z}''(a) = \mathbf{z}'(a)$ for every $a \in A - \{a_1, a_2, a_3\}$. It is easy to see that $\mathbf{z}''$ is an (r, k)-branching flow, so $X - a_3$ is an (r, k)-flow branching, a contradiction to the minimality of X. ◀

Given a positive integer k and a rooted digraph $D = (V + r, A)$ with $|V + r| \geq 2k$, a *classic (r, k)-core* is an (r, k)-flow branching $X = (V' + r, A')$ that is a subdigraph of D with $|V'| = 2k - 1$. The XP algorithm of Theorem 8 is based on the following result, again reformulated using our terminology.

▶ **Lemma 24** (Bang-Jensen, Havet, and Yeo [3])**.** *Let k be a positive integer and $D = (V + r, A)$ be a rooted digraph with $|V| \geq 2k - 1$. Then D contains two arc-disjoint spanning (r, k)-flow branchings if and only if D contains an extendable pair of arc-disjoint classic (r, k)-cores. Further, the two arc-disjoint spanning (r, k)-flow branchings can be constructed in polynomial time from the extendable pair of arc-disjoint classic (r, k)-cores.*

2.4 Preliminaries on (r, k)-safe spanning trees

We first define an equivalent for extendability in undirected graphs. Given a rooted graph $G = (V + r, E)$, a pair of subtrees (X_1, X_2) is called *completable*[2] if there are edge-disjoint spanning trees T_1, T_2 of G such that $E(X_i) \subseteq E(T_i)$. The following result that allows to test completability can be established using matroid theory as mentioned in [3].

▶ **Lemma 25** (Bang-Jensen, Havet, and Yeo [3])**.** *Given a graph $G = (V + r, E)$ and a pair of subtrees (X_1, X_2), we can decide in polynomial time whether (X_1, X_2) is completable.*

[2] Note that we require a completable pair of trees to be edge-disjoint. This is in contrast with the fact that, in Subsection 2.2 and Subsection 2.3, we do *not* require the elements in an extendable pair to be arc-disjoint. We adopt this asymmetric choice for technical reasons arising from the proofs.

Given a positive integer k and a rooted graph $G = (V + r, E)$ with $|V + r| \geq 2k - 1$, a *classic* (r, k)*-certificate* is an (r, k)-safe subtree $X = (V' + r, E')$ of G with $|V'| = 2k - 2$. The XP algorithm of Theorem 11 is based on the following result, again restated using our terminology.

▶ **Lemma 26** (Bang-Jensen, Havet, and Yeo [3]). *Let $G = (V + r, E)$ be a rooted graph and k a positive integer. Then G contains two edge-disjoint (r, k)-safe spanning trees if and only if G contains a completable pair of classic (r, k)-certificates. Further, given a completable pair of classic (r, k)-certificates, we can compute two edge-disjoint (r, k)-safe spanning trees in polynomial time.*

2.5 Preliminaries on parameterized complexity

We refer the reader to [5] for basic background on parameterized complexity, and we recall here only some basic definitions used in this article. A *parameterized problem* is a decision problem whose instances are pairs $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is called the *parameter*. A parameterized problem L is *fixed-parameter tractable* (FPT) if there exists an algorithm $\mathcal{A}$, a computable function f, and a constant c such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an FPT *algorithm*) correctly decides whether $I \in L$ in time bounded by $f(k) \cdot |I|^c$. A parameterized problem L is *slice-wise polynomial* (XP) if there exists an algorithm $\mathcal{A}$ and two computable functions f, g such that given an instance $I = (x, k)$, $\mathcal{A}$ (called an XP *algorithm*) correctly decides whether I is a positive instance of L in time bounded by $f(k) \cdot |I|^{g(k)}$.

A *kernelization algorithm*, or just *kernel*, for a parameterized problem L is an algorithm $\mathcal{A}$ that, in polynomial time, generates from an instance $I = (x, k)$ of L an equivalent instance $I' = (x', k')$ of L such that $|x'| + k' \leq f(k)$, for some computable function $f : \mathbb{N} \to \mathbb{N}$. If $f(k)$ is bounded from above by a polynomial of the parameter k, we say that L admits a *polynomial kernel*. It is well-known [5] that a parameterized problem is FPT if and only if it admits a (not necessarily polynomial) kernel.

3 An FPT algorithm for packing k-safe spanning arborescences

This section is concerned with proving Theorem 5. Given a rooted digraph $D = (V + r, A)$ and a positive integer k, we say that a vertex $v \in V + r$ is *large* if $|N_D^+(v)| \geq 6k - 5$, and *small* otherwise. We let L_D (resp. S_D) be the set of vertices in $V + r$ which are large (resp. small) in D.

We are now ready to introduce a new notion of (r, k)-kernels for k-safe spanning r-arborescences. A *compact* (r, k)*-kernel* is a subdigraph $X = (V' + r, A')$ of D with $|V'| \leq 2k - 2$ satisfying the following:

- X is an r-arborescence,
- all vertices in $(V' + r) \cap L_D$ are sinks of X, and
- a k-safe r-arborescence Y can be obtained from X by adding a set V^* of $2k - 2 - |V'|$ new vertices and adding an arc from a vertex in $(V' + r) \cap L_D$ to v for every $v \in V^*$.

Observe that Y is not necessarily a subdigraph of D. We now give the following algorithmic result.

▶ **Proposition 27.** *Let $D = (V + r, A)$ be a rooted digraph, k a positive integer, and $X = (V' + r, A')$ a subdigraph of D with $|V'| \leq 2k - 2$. We can test in time $2^{O(k \log k)}$ whether X is a compact (r, k)-kernel. Further, if this is the case, we can find a k-safe r-arborescence Y with the properties mentioned in the definition of compact (r, k)-kernels in the same running time.*

Proof. We first verify whether X is an r-arborescence such that all the vertices in $(V'+r)\cap L_D$ are sinks in X. If this is the case, we add a set V^* of $2k-2-|V'|$ new vertices to X. We then test all possibilities to add one arc from $(V+r)\cap L_D$ to v for every $v\in V^*$. As $|V'+r|\leq 2k$ and $|V^*|\leq 2k$, there are at most $(2k)^{2k}=2^{O(k\cdot\log k)}$ possibilities to check. For each of these possibilities, we can then check in polynomial time whether the obtained graph is a k-safe arborescence. ◀

The following result shows that compact (r,k)-kernels can be used in a similar way as classic (r,k)-kernels.

▶ **Lemma 28.** *Let k be a positive integer and $D=(V+r,A)$ be a 2-root-connected rooted digraph with $|V|\geq 2k-2$. Then D contains two arc-disjoint k-safe spanning r-arborescences if and only if D contains an extendable pair of arc-disjoint compact (r,k)-kernels. Further, given an extendable pair of arc-disjoint compact (r,k)-kernels, we can find a pair of arc-disjoint k-safe spanning r-arborescences in time $2^{O(k\log k)}\cdot n^{O(1)}$.*

Proof. By Lemma 20, for the first part it suffices to prove that D contains a pair of arc-disjoint extendable compact (r,k)-kernels if and only if D contains a pair of arc-disjoint extendable classic (r,k)-kernels.

