Constant Approximating Disjoint Paths on Acyclic Digraphs Is W[1]-Hard

Michał Włodarczyk
University of Warsaw, Poland

Abstract

In the Disjoint Paths problem, one is given a graph with a set of k vertex pairs (s_i, t_i) and the task is to connect each s_i to t_i with a path, so that the k paths are pairwise disjoint. In the optimization variant, Max Disjoint Paths, the goal is to maximize the number of vertex pairs to be connected. We study this problem on acyclic directed graphs, where Disjoint Paths is known to be W[1]-hard when parameterized by k. We show that in this setting Max Disjoint Paths is W[1]-hard to c-approximate for any constant c. To the best of our knowledge, this is the first non-trivial result regarding the parameterized approximation for Max Disjoint Paths with respect to the natural parameter k. Our proof is based on an elementary self-reduction that is guided by a certain combinatorial object constructed by the probabilistic method.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability

Keywords and phrases fixed-parameter tractability, hardness of approximation, disjoint paths

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.57

Funding *Michał Włodarczyk*: Supported by Polish National Science Centre SONATA-19 grant number 2023/51/D/ST6/00155.

1 Introduction

The Disjoint Paths problem has attracted a lot of attention both from the perspective of graph theory and applications [23, 46, 50, 52]. Both decision variants, where one requires the paths to be either vertex-disjoint or edge-disjoint, are known to be NP-hard already on very simple graph classes [27, 37, 44, 45]. This has motivated the study of Disjoint Paths through the lens of parameterized complexity. Here, the aim is to develop algorithms with a running time of the form $f(k) \cdot n^{\mathcal{O}(1)}$, where f is some computable function of a parameter k and n is the input size. A problem admitting such an algorithm is called *fixed-parameter tractable* (FPT). In our setting, k is the number of vertex pairs to be connected. On undirected graphs, both variants of Disjoint Paths have been classified as FPT thanks to the famous Graph Minors project by Robertson and Seymour [49] (see [32, 36] for later improvements). This was followed by a line of research devoted to designing faster FPT algorithms on planar graphs [1, 13, 41, 48, 54].

On directed graphs, there is a simple polynomial transformation between the vertex-disjoint and the edge-disjoint variants, so these two problems turn out equivalent. Here, the problem becomes significantly harder: It is already NP-hard for $k = 2$ [22]. The situation is slightly better for acyclic digraphs (DAGs) where Disjoint Paths can be solved in time $n^{\mathcal{O}(k)}$ [22] but it is W[1]-hard [51] (cf. [2]) hence unlikely to be FPT. In addition, no $n^{o(k)}$-time algorithm exists under the assumption of the Exponential Time Hypothesis (ETH) [12]. Very recently, it has been announced that Disjoint Paths is FPT on Eulerian digraphs [5]. It is also noteworthy that the vertex-disjoint and edge-disjoint variants are not equivalent on planar digraphs as the aforementioned reduction does not preserve planarity. Indeed, here the vertex-disjoint version is FPT [17] whereas the edge-disjoint version is W[1]-hard [12].

In the optimization variant, called Max Disjoint Paths, we want to maximize the number of terminals pairs connected by disjoint paths. The approximation status of this problem has been studied on various graph classes [8, 10, 15, 14, 20, 34, 35]. On acyclic digraphs the best approximation factor is $\mathcal{O}(\sqrt{n})$ [9] and this cannot be improved unless P=NP [7]. A different relaxation is to allow the algorithm to output a solution in which every vertex appears in at most c paths (or to conclude that there is no vertex-disjoint solution). Kawarabayashi, Kobayashi, and Kreutze [31] used the directed half-integral grid theorem to design a polynomial-time algorithm for directed Disjoint Paths with congestion $c = 4$ for every k. In other words, such a relaxed problem belongs to the class XP. Subsequently, the congestion factor has been improved to $c = 3$ [33] and $c = 2$ [24].

Hardness of FPT approximation. For problems that are hard from the perspective of both approximation and FPT algorithms, it is natural to exploit the combined power of both paradigms and consider FPT approximation algorithms. Some prominent examples are an FPT approximation scheme for k-Cut [43] and an FPT 2-approximation for Directed Odd Cycle Transversal [42] parameterized by the solution size k. However, several important problems proved to be resistant to FPT approximation as well. The first hardness results in this paradigm have been obtained under a relatively strong hypothesis, called Gap-ETH [6]. Subsequently, an $\mathcal{O}(1)$-approximation for k-Clique was shown to be W[1]-hard [39] and later the hardness bar was raised to $k^{o(1)}$ [29]. In turn, k-Dominating Set is W[1]-hard to $f(k)$-approximate for any function f [30] and W[2]-hard to $\mathcal{O}(1)$-approximate [40]. More results are discussed in the survey [21].