First let $(X_1=(V_1+r,A_1),X_2=(V_2+r,A_2))$ be an extendable pair of arc-disjoint classic (r,k)-kernels. Let $X_i'=(V_i'+r,A_i')$ be obtained from X_i by deleting $B^v_{X_i}-v$ for every $v\in(V_i+r)\cap L_D$. By construction, the X_i' are r-arborescences and all vertices in $(V_i'+r)\cap L_D$ are sinks in X_i'. Let Y_i be obtained from X_i' by adding the vertices in V_i-V_i' and adding an arc from a vertex $v\in(V_i'+r)\cap L_D$ to a vertex $w\in V_i-V_i'$ whenever $w\in V(B^v_{X_i})-v$. Observe that Y_i is an arborescence with $|V(Y_i)|=2k-1$. Further, note that $|V_i+r-V(B^v_{Y_i})|=|V_i+r-V(B^v_{X_i})|\geq k$ for every $v\in V_i'$ and $|V(B^v_{Y_i})|=0$ for every $v\in V_i-V_i'$. This yields that $|V_i+r-V(B^v_{Y_i})|\geq|V_i+r-V(B^v_{X_i})|\geq k$ for every $v\in V_i$ and so Y_i is a k-safe arborescence. By definition, we obtain that (X_1',X_2') is a pair of arc-disjoint compact (r,k)-kernels. Further, $D-A_i$ is a subdigraph of $D-A_i'$ that is root-connected, so $D-A_i'$ is root-connected as well. This yields that (X_1',X_2') is extendable.

Now let $(X_1=(V_1+r,A_1),X_2=(V_2+r,A_2))$ be an extendable pair of arc-disjoint compact (r,k)-kernels. By definition, there are k-safe arborescences Y_1,Y_2 such that Y_i is obtained from X_i by adding a set V_i^* of $2k-2-|V_i|$ new vertices and an arc from a vertex in $(V_i+r)\cap L_D$ to v for every $v\in V_i^*$. Let $(X_1'=(V_1'+r,A_1'),X_2'=(V_2'+r,A_2'))$ be a pair of subdigraphs of D that are vertex-maximal with the following properties:

- **(i)** X_i' is obtained from X_i by repeatedly adding another vertex $v\in V-V_i$ and an arc of A that goes from a vertex in $(V_i+r)\cap L_D$ to v,
- **(ii)** $d^+_{X_i'}(v)\leq d^+_{Y_i}(v)$ for every $v\in(V_i+r)\cap L_D$,
- **(iii)** A_1' and A_2' are disjoint, and
- **(iv)** (X_1',X_2') is extendable.

Note that (X_1,X_2) satisfies conditions **(i)**-**(iv)**, so (X_1',X_2') is well-defined. Also observe that if condition **(ii)** is satisfied with equality for every $v\in(V_i+r)\cap L_D$, then X_i' is isomorphic to Y_i, so X_i' is a k-safe arborescence and by definition also a classic (r,k)-kernel. If this is the case for both X_1' and X_2', we are done by conditions **(iii)** and **(iv)**.

We may therefore suppose by symmetry that there is a vertex $v\in(V_1+r)\cap L_D$ with $d^+_{X_1'}(v)<d^+_{Y_1}(v)$. For any $a=vz_a$ with $z_a\in N^+_D(v)-V_1'$, let $X_1^a=(V_1'+r+z_a,A_1'\cup a)$. By the maximality of X_1', we obtain that X_1^a violates one of conditions **(i)**-**(iv)** for every $a\in\delta^+_D(v)$ with $z_a\in N^+_D(v)-V_1'$. By construction and the choice of v, X_1^a satisfies conditions

(i) and **(ii)** for every $a \in \delta_D^+(v)$ with $z_a \in N_D^+(v) - V_1'$. If X_1^a violates **(iii)**, then $z_a \in V_2'$. By **(ii)**, we have $|V_2'| \leq |V_2 \cup V_2^*| = 2k-2$ and so this is the case for at most $2k-2$ vertices $z_a \in N_D^+(v) - V_1'$. If X_1^a does not satisfy **(iv)**, then a is critical in $D - A_1'$. As D is 2-root-connected and X_1' is an arborescence by construction, Lemma 16 implies that this is the case for at most $|A_1'| = |V_1'| \leq |V_1 \cup V_1^*| = 2k-2$ vertices $z_a \in N_D^+(v) - V_1'$. As $v \in L_D$ and $|V_1'| \leq |V_1 \cup V_1^*| = 2k-2$, we have $|N_D^+(v) - V_1'| \geq (6k-5) - (2k-2) > (2k-2) + (2k-2)$, so there is at least one vertex in $z \in N_D^+(v) - V_1'$ and an arc $a = vz$ such that X_1^a does not violate any of conditions **(i)**-**(iv)**, a contradiction.

Observe that the second part of the proof yields an algorithm for computing a pair of arc-disjoint extendable classic (r,k)-kernels from a pair of arc-disjoint extendable compact (r,k)-kernels. First observe that by the second part of Proposition 27, we can find Y_1, Y_2 in $2^{O(k \log k)}$ time. Now, defining X_2^a similarly to X_1^a, every time we try to add an arc $a = vz$ to X_i' for $i = 1$ or 2, we test, if (X_i^a, X_{3-i}') satisfies conditions **(i)**-**(iv)**. Conditions **(i)**-**(iii)** can clearly be checked in polynomial time and, by Lemma 17, condition **(iv)** can also be checked in polynomial time. Never testing an arc that is parallel to one that we have tested already, after at most $4k-4$ failed attempts, we manage to add a new vertex to V_i'. We repeat this procedure $|V_i^*| \leq 2k-2$ times. It follows that a pair of arc-disjoint extendable classic (r,k)-kernels can be computed in time $k^2 \cdot n^{O(1)} = n^{O(1)}$. By the second part of Lemma 20, we can then find the arc-disjoint k-safe spanning arborescences in D in $n^{O(1)}$ time. Therefore, the overall running time of the algorithm is $2^{O(k \log k)} \cdot n^{O(1)}$, as claimed. ◀

We are now ready to proceed to the proof of Theorem 5.

Proof of Theorem 5. We may suppose that there are at most two parallel arcs from u to v for any $u, v \in V + r$. If $|V| < 2k-2$, the problem can be solved by a brute force algorithm in time $2^{O(k^2)}$, by generating all pairs of subdigraphs of D and checking whether any of these pairs satisfies the required conditions. We may hence also suppose that $|V| \geq 2k-2$.

By Theorem 1 and the remark after, we can first decide in time $n^{O(1)}$ if D is 2-root-connected. If it is not, the answer is negative by Theorem 1, so we may suppose it is.

We first explain how to test whether a candidate for the extendable pair of arc-disjoint compact (r,k)-kernels is indeed an extendable pair of arc-disjoint compact (r,k)-kernels. After, we show that the number of candidates is bounded. Let $X_1 = (V_1 + r, A_1), X_2 = (V_2 + r, A_2)$ be two subdigraphs of D with $|V_1|, |V_2| \leq 2k-2$. By the first part of Proposition 27, we can check in time $2^{O(k \cdot \log k)}$ whether both X_1 and X_2 are compact (r,k)-kernels. By Lemma 17, we can therefore decide in time $2^{O(k \cdot \log k)} + n^{O(1)}$ if (X_1, X_2) is an extendable pair of arc-disjoint compact (r,k)-kernels in D. By Lemma 28, it therefore suffices to prove that there are at most $2^{O(k^2 \cdot \log k)}$ possible candidates for the extendable pair of arc-disjoint compact (r,k)-kernels, and that these can be generated within this running time.

Let $X = (V' + r, A')$ be a compact (r,k)-kernel in D. Observe that by definition of compact (r,k)-kernels, all vertices in $L_D \cap (V' + r)$ are sinks in X. Further, as X can be extended to a k-safe arboresecence on $2k-2$ vertices, B_v^X contains at most $k-1$ vertices for every $v \in V'$. In particular, every vertex in V' can be reached from r by a directed path all of whose interior vertices are in S_D and whose length is at most $k-1$. As every vertex in S_D has at most $6k-6$ out-neighbors, we obtain that the number of vertices in V that can be reached by such a path is at most $(6k-6) + (6k-6)^2 + \ldots + (6k-6)^{k-1} \leq (6k)^k$. As V' contains at most $2k-2$ vertices, there are at most $\binom{(6k)^k}{2k-2} \leq (6k)^{2k^2}$ possibilities to choose V'.

Now suppose that we have chosen V' of size $2k-2$. As there are at most two arcs in the same direction between any two vertices, there are at most $4\binom{|V'+r|}{2} \leq 16k^2$ arcs that

have their head and tail in $V' + r$. As $|A'| = 2k - 2$, there are at most $\binom{16k^2}{2k-2} \leq (16k^2)^{2k}$ possibilities to choose A'. It follows that there are at most $(6k)^{2k^2} \cdot (16k^2)^{2k}$ possibilities to choose a compact (r,k)-kernel X. As these can be computed by a brute force method, the algorithm can finish after checking less than $f(k) = \binom{(6k)^{2k^2} \cdot (16k^2)^{2k}}{2} = 2^{O(k^2 \cdot \log k)}$ candidates for the extendable pair of arc-disjoint compact (r,k)-kernels.

If no extendable pair of arc-disjoint compact (r,k)-kernels exists, by Lemma 28, D does not contain two arc-disjoint k-safe spanning arborescences. On the other hand, once we have found an extendable pair of arc-disjoint compact (r,k)-kernels, we can compute the two arc-disjoint k-safe spanning r-arborescences in polynomial time by the second part of Lemma 28. The overall running time of the obtained algorithm is $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. ◀

4 An FPT algorithm for packing spanning (r,k)-flow branchings

This section is concerned with proving Theorem 9. Slightly modifying the terminology introduced in Section 3, given a 2-root-connected digraph $D = (V + r, A)$ and a positive integer k, we say that a vertex v is *large* if $|N_D^+(v)| \geq 20k^2 + 1$, and *small* otherwise. Again, we let L_D (resp. S_D) be the set of vertices in $V + r$ which are large (resp. small) in D.

We are now ready to introduce a new notion of (r,k)-cores for spanning (r,k)-flow branchings. A *compact* (r,k)*-core* is a subdigraph $X = (V' + r, A')$ of D with $|V'| \leq 2k - 1$ satisfying the following:

- all vertices in $(V' + r) \cap L_D$ are sinks in X and
- an (r,k)-flow branching Y can be obtained from X by adding a set V^* of $2k - 1 - |V'|$ new vertices and adding an arc from a vertex in $(V' + r) \cap L_D$ to v for every $v \in V^*$.

Observe that Y is not necessarily a subdigraph of D. We now give the following algorithmic result which plays the analogous role of Proposition 27 for flow branchings.

▶ **Proposition 29.** *Let $D = (V + r, A)$ be a rooted digraph, k a positive integer, and $X = (V' + r, A')$ a subdigraph of D with $|V'| \leq 2k - 1$. We can test in time $2^{O(k \log k)}$ whether X is a compact (r,k)-core. Further, if this is the case, we can find an (r,k)-flow branching Y with the properties mentioned in the definition of compact (r,k)-cores in the same running time.*

Proof. We first test if all vertices in $(V' + r) \cap L_D$ are sinks in X. We then add a set V^* of $2k - 1 - |V'|$ new vertices to X. We then test all possibilities to add one arc from $(V' + r) \cap L_D$ to v for every $v \in V_i^*$. As $|V'|, |V^*| \leq 2k$, there are at most $2k^{2k} = 2^{O(k \cdot \log k)}$ possibilities to check. By Lemma 21, we can check in time polynomial in k whether each of the resulting graphs is an (r,k)-flow branching. ◀

The following result, which is similar to Lemma 28, shows that compact (r,k)-cores can be used in a similar way as classic (r,k)-cores.

▶ **Lemma 30.** *Let k be a positive integer and $D = (V + r, A)$ be a 2-root-connected rooted digraph with $|V| \geq 2k - 1$. Then D has two arc-disjoint spanning (r,k)-flow branchings if and only if D contains an extendable pair of arc-disjoint compact (r,k)-cores. Further, given an extendable pair of triple-free arc-disjoint compact (r,k)-cores, we can find a pair of arc-disjoint spanning (r,k)-flow branchings in time $2^{O(k \cdot \log k)} \cdot n^{O(1)}$.*

Proof. By Lemma 24, for the first part it suffices to prove that D contains an extendable pair of arc-disjoint compact (r,k)-cores if and only if D contains an extendable pair of arc-disjoint classic (r,k)-cores.

First let $(X_1 = (V_1 + r, A_1), X_2 = (V_2 + r, A_2))$ be an extendable pair of arc-disjoint classic (r,k)-cores. Let $X_i' = (V_i' + r, A_i')$ be obtained from X_i by first deleting all arcs in A_i whose tail is a large vertex and then restricting to the subdigraph that is root-connected from r. As $A_i' \subseteq A_i$, (X_1', X_2') is extendable. It remains to show that X_1' and X_2' are compact (r,k)-cores. By construction, all vertices in $(V_i' + r) \cap L_D$ are sinks in X_i'. If $X_i' = (r, \emptyset)$, let Y_i be obtained by attaching $2k-1$ outgoing arcs to r. Clearly, Y_i is an (r,k)-flow branching and so X_i' is a compact (r,k)-core. We may hence suppose that $X_i' \neq (r, \emptyset)$. As X_i is an (r,k)-flow branching, there is an (r,k)-branching flow $\mathbf{z} : A_i \to \mathbb{Z}_{\geq 0}$ in X_i. Create Y_i from X_i' by attaching $\mathbf{z}(\delta^-_{X_i'}(v)) - 1$ arcs directed away from v to every vertex $v \in V_i' \cap L_D$. Assigning $\mathbf{z}'(a) = 1$ for all arcs leaving a large vertex in Y_i and $\mathbf{z}'(a) = \mathbf{z}(a)$ for all remaining arcs, we obtain that $\mathbf{z}'$ is an (r,k)-branching flow in Y_i, so Y_i is an (r,k)-flow branching. Furthermore, we have $|V(Y_i) - r| = \mathbf{z}'(\delta^+_{Y_i}(r)) = \mathbf{z}(\delta^+_{X_i}(r)) = |V_i| = 2k-1$. It follows by definition that X_i' is a compact (r,k)-core.