Proving approximation hardness under Gap-ETH is easier compared to the assumption FPT≠W[1] because Gap-ETH already assumes hardness of a problem with a *gap*. Indeed, relying just on FPT≠W[1] requires the reduction to perform some kind of *gap amplification*, alike in the PCP theorem [19]. Very recently, the so-called *Parameterized Inapproximability Hypothesis* (PIH) has been proven to follow from ETH [25]. This means that ETH implies FPT approximation hardness of Max 2-CSP parameterized by the number of variables within some constant approximation factor $c > 1$, which has been previously used as a starting point for parameterized reductions [4, 26, 42, 47]. It remains open whether PIH can be derived from the weaker assumption FPT≠W[1].

Lampis and Vasilakis [38] showed that undirected Max Vertex-Disjoint Paths admits an FPT approximation scheme when parameterized by *treedepth* but, assuming PIH, this is not possible under parameterization by *pathwdith*. See [11, 20] for more results on approximation for Max Disjoint Paths under structural parameterizations. Bentert, Fomin, and Golovach [3] considered the Max Vertex-Disjoint Shortest Paths problem where we additionaly require each path in a solution to be a shortest path between its endpoints. They ruled out FPT(k) approximation with factor $k^{o(1)}$ for this problem assuming FPT≠W[1] and with factor $o(k)$ assuming Gap-ETH.

Our contribution. We extend the result by Slivkins [51] by showing that Max Disjoint Paths on acyclic digraphs does not admit an FPT algorithm that is a q-approximation, for any constant q. We formulate our hardness result as W[1]-hardness of the task of distinguishing between instances that are fully solvable from those in which less than a $\frac{1}{q}$-fraction of the requests can be served at once. Since a q-approximation algorithm could be used to tell these two scenarios apart, the following result implies hardness of approximation. We refer to a pair (s_i, t_i) as a *request* that should be *served* by a path connecting s_i to t_i.

▶ **Theorem 1.** *Let $q \in \mathbb{N}$ be a constant. It is W[1]-hard to distinguish whether for a given instance of k-Dag Disjoint Paths:*
1. *all the requests can be served simultaneously, or*
2. *no set of k/q requests can be served simultaneously.*

Our proof is elementary and does not rely on coding theory or communication complexity as some previous W[1]-hardness of approximation proofs [30, 39]. Instead, we give a gap-amplifying self-reduction that is guided by a certain combinatorial object constructed via the probabilistic method.

Techniques. A similar parameterized gap amplification technique has been previously applied to the k-Steiner Orientation problem: given a graph G with both directed and undirected edges, together with a set of vertex pairs $(s_1, t_1), \ldots, (s_k, t_k)$, we want to orient all the undirected edges in G to maximize the number of pairs (s_i, t_i) for which t_i is reachable from s_i. The problem is W[1]-hard and the gap amplification technique can be used to establish W[1]-hardness of constant approximation [53]. The idea is to create multiple copies of the original instance and connect them sequentially into many layers, in such a way that the fraction of satisfiable requests decreases as the number of layers grows. What distinguishes k-Steiner Orientation from our setting though is that therein we do not require the (s_i, t_i)-paths to be disjoint. So it is allowed to make multiple copies of each request (s_i, t_i) and connect the t_i-vertices to the s_i-vertices in the next layer in one-to-many fashion. Such a construction obviously cannot work for Dag Disjoint Paths. Instead, will we construct a combinatorial object yielding a scheme of connections between the copies of the original instance, with just one-to-one relation between the terminals from the consecutive layers.

Imagine a following construction: given an instance I of k-Dag Disjoint Paths we create $2k$ copies of I: $I^1_1, \ldots, I^1_k$ and $I^2_1, \ldots, I^2_k$. Next, for each $i \in [k]$ we choose some permutation $\pi_i \colon [k] \to [k]$ and for each $j \in [k]$ we connect the sink t_j in I^1_i to the source s_i in $I^2_{\pi_i(j)}$. See Figure 1 on page 6. Then for each $(i, j) \in [k]^2$ we request a path from the source s_j in I^1_i to the sink t_i in $I^2_{\pi_i(j)}$. Observe that if I is a yes-instance then we can still serve all the requests in the new instance. **However, when I is a no-instance, then there is a family $\mathcal{F}$ of $2k$ many k-tuples from $[k]^2$ so that each tuple represents k requests that cannot be served simultaneously.** Each tuple corresponds to some k requests that have to be routed through a single copy of I, which is impossible when I is a no-instance.