Now let $(X_1 = (V_1 + r, A_1), X_2 = (V_2 + r, A_2))$ be an extendable pair of arc-disjoint compact (r,k)-cores. Possibly deleting arcs, we may suppose by Lemma 23 and as $|V| \geq 2k-1$ that X_1 and X_2 are triple-free. By definition, there are (r,k)-flow branchings Y_1, Y_2 such that Y_i is obtained from X_i by adding a set V_i^* of $2k-1-|V_i|$ new vertices and adding an arc from a vertex in $(V_i + r) \cap L_D$ to v for every $v \in V_i^*$. Let $X_1' = (V_1' + r, A_1')$ and $X_2' = (V_2' + r, A_2')$ be subdigraphs of D that are vertex-maximal with the following properties:

- **(i)** X_i' is obtained from X_i by repeatedly adding another vertex in $v \in V - V_i$ and an arc of A that goes from a vertex in $(V_i + r) \cap L_D$ to v,
- **(ii)** $d^+_{X_i'}(v) \leq d^+_{Y_i}(v)$ for every $v \in (V_i + r) \cap L_D$,
- **(iii)** A_1' and A_2' are disjoint, and
- **(iv)** (X_1', X_2') is extendable.

Note that (X_1, X_2) satisfies conditions **(i)**-**(iv)**, so (X_1', X_2') is well-defined. Further, observe that if condition **(ii)** is satisfied with equality for every $v \in (V_i + r) \cap L_D$, then X_i' is isomorphic to Y_i, so X_i' is an (r,k)-flow branching, thus by definition also a classic (r,k)-core. If this is the case for both X_1' and X_2', we are done by conditions **(iii)** and **(iv)**.

We may therefore suppose by symmetry that there is some $v \in (V_1 + r) \cap L_D$ with $d^+_{X_1'}(v) < d^+_{Y_1}(v)$. For any $a = vz_a$ with $z_a \in N^+_D(v) - V_1'$, let $X_1^a = (V_1' + r + z_a, A_1' \cup a)$. By the maximality of X_1', X_1^a violates one of conditions **(i)**-**(iv)** for every $a \in \delta^+_D(v)$ with $z_a \in N^+_D(v) - V_1'$. By construction and the choice of v, X_1^a satisfies conditions **(i)** and **(ii)** for every $a \in \delta^+_D(v)$ with $z_a \in N^+_D(v) - V_1'$. If X_1^a violates condition **(iii)**, then $z_a \in V_2'$. As $|V_2'| \leq |V(Y_2)| = 2k$, this is the case for at most $2k$ vertices in $N^+_D(v) - V_1'$. If X_1^a does not satisfy **(iv)**, then a is critical in $D - A_1'$. As X_1 is triple-free, $|V_1'| \leq 2k$ and by construction, we obtain that $|A_1'| \leq 4\binom{|V_1'|}{2} \leq 16k^2$. Now Lemma 16 implies that this is the case for at most $16k^2$ vertices in $N^+_D(v) - V_1'$. As $v \in L_D$ and $|V_1'| \leq |V_1 \cup V_1^*| = 2k-1$, we have $|N^+_D(v) - V_1'| \geq (20k^2 + 1) - (2k-1) > 2k + 16k^2$, so there is at least one vertex $z \in N^+_D(v) - V_1'$ and an arc $a = vz$ such that X_1^a does not violate any of conditions **(i)**-**(iv)**, a contradiction.

Observe that the second part of the proof yields an algorithm for computing a pair of arc-disjoint extendable classic (r,k)-cores from a pair of triple-free arc-disjoint extendable compact (r,k)-cores. First observe that by the second part of Proposition 29, we can find Y_1, Y_2 in time $2^{O(k \log k)}$. Now, defining X_2^a similarly to X_1^a, every time we try to add an arc $a = vz$ to X_i' for $i = 1$ or 2, we test if (X_i^a, X_{3-i}') satisfies conditions **(i)**-**(iv)**. Conditions **(i)**-**(iii)** can clearly be checked in polynomial time and, by Lemma 17, condition **(iv)** can also be checked in polynomial time. Never checking an arc a which is parallel to an arc we

have already checked, after at most $20k^2$ failed attempts, we manage to add a new vertex to V_i'. We repeat this procedure at most $|V_i^*| \leq 2k$ times. It follows that a pair of arc-disjoint extendable classic (r,k)-cores can be computed in time $40k^3 \cdot n^{O(1)} = n^{O(1)}$. By the second part of Lemma 24, we can then find the arc-disjoint spanning (r,k)-flow branchings in D in polynomial time. The overall running time of the algorithm is $2^{O(k \cdot \log k)} \cdot n^{O(1)}$, as claimed. ◀

We are now ready to proceed to the proof of Theorem 9.

Proof of Theorem 9. First consider the case where $|V| < 2k-1$. Observe that any arc-minimal spanning (r,k)-flow branching has at most $|V|$ parallel arcs between any two vertices. It follows that, for any two vertices u, v, at most $\gamma := 4k^2$ different distributions of the arcs between u and v among the two candidates for the spanning (r,k)-flow branchings have to be considered, including taking none of these arcs. Since there are $\mu := \binom{|V|}{2} = O(k^2)$ pairs of vertices, the total number of choices for these distributions is $\gamma^\mu = 2^{O(k^2 \cdot \log k)}$. The problem can therefore be solved by a brute force algorithm in time $2^{O(k^2 \cdot \log k)}$, by generating all $2^{O(k^2 \cdot \log k)}$ pairs of candidate subdigraphs of D and checking whether any of these pairs satisfies the required conditions. We may hence suppose that $|V| \geq 2k-1$.

By Lemma 23, we may also suppose that there are at most four parallel arcs between any two vertices in D. By Theorem 1 and the remark after, we can first decide in polynomial time if D is 2-root-connected. If it is not, the answer is negative by Theorem 1 and the fact that every (r,k)-flow branching contains an r-arborescence, so we may suppose it is.

We first explain how to test whether a candidate for the extendable pair of arc-disjoint compact (r,k)-cores is indeed an extendable pair of arc-disjoint compact (r,k)-cores. After, we show that the number of candidates is bounded. Let $X_1 = (V_1 + r, A_1), X_2 = (V_2 + r, A_2)$ be two subdigraphs of D with $|V_1|, |V_2| \leq 2k-1$. By the first part of Proposition 29, we can check in time $2^{O(k \cdot \log k)}$ whether both X_1 and X_2 are compact (r,k)-cores. By Lemma 17, we can therefore decide in time $2^{O(k \cdot \log k)} + n^{O(1)}$ if (X_1, X_2) is an extendable pair of arc-disjoint compact (r,k)-cores in D. By Lemma 30, it therefore suffices to prove that there are at most $2^{O(k^2 \cdot \log k)}$ possible candidates for the extendable pair of arc-disjoint compact (r,k)-cores, and that these can be generated within the same running time.

Let $X = (V' + r, A')$ be a compact (r,k)-core in D. Observe that every vertex in V' can be reached from r by a directed path all of whose internal vertices are in S_D and whose length is at most $2k-1$. As every vertex in S_D has at most $20k^2$ out-neighbors, we obtain that the number of vertices in V that can be reached by such a path is at most $20k^2 + (20k^2)^2 + \ldots + (20k^2)^{2k-1} \leq (20k^2)^{2k}$. As V' contains at most $2k-1$ vertices, there are at most $\binom{(20k^2)^{2k}}{(2k-1)} \leq (20k^2)^{4k^2}$ possibilities to choose V'. Now suppose that we have chosen V' of size at most $2k-1$. As there are at most four arcs in the same direction between any two vertices, there are at most $8\binom{|V'|}{2} \leq 32k^2$ arcs that have their head and tail in $V' + r$. As all arcs of A' have both ends in $V' + r$, there are at most 2^{32k^2} possibilities to choose A'. It follows that there are at most $(20k^2)^{4k^2} \cdot 2^{32k^2}$ possibilities to choose a compact (r,k)-core X. As these can be computed by a brute force method, the algorithm can finish after checking less than $f(k) = \binom{(20k^2)^{4k^2} \cdot 2^{32k^2}}{2} = 2^{O(k^2 \cdot \log k)}$ candidates for the extendable pair of compact (r,k)-cores.

If no such extendable pair of arc-disjoint compact (r,k)-cores exists, by Lemma 30, D does not contain two arc-disjoint spanning (r,k)-flow branchings. On the other hand, if we find an extendable pair of arc-disjoint compact extendable (r,k)-cores, we also find such a pair (X_1, X_2) where X_1 and X_2 are arc-minimal, so by Lemma 23 triple-free. By the second

part of Lemma 30, we can compute the two arc-disjoint spanning (r,k)-flow branchings in polynomial time. The overall running time of the obtained algorithm is $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. ◀

5 An FPT algorithm for packing (r,k)-safe spanning trees

This section is concerned with proving Theorem 12. Again, slightly modifying the terminology introduced in Section 3 and reused in Section 4, given a rooted graph $G = (V + r, E)$ and a positive integer k, we say that a vertex $v \in V + r$ is *large* if $|N_G(v)| \geq 8k - 7$, and *small* otherwise. And again, we let L_G (resp. (S_G)) be the set of vertices in $V + r$ which are large (resp. small) in G.

We are now ready to introduce a new notion of certificates for (r,k)-safe spanning trees. A *compact* (r,k)*-certificate* is a subgraph $X = (V' + r, E')$ of G with $|V'| \leq 2k - 2$ satisfying the following:

- X is a tree,
- all vertices in $(V' + r) \cap L_G$ are terminals of X, and
- an (r,k)-safe tree Y can be obtained from X by adding a set V^* of $2k - 2 - |V'|$ new vertices and adding an edge from a vertex in $(V' + r) \cap L_G$ to v for every $v \in V^*$.

Observe that Y is not necessarily a subgraph of G. We now give the following algorithmic result which plays the analogous role of Proposition 27 and Proposition 29 for k-safe spanning trees.

▶ **Proposition 31.** *Let $G = (V + r, E)$ be a rooted graph, k a positive integer, and $X = (V' + r, E')$ a subgraph of G with $|V'| \leq 2k - 2$. We can test in time $2^{O(k \log k)}$ whether X is a compact (r,k)-certificate. Further, if this is the case, we can construct an (r,k)-safe spanning tree Y with the properties mentioned in the definition of compact (r,k)-certificate in the same running time.*

Proof. We first check whether X is a tree such that all the vertices in $(V' + r) \cap L_G$ are terminals of X. If this is the case, we add a set V^* of $2k - 2 - |V'|$ new vertices to X. We then test all possibilities to add one edge from $(V' + r) \cap L_G$ to v for every $v \in V^*$. As $|V' + r| \leq 2k$ and $|V^*| \leq 2k$, there are at most $(2k)^{(2k)} = 2^{O(k \cdot \log k)}$ possibilities to check. For each of them, we can check in time polynomial in k if the obtained graph is an (r,k)-safe spanning tree. ◀

The following result, which is similar to Lemma 28 and Lemma 30, shows that compact certificates can be used in a similar way as classic certificates.

▶ **Lemma 32.** *Let k be a positive integer and $G = (V + r, E)$ be a rooted graph with $|V| \geq 2k - 2$. Then G has two edge-disjoint (r,k)-safe spanning trees if and only if G contains a completable pair of compact (r,k)-certificates. Further, given a completable pair of compact (r,k)-certificates, we can find a pair of edge-disjoint (r,k)-safe spanning trees in time $2^{O(k \cdot \log k)} \cdot n^{O(1)}$.*

Proof. By Lemma 26, for the first part it suffices to prove that G contains a completable pair of compact (r,k)-certificates if and only if G contains a completable pair of classic (r,k)-certificates.

First let $(X_1 = (V_1 + r, E_1), X_2 = (V_2 + r, E_2))$ be a completable pair of classic (r,k)-certificates. Let $X_i' = (V_i' + r, E_i')$ be obtained from X_i by deleting $C^v_{X_i}$ for every $v \in (V_i + r) \cap L_G$. By construction, the X_i' are trees and all vertices in $(V_i' + r) \cap L_G$ are terminals in X_i'. Let Y_i be obtained from X_i' by adding the vertices in $V_i - V_i'$ and adding an edge from

a vertex $v \in (V_i' + r) \cap L_G$ to a vertex $w \in V_i - V_i'$ whenever $w \in V(C^v_{X_i})$. Observe that Y_i is a tree with $|V(Y_i)| = 2k-1$. Further, we have $|V_i - V(C^v_{Y_i})| = |V_i - V(C^v_{X_i})| \geq k$ for every $v \in V_i'$ and $|V(C^v_{Y_i})| = 0$ for every $v \in V_i - V_i'$. This yields that $|V_i - V(C^v_{Y_i})| \geq |V_i - V(C^v_{X_i})| \geq k$ for every $v \in V_i$ and so Y_i is an (r,k)-safe tree. By definition, we obtain that (X_1', X_2') is a pair of compact (r,k)-certificates. Further, as $E(X_i') \subseteq E(X_i)$ and (X_1, X_2) is completable, we obtain that (X_1', X_2') is completable.