We can now iterate this argument. In the next step we repeat this construction k times (but possibly with different permutations), place such k instances next to each other, and create the third layer comprising now k^2 copies of I. Then for each $i \in [k]$ we need a permutation $\pi_i \colon [k^2] \to [k^2]$ describing the connections between the sinks from the second layer to the sources from the third layer. Again, if I is a no-instance, we obtain a family $\mathcal{F}$ of $3k^2$ many k-tuples from $[k]^3$ corresponding to subsets of requests that cannot be served simultaneously. We want to show that after $d = f(k)$ many iterations no subset A of 50% requests can be served. In other words, the family $\mathcal{F}$ should always contain a tuple contained in A, certifying that A is not realizable. This will give a reduction from the exact version of Dag Disjoint Paths to a version of Dag Disjoint Paths with gap $\frac{1}{2}$. The crux of the proof is to find a collection of permutations that will guarantee the desired property of $\mathcal{F}$.

It is convenient to think about this construction as a game in which the first player chooses the permutations governing the connections between the layers (thus creating an instance of Dag Disjoint Paths) and the second player picks a subset A of 50% requests. The first player wins whenever the family $\mathcal{F}$ of forbidden k-tuples includes a tuple contained in A. We need to show that the first player has a single winning strategy against every possible strategy of the second player. We will prove that a good strategy for the first player is to choose every permutation independently and uniformly at random. In fact, for a sufficiently large d and any fixed strategy A of the second player, the probability that A wins against a randomized strategy is smaller than 2^{-k^d}. Since the number of possible strategies for the second player is at most 2^{k^d} (because there are k^d requests), **the union bound implies that the first player has a positive probability of choosing a strategy that guarantees a victory against every strategy of the second player.** This translates to the existence of a family of permutations for which the gap amplification works.

2 Preliminaries

We follow the convention $[n] = \{1, 2, \ldots, n\}$ and use the standard graph theoretic terminology from Diestel's book [18]. We begin by formalizing the problem.

Max Disjoint Paths **Parameter:** k
Input: A digraph D, a set $\mathcal{T}$ of k pairs $(s_i, t_i) \in V(D)^2$.
Task: Find a largest collection $\mathcal{P}$ of vertex-disjoint paths so that each path $P \in \mathcal{P}$ is an (s_i, t_i)-path for some $(s_i, t_i) \in \mathcal{T}$.

We refer to the pairs from $\mathcal{T}$ as *requests*. A solution $\mathcal{P}$ is said to *serve* request (s_i, t_i) if it contains an (s_i, t_i)-path. The condition of vertex-disjointedness implies that each request can be served by at most one path in $\mathcal{P}$. A *yes-instance* is an instance admitting a solution serving all the k-requests. Otherwise we deal with a *no-instance*. (Max) Dag Disjoint Paths is a variant of (Max) Disjoint Paths where the input digraph is assumed to be acyclic.

Notation for trees. For a rooted tree T and $v \in V(T)$ we denote by $\mathsf{Children}(v)$ the set of direct descendants of v. A vertex v in a rooted tree is a leaf if $\mathsf{Children}(v) = \emptyset$. We refer to the set of leaves of T as $L(T)$. The depth of a vertex $v \in V(T)$ is defined as its distance from the root, measured by the number of edges. In particular, the depth of the root equals 0. The set of vertices of depth i in T is called the i-th layer of T.

For $v \in V(T)$ we write T^v to denote the subtree of T rooted at v. We can additionally specify an integer $\ell \geq 1$ and write $T^{v,\ell}$ for the tree comprising the first ℓ layers of T^v. In particular, the tree $T^{v,1}$ contains only the vertex v.

For $k, d \in \mathbb{N}$ we denote by $T_{k,d}$ the full k-ary rooted tree of depth d. We have $|L(T_{k,d})| = k^d$. A subset $A \subseteq L(T_{k,d})$ is called a q-subset for $q \in \mathbb{N}$ if $|A| \geq |L(T_{k,d})| \,/\, q$.

Fixed parameter tractability. We provide only the necessary definitions here; more information can be found in the book [16]. A parameterized problem can be formalized as a subset of $\Sigma^* \times \mathbb{N}$. We say that a problem is *fixed parameter tractable* (FPT) if it admits an algorithm solving an instance (I, k) in running time $f(k) \cdot |I|^{\mathcal{O}(1)}$, where f is some computable function.

To argue that a parameterized problem is unlikely to be FPT, we employ FPT-reductions that run in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ and transform an instance (I, k) into an equivalent one (I', k') where $k' = g(k)$. A canonical parameterized problem that is believed to lie outside the class FPT is k-Clique. The problems that are FPT-reducible to k-Clique form the class W[1].

Negative association. We introduce the following concept necessary for our probabilistic argument. There are several definitions capturing negative dependence between random variables; intuitively it means that when one variable takes a high value then a second one is more likely to take a low value. Negative association formalizes this idea in a strong sense.

▶ **Definition 2.** *A collection of random variables $X_1, X_2, \ldots, X_n \in \mathbb{R}$ is said to be* negatively associated *if for every pair of disjoint subsets $A_1, A_2 \subseteq [n]$ and every pair of increasing functions $f_1 \colon \mathbb{R}^{|A_1|} \to \mathbb{R}$, $f_2 \colon \mathbb{R}^{|A_2|} \to \mathbb{R}$ it holds that*

$$\mathbb{E}\left[f_1(X_i \mid i \in A_1) \cdot f_2(X_i \mid i \in A_2)\right] \leq \mathbb{E}\left[f_1(X_i \mid i \in A_1)\right] \cdot \mathbb{E}\left[f_2(X_i \mid i \in A_2)\right].$$

We make note of several important properties of negative association.

▶ **Lemma 3** ([28, Prop. 3, 6, 7])**.** *Consider a collection of random variables $X_1, X_2, \ldots, X_n \in \mathbb{R}$ that is negatively associated. Then the following properties hold.*

1. *For every family of disjoint subsets $A_1, \ldots, A_k \subseteq [n]$ and increasing functions $f_1, \ldots, f_k$, $f_i \colon \mathbb{R}^{|A_i|} \to \mathbb{R}$, the collection of random variables*

$$f_1(X_i \mid i \in A_1),\ f_2(X_i \mid i \in A_2),\ \ldots,\ f_k(X_i \mid i \in A_k)$$

   is negatively associated.
2. *If random variables $Y_1, \ldots, Y_n$ are negatively associated and independent from $X_1, \ldots, X_n$ then the collection $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ is negatively associated.*
3. *For every sequence $(x_1, x_2, \ldots, x_n)$ of real numbers we have*

$$\mathbb{P}\left[X_i \leq x_i \mid i \in [n]\right] \leq \prod_{i=1}^{n} \mathbb{P}\left[X_i \leq x_i\right].$$

▶ **Lemma 4.** *Let $n, k \in \mathbb{N}$. For $i \in [k]$ let $\mathcal{X}^i = (X^i_1, \ldots, X^i_n)$ be a sequence of real random variables that are negatively associated. Suppose that $\mathcal{X}^1, \ldots, \mathcal{X}^k$ are independent from each other. Then the random variables $(\sum_{i=1}^k X^i_1, \ldots, \sum_{i=1}^k X^i_n)$ are negatively associated.*

Proof. By Lemma 3(2) we know that the union $\mathcal{X}^1 \cup \cdots \cup \mathcal{X}^k$ forms a collection of nk random variables that are negatively associated. We divide it into n disjoint subsets of the form $(\{X^1_j, \ldots, X^k_j\})_{j=1}^n$ and apply Lemma 3(1) for the increasing function $f \colon \mathbb{R}^k \to \mathbb{R}$, $f(x_1, \ldots, x_k) = \sum_{i=1}^k x_i$. ◀

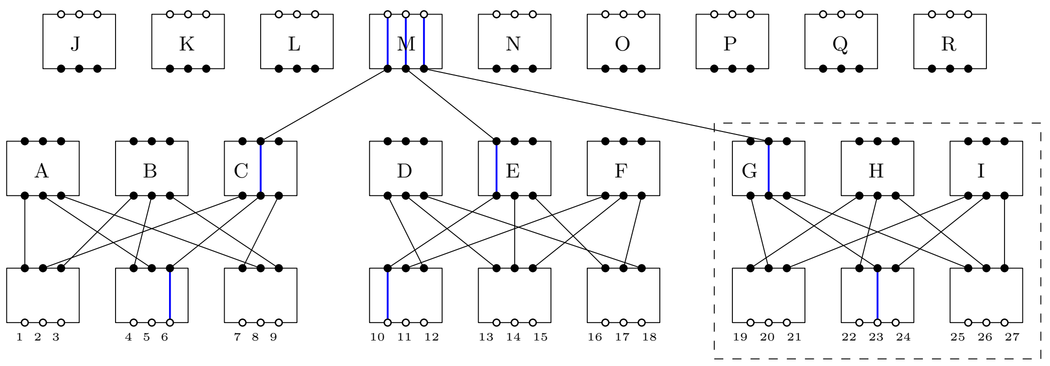
Negative association occurs naturally in situations like random sampling without replacement. A scenario important for us is when an ordered sequence of numbers is being randomly permuted. Intuitively, observing a high value at some index removes this value from the pool and decreases the chances of seeing high values at the remaining indices.