Now let $(X_1 = (V_1 + r, E_1), X_2 = (V_2 + r, E_2))$ be a completable pair of compact (r,k)-certificates. By definition, there are (r,k)-safe trees Y_1, Y_2 such that Y_i is obtained from X_i by adding a set V_i^* of $2k - 2 - |V_i|$ new vertices and an edge from a vertex in $(V_i + r) \cap L_G$ to v for every $v \in V_i^*$. Let $(X_1' = (V_1' + r, E_1'), X_2' = (V_2' + r, E_2'))$ be a pair of subgraphs of G that are vertex-maximal with the following properties:

- **(i)** X_i' is obtained from X_i by repeatedly adding another vertex $v \in V - V_i$ and an edge of E that goes from a vertex in $(V_i + r) \cap L_G$ to v,
- **(ii)** $d_{X_i'}(v) \leq d_{Y_i}(v)$ for every $v \in (V_i + r) \cap L_G$, and
- **(iii)** (X_1', X_2') is completable.

Note that (X_1, X_2) satisfies conditions **(i)**-**(iii)**, so (X_1', X_2') is well-defined. Observe that if condition **(ii)** is satisfied with equality for every $v \in (V_i + r) \cap L_G$, then X_i' is isomorphic to Y_i, so X_i' is an (r,k)-safe tree and by definition also a classic (r,k)-certificate. If this is the case for both X_1' and X_2', we are done by condition **(iii)**.

We may therefore suppose by symmetry that there is a vertex $v \in (V_1 + r) \cap L_G$ with $d_{X_1'}(v) < d_{Y_1}(v)$. For any $e = vz_e \in \delta_G(v)$ with $z_e \in N_G(v) - V_1'$, let $X_1^e = (V_1' + r + z_e, E_1' \cup e)$. By the maximality of X_1', we obtain that X_1^e violates one of conditions **(i)**-**(iii)** for every $e = vz_e \in \delta_G(v)$ with $z_e \in N_G(v) - V_1'$. By construction and the choice of v, X_1^e satisfies conditions **(i)** and **(ii)** for every $e = vz_e \in \delta_G(v)$ with $z_e \in N_G(v) - V_1'$. It follows that (X_1^e, X_2') violates condition **(iii)** for every $e = vz_e \in \delta_G(v)$ with $z_e \in N_G(v) - V_1'$. As (X_1', X_2') is completable, there are two edge-disjoint spanning trees T_1, T_2 of G such that $E_i' \subseteq E(T_i)$ for $i = 1, 2$.

▶ **Claim 33.** *There is no $e = vz_e \in \delta_G(v) - (E(T_1) \cup E(T_2))$ with $z_e \in N_G(v) - V_1'$ such that (X_1^e, X_2') violates condition* **(iii)**.

Proof. Suppose otherwise. By Lemma 18, there is an edge $f \in E(T_1)$ incident to z_e such that $T_1' = T_1 - f + e$ is a spanning tree of G. As $z_e \notin V_1'$, we obtain that $f \notin E_1'$, yielding $E(X_1^e) \subseteq E(T_1')$. As T_1' and T_2 are edge-disjoint, we obtain that (X_1^e, X_2') is completable, a contradiction. ◀

▶ **Claim 34.** *There are at most $6k - 6$ vertices $z \in N_G(v) - V_1'$ such that (X_1^e, X_2') violates condition* **(iii)** *for some $e = vz \in E(T_2)$.*

Proof. Suppose otherwise. As $|V_2'| \leq |V(Y_2)| - 1 = 2k - 2$, we obtain that there are at least $4k - 3$ vertices $z \in N_G(v) - V_1'$ such that (X_1^e, X_2') violates condition **(iii)** for some $e = vz \in E(T_2) - E_2'$. Let $\sigma : E(T_2) \to E(T_1)$ be a tree-mapping function from T_2 to T_1. By Lemma 19 and since $|E_1'| \leq 2k - 2$, there is some $z \in N_G(v) - V_1'$ and an edge $e = vz \in E(T_2) - E_2'$ such that $\sigma(e) \in E(T_1) - E_1'$. By definition of tree-mapping functions, $T_1' = T_1 - \sigma(e) + e$ and $T_2' = T_2 - e + \sigma(e)$ are edge-disjoint spanning trees of G. As $E(X_1^e) \subseteq E(T_1')$ and $E(X_2') \subseteq E(T_2')$, we obtain that (X_1^e, X_2') is completable, a contradiction. ◀

As $v \in L_G$ and $|V_1'| \leq |V_1 \cup V_1^*| = 2k-2$, we have $|N_G(v) - V_1'| \geq (8k-7) - (2k-2) > 6k-6$. It now follows from Claim 33 and Claim 34 that there is at least one vertex in $z \in N_G(v) - V_1'$ and an edge $e = vz$ such that X_1^e does not violate any of conditions **(i)**-**(iii)**, a contradiction.

Observe that the second part of the proof yields an algorithm for computing a completable pair of classic (r,k)-certificates from a pair of completable compact (r,k)-certificates. First observe that by the second part of Proposition 29, we can find Y_1, Y_2 in time $2^{O(k \log k)}$. Now, defining X_2^a similarly to X_1^a, every time we try to add an edge e to X_i', for $i = 1$ or 2, we test if (X_i^e, X_{3-i}') satisfies conditions **(i)**-**(iii)**. Conditions **(i)**-**(ii)** can clearly be checked in polynomial time and, by Lemma 25, condition **(iii)** can also be checked in polynomial time. Never checking an edge that is parallel to one we have already checked, after at most $6k-6$ failed attempts, we manage to add a new vertex to V_i'. We repeat this procedure $|V_i^*| \leq 2k-2$ times. It follows that a completable pair of classic (r,k)-certificates can be computed in time $k^2 \cdot n^{O(1)} = n^{O(1)}$. By the second part of Lemma 26, we can then find two edge-disjoint (r,k)-safe spanning trees in G in polynomial time. The overall running time of the algorithm is $2^{O(k \log k)} \cdot n^{O(1)}$, as claimed. ◀

We are now ready to proceed to the proof of Theorem 12.

Proof of Theorem 12. We may suppose that there are at most two parallel edges from u to v for any $u, v \in V + r$. If $|V| < 2k-2$, the problem can be solved by a brute force algorithm in time $2^{O(k^2)}$, by generating all pairs of subgraphs of G and checking whether any of these pairs satisfies the required conditions. We may hence also suppose that $|V| \geq 2k-2$.

We first explain how to test whether a candidate for the completable pair of compact (r,k)-certificates is indeed a completable pair of compact (r,k)-certificates. After, we show that the number of candidates is bounded.

Let $X_1 = (V_1 + r, E_1), X_2 = (V_2 + r, E_2)$ be two subgraphs of G. By the first part of Proposition 31, we can check in time $2^{O(k \cdot \log k)}$ whether both X_1 and X_2 are compact (r,k)-certificates. By Lemma 25, we can therefore decide in time $2^{O(k \cdot \log k)} + n^{O(1)}$ if (X_1, X_2) is a completable pair of compact (r,k)-certificates in G. By Lemma 32, it therefore suffices to prove that there are at most $2^{O(k^2 \cdot \log k)}$ possible candidates for the completable pair of compact (r,k)-certificates, and that they can be generated within the same running time.