▶ **Theorem 5** ([28, Thm. 2.11])**.** *Consider a sequence $(x_1, x_2, \ldots, x_n)$ of real numbers. Let $\Pi \colon [n] \to [n]$ be a random variable representing a permutation of the set $[n]$ chosen uniformly at random. For $i \in [n]$ we define a random variable $X_i = x_{\Pi^{-1}(i)}$. Then the random variables $X_1, X_2, \ldots, X_n$ are negatively associated.*

3 The reduction

Our main objects of interest are collections of functions associated with the nodes of the full k-ary rooted tree. Such a function for a node v gives an ordering of leaves in the subtree of v.

▶ **Definition 6.** *A* scheme *for $T_{k,d}$ is a collection of functions, one for each node in $T_{k,d}$, such that the function f_v associated with $v \in V(T_{k,d})$ is a bijection from $L(T^v_{k,d})$ to $[|L(T^v_{k,d})|]$. Let* $\mathsf{Schemes}(k,d)$ *denote the family of all schemes for $T_{k,d}$.*



J	K	L	N	O	M	P	R	Q	M	Q	L	N	O	J	P	R	K	P	K	L	N	M	O	J	R	Q
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

A	C	B	B	A	C	C	A	B	E	F	D	D	E	F	E	F	D	H	G	I	H	G	I	G	H	I
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

Figure 1 An illustration for Definition 7 with $k = d = 3$. The boxes represent copies of an instance I with $|\mathcal{T}| = 3$, the large instance is $J_{3,3}(I, \beta)$ for the scheme β listed at the bottom, and the dashed rectangle surrounds the instance $J_3 = J_{3,2}(I, \beta_3)$ where β_3 is a truncation of β to the right subtree of $T_{3,3}$. The hollow disks represent the sinks and sources on the large instance. All the arcs are oriented upwards. The leaves of $T_{3,3}$ are numbered as $1, 2, \dots, 27$. For the sake of legibility, most of the arcs in the last layer are omitted and the copies of the original instance within layers $2, 3$ are marked with letters. The letters are also used in the representation of the scheme β which contains 9 bijections between sets of size 3 and 3 bijections between sets of size 9 (and one bijection for size 27, which is immaterial here). The blue lines exemplify vertex pairs which belong to the request set of the large instance; the sources (in the layer 1) indexed by $6, 10, 23$ are mapped to the sinks in the copy M (in the layer 3). If a subset $\Gamma \subseteq [27]$ includes $6, 10, 23$ then it has a collision with respect to the scheme β. If we work with a no-instance then such a subset Γ of requests cannot be served as this would require routing three of them through the copy M.

We will now formalize the idea of connecting multiple copies of an instance. On an intuitive level, we construct a d-layered instance by taking k many $(d-1)$-layered instances and adding a new layer comprising k^{d-1} copies of the original instance I. Then we map the sinks in the layer $(d-1)$ to the sources in the layer d according to k bijections read from a scheme. These mappings govern how we place the arcs towards the layer d and which vertex pairs form the new request set. We need a scheme $\beta \in \mathsf{Schemes}(k, d)$ to arrange all the arcs between the layers.

In order to simplify the notation we introduce the following convention. Suppose that an instance J is being build with multiple disjoint copies of an instance $I = (D, k, \mathcal{T})$, referred to as $I_1, I_2, \dots$. Then we refer to the copy of the vertex $s_i \in V(D)$ (resp. t_i) in I_j as $I_j[s_i]$ (resp. $I_j[t_i]$).