Let $X = (V' + r, E')$ be a compact (r,k)-certificate in G. Observe that every vertex in V' can be reached from r by a path all of whose interior vertices are in S_G and whose length is at most $k-1$. As every vertex in S_G has at most $8k-8$ neighbors, we obtain that the number of vertices in V that can be reached by such a path is at most $(8k-8) + (8k-8)^2 + \ldots + (8k-8)^{k-1} \leq (8k)^k$. As V' contains at most $2k-2$ vertices, there are at most $\binom{(8k)^k}{2k-2} \leq (8k)^{2k^2}$ possibilities to choose V'.

Now suppose that we have chosen V' of size $2k-2$. Observe that there are at most $2\binom{|V'+r|}{2} \leq 8k^2$ edges that have both ends in $V' + r$. As $|E'| = 2k-2$, there are at most $\binom{8k^2}{2k-2} \leq (8k^2)^{2k}$ possibilities to choose A'. It follows that there are at most $(8k)^{2k^2} \cdot (8k^2)^{2k}$ possibilities to choose a compact (r,k)-certificate X. As these can be computed by a brute force method, the algorithm can finish after checking less than $f(k) = \binom{(8k)^{2k^2} \cdot (8k^2)^{2k}}{2} = 2^{O(k^2 \cdot \log k)}$ candidates for the pair of compact (r,k)-certificates.

If no completable pair of compact (r,k)-certificates exists, by Lemma 26, G does not contain two edge-disjoint (r,k)-safe spanning trees. On the other hand, once we have found a pair of completable compact (r,k)-certificates, we can compute in polynomial time the two edge-disjoint (r,k)-safe spanning trees by the second part of Lemma 26. The overall running time of the obtained algorithm is $2^{O(k^2 \cdot \log k)} \cdot n^{O(1)}$. ◀

6 A hardness result for packing (r, k)-safe spanning trees

In this section we prove Theorem 13. It is well-known that the 3-Sat problem is NP-complete. Further, we will need the following lemma derived from the ETH using the so-called Sparsification Lemma [10].

▶ **Lemma 35** (Impagliazzo et al. [10]). *Assuming the* ETH*, there is an $\varepsilon > 0$ such that there is no algorithm for solving a* 3-Sat *formula with ℓ variables and m clauses in time $2^{\varepsilon m} \cdot (\ell + m)^{O(1)}$.*

The proof of Theorem 13 given below is strongly inspired from the reduction given in [3, Theorem 5.2], but we provide it here entirely for the sake of completeness.

Proof of Theorem 13. Observe that, given a rooted graph $G = (V + r, E)$ and two positive integers p and k, G contains an (r, k)-safe spanning tree if and only if the graph that is obtained from G by replacing each of its edges by p parallel copies of itself contains p edge-disjoint (r, k)-safe spanning trees. Hence, it suffices to prove the statement for $p = 1$. Let ϕ be an instance of 3-Sat, with variables $x_1, x_2, \ldots, x_\ell$ and clauses $C_1, C_2, \ldots, C_m$. Adding a variable that is not contained in any clause if necessary, we can assume that ℓ is even. We construct a simple rooted graph $G = (V + r, E)$ as follows; see Figure 1 for an illustration. For $i = 1, \ldots, \ell$, let V_i be an independent set containing two vertices v_i and $\bar{v_i}$, and let r and t be two extra vertices. Add all possible edges between r and V_1, between V_i and V_{i+1} for $i = 1, \ldots, \ell - 1$, and between V_ℓ and t. Next, add m vertices $c_1, \ldots, c_m$ and link c_i to v_j (resp. $\bar{v_j}$) with a path containing $\ell/2$ interior vertices if x_i (resp. $\bar{x_i}$) is a literal of C_i. Finally, let $k := 1 + \ell + 3\ell m/2 + m$ and add to G an independent set Q on q vertices all linked to t, where $q > k - \ell - 1$ will be specified later. Notice that we have $n = |V| = q + k + \ell + 1$.

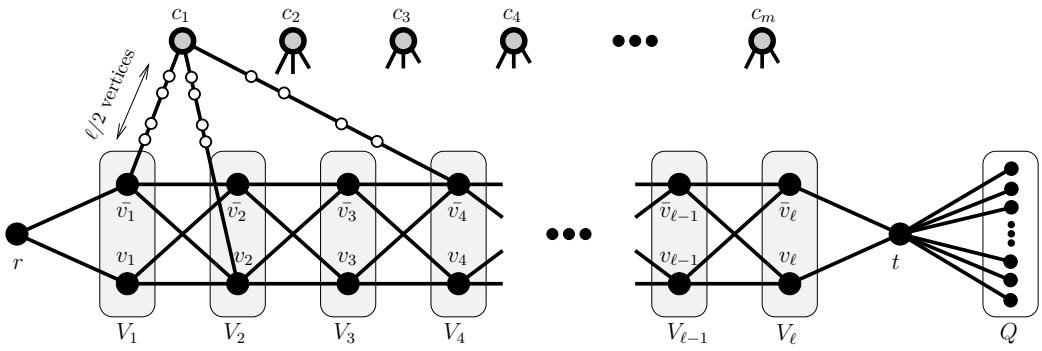

Figure 1 The rooted graph G in the proof of Theorem 13 with $C_1 = \bar{x_1} \vee x_2 \vee \bar{x_4}$.

We now prove that ϕ is satisfiable if and only if G admits an (r, k)-safe spanning tree. First assume that G contains an (r, k)-safe spanning tree T. Observe that, by construction and the definition of (r, k)-safe spanning trees, $T - r$ has exactly two components whose vertex sets we denote by T_1 and T_2. We may assume that T_1 contains t. Then $G[T_1]$ contains a path P from V_1 to t and T_1 contains Q. As a shortest path from V_1 to t contains $\ell + 1$ vertices, we have $|T_1| \geq q + \ell + 1$ and $n - |T_1| \leq k$. As T is (r, k)-safe, we obtain that $|T_1| = q + \ell + 1$ and T_1 consists exactly of the vertex set of the path P, which contains exactly $\ell + 1$ vertices, and the whole set Q. In particular, P intersects each V_i in exactly one vertex. Now for $i = 1, \ldots, \ell$, we set x_i to true if P contains $\bar{v_i}$ and to false if P contains v_i. For each clause C_j, there must be a path from the corresponding vertex c_j to V_1 in T_2. Then one of

the vertices corresponding to a literal of C_j must not be contained in P and so, this literal is set to true and C_j is satisfied by it. It follows that the constructed assignment satisfies ϕ.

Conversely, assume that ϕ admits a truth assignment. Let P be the path defined so that, for every $i = 1, \ldots, \ell$, it contains v_i if x_i is set to false and $\bar{v}_i$ if x_i is set to true. Further, let $T_1 = V(P) \cup t \cup Q$. As ϕ is satisfied, for every $1 \leq j \leq m$, there exists a path from c_j to $V_1 \cup \cdots \cup V_\ell$ in $G - (T_1 \cup r)$. It follows that $G - (T_1 \cup r)$ is connected and we select a spanning tree of it. The union of this spanning tree with $G[T_1]$, r, and the edges incident to r is a spanning tree of G. In order to see that this spanning tree is (r, k)-safe, it suffices to observe that, as $q > k - \ell - 1$, we have $|T_1| = q + \ell + 1 > k$ and $|V - T_1| = q + k + 1 + \ell - (q + \ell + 1) = k$.

If we fix $q = k$, then the size of G is bounded by a polynomial in ℓ and m. Thus, the above reduction implies that, as 3-Sat is NP-complete, given a rooted graph G and an integer k, deciding whether G admits an (r, k)-safe spanning tree is NP-complete.

Let now ε be a positive constant and assume that k is an integer function satisfying $(\log(n))^{2+\varepsilon} \leq k(n) \leq \frac{n}{2}$ for every $n > 0$. Furthermore, suppose that there exists a constant C^* such that for every integer $c \geq C^*$ there exists an n such that $k(n) = c$. Finally, for the sake of a contradiction assume that there exists a polynomial-time algorithm $\mathbf{A}$, running in time $O(n^{c_0})$ for some $c_0 > 0$, for deciding if a given rooted graph $G = (V + r, E)$ on n vertices contains an $(r, k(n))$-safe spanning tree. Then let ϕ be a 3-Sat formula with ℓ variables and m clauses. We may assume that ℓ and m are large enough so that $1 + \ell + 3\ell m/2 + m \geq C^*$. Adding trivial clauses if necessary, we may also assume that $\ell \leq m$. By assumption, there exists n such that $k(n) = 1 + \ell + 3\ell m/2 + m$. So, in the above reduction, we choose q to be $n - (k(n) + \ell + 1)$, in order to have $n = q + k(n) + \ell + 1$. Then, using algorithm $\mathbf{A}$, one could decide if ϕ is satisfiable in time $O(n^{c_0}) = O(2^{c_0 \cdot \log n}) = O(2^{c_0 \cdot k(n)^{1/(2+\varepsilon)}})$, where we have used the assumption that $k(n) \geq (\log(n))^{2+\varepsilon}$. Moreover, in the previous construction, we have $k(n) = 1 + \ell + 3\ell m/2 + m \leq 3m + 3m^2/2 \leq 3m^2$. So we could decide whether ϕ is satisfiable in time $O(2^{c_0 \cdot (3m^2)^{1/(2+\varepsilon)}}) = O(2^{c_0' \cdot m^{\varepsilon'}})$ with $\varepsilon' = 2/(2+\varepsilon) < 1$, a contradiction to Lemma 35 assuming the ETH. ◀

7 Conclusion

We considered three problems on finding certain disjoint substructures in graphs and digraphs. While in our proofs we restrict to finding two of these substructures for the sake of simplicity, our results can be generalized to allow for finding an arbitrary number of them using the same proof techniques. More concretely, the following results can be established using the techniques of this article. As in [3], we omit the proofs of these generalized statements.

▶ **Theorem 36.** *Given a rooted digraph $D = (V + r, A)$ and an integer $p \geq 2$, deciding whether D contains p arc-disjoint k-safe spanning r-arborescences is* FPT *with parameter k. More precisely, the problem can be solved in time $2^{O(p \cdot k^2 \cdot \log k)} \cdot n^c$, where c is a constant depending on p. Further, if they exist, the p arc-disjoint k-safe spanning r-arborescences can be computed within the same running time.*

▶ **Theorem 37.** *Given a rooted digraph $D = (V + r, A)$ and an integer $p \geq 2$, deciding whether D contains p arc-disjoint (r, k)-flow branchings is* FPT *with parameter k. More precisely, the problem can be solved in time $2^{O(p \cdot k^2 \cdot \log k)} \cdot n^c$, where c is a constant depending on p. Further, if they exist, the p arc-disjoint (r, k)-flow branchings can be computed within the same running time.*

▶ **Theorem 38.** *Given a rooted graph $G = (V + r, E)$ and an integer $p \geq 2$, deciding whether G contains p arc-disjoint (r, k)-safe spanning trees is* FPT *with parameter k. More precisely,*

the problem can be solved in time $2^{O(p \cdot k^2 \cdot \log k)} \cdot n^c$*, where* c *is a constant depending on* p*. Further, if they exist, the* p *edge-disjoint* (r,k)*-safe spanning trees can be computed within the same running time.*

It is natural to ask whether the dependency on k of our FPT algorithms can be improved.

There is still a significant gap between the lower bounds in Corollaries 3,7 and 14, which are $2^{O(k^{1-\varepsilon})}$ or $2^{O(k^{1/2-\varepsilon})}$, and the function $2^{O(k^2 \cdot \log k)}$ in our FPT algorithms.

We did not focus on optimizing the polynomial factors in n of our algorithms, and we leave it for further research. Further, we leave as an open question whether any of the considered problems admit a polynomial kernel parameterized by k. Finally, it would be interesting to find a theorem on packing k-safe mixed arborescences in mixed graphs, hence generalizing both Theorem 36 and Theorem 38.

References

1 Jørgen Bang-Jensen and Stéphane Bessy. (arc-)disjoint flows in networks. *Theoretical Computer Science*, 526:28–40, 2014. doi:10.1016/j.tcs.2014.01.011.

2 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. London, 2nd edition, 2009. URL: https://dblp.org/rec/books/daglib/0022205.bib.

3 Jørgen Bang-Jensen, Frédéric Havet, and Anders Yeo. The complexity of finding arc-disjoint branching flows. *Discrete Applied Mathematics*, 209:16–26, 2016. doi:10.1016/j.dam.2015.10.012.

4 Jørgen Bang-Jensen and Anders Yeo. Balanced branchings in digraphs. *Theoretical Computer Science*, 595:107–119, 2015. doi:10.1016/j.tcs.2015.06.026.

5 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

6 Jack Edmonds. Edge-disjoint branchings. In B. Rustin, editor, *Combinatorial Algorithms*, pages 91–96, New York, 1973. Academic Press.

7 Jack Edmonds. Some well-solved problems in combinatorial optimization. In B. Roy, editor, *Combinatorial Programming: Methods and Applications (Proceedings of the NATO Advanced Study Institute, Versailles, 1974)*, Reidel, Dordrecht, 1975. Academic Press. doi:10.1007/978-94-011-7557-9_15.

8 András Frank. *Connections in Combinatorial Optimization*. 2011. URL: https://global.oup.com/academic/product/connections-in-combinatorial-optimization-9780199205271.

9 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

10 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

11 László Lovász. Connectivity in digraphs. *Journal of Combinatorial Theory, Series B*, 15(2):174–177, 1973. doi:10.1016/0095-8956(73)90018-X.

12 William T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, 36(1):221–230, 1961. doi:10.1112/jlms/s1-36.1.221.