▶ **Definition 7.** *Given an instance $I = (D, k, \mathcal{T})$ of* Dag Disjoint Paths *and a scheme $\beta = (f_v)_{v \in V(T_{k,d})} \in \mathsf{Schemes}(k, d)$ we construct an instance $J_{k,d}(I, \beta) = (D', k^d, \mathcal{T}')$ of* Dag Disjoint Paths*. The elements of $\mathcal{T}'$ will be indexed by the leaves of $T_{k,d}$ as $(s_v, t_v)_{v \in L(T_{k,d})}$ while the elements of $\mathcal{T}$ (in the instance I) are indexed by $1, \dots, k$ as $(s_i, t_i)_{i \in [k]}$.*

If $d = 1$, we simply set $J_{k,1}(I, \beta) = I$, ignoring β. We index $\mathcal{T}$ by $L(T_{k,1})$ in an arbitrary order.

Consider $d > 1$. Let r be the root of $T_{k,d}$ with $\mathsf{Children}(r) = \{u_1, \dots, u_k\}$. For $i \in [k]$ let β_i be the truncation of β to the nodes in the subtree $T^{u_i}_{k,d}$ and $J_i = (D_i, k^{d-1}, \mathcal{T}_i)$ be the instance $J_{k,d-1}(I, \beta_i)$. We take a disjoint union of $J_1, \dots, J_k$ and k^{d-1} copies of I referred to as $I_1, I_2, \dots$ (see Figure 1). These k^{d-1} copies of I form layer d.

Recall that for $i \in [k]$ the bijection f_{u_i} maps $L(T^{u_i}_{k,d})$ to $[k^{d-1}]$. For each $i \in [k]$ and $v \in L(T^{u_i}_{k,d})$ we insert an arc from $J_i[t_v]$ to $I_{f_{u_i}(v)}[s_i]$. Then we add the pair $(J_i[s_v], I_{f_{u_i}(v)}[t_i])$ to $\mathcal{T}'$. This pair is assigned index $\iota(v)$ in $\mathcal{T}'$ where ι is the natural embedding $L(T^{u_i}_{k,d}) \to L(T_{k,d})$.

Note that whenever D is acyclic then D' is acyclic as well so the procedure indeed outputs an instance of Dag Disjoint Paths. It is also clear that when I admits a solution serving all the k requests, it can be used to serve all the requests in $J_{k,d}(I, \beta)$.

▶ **Observation 8.** *Let $k, d \in \mathbb{N}$ and $\beta \in \mathsf{Schemes}(k, d)$. If $I = (D, k, \mathcal{T})$ is a yes-instance of Dag Disjoint Paths then $J_{k,d}(I, \beta)$ is a yes-instance as well.*

The case when I is a no-instance requires a more careful analysis. We introduce the notion of a *collision* that certifies that some subset of requests cannot be served.

▶ **Definition 9.** *Let $k, d \in \mathbb{N}$, $A \subseteq L(T_{k,d})$, and $\beta = (f_v)_{v \in V(T_{k,d})} \in \mathsf{Schemes}(k, d)$. We say that $u \in V(T_{k,d})$ forms a* collision *with respect to (A, β) if A contains elements $a_1, \ldots, a_k$ such that:*

1. *for each $i \in [k]$ the node a_i is a descendant of $u_i \in \mathsf{Children}(u)$ where $u_1, \ldots, u_k$ are distinct,*
2. *$f_{u_1}(a_1) = f_{u_2}(a_2) = \cdots = f_{u_k}(a_k)$.*

▶ **Lemma 10.** *Let $k, d \in \mathbb{N}$, $A \subseteq L(T_{k,d})$, and $\beta = (f_v)_{v \in V(T_{k,d})} \in \mathsf{Schemes}(k, d)$. Suppose that there exists a collision with respect to (A, β). Let $I = (D, k, \mathcal{T})$ be a no-instance of Dag Disjoint Paths. Then no solution to the instance $(D', k^d, \mathcal{T}') = J_{k,d}(I, \beta)$ can simultaneously serve all the requests $\{(s_v, t_v)_{v \in A}\}$.*

Proof. We will prove the lemma by induction on d. In the case $d = 1$ we have $J_{k,1}(I, \beta) = I$ and the only possibility of a collision is when $A = L(T_{k,1})$ so $\{(s_v, t_v)_{v \in A}\}$ is the set of all the requests. By definition, we cannot serve all the requests in a no-instance. Let us assume $d > 1$ from now on.

First suppose that the collision occurs at the root $r \in V(T_{k,d})$. Let $\mathsf{Children}(r) = \{u_1, \ldots, u_k\}$. Then there exists $A' = \{a_1, \ldots, a_k\} \subseteq A$ such that a_i is a descendant of u_i and $f_{u_1}(a_1) = f_{u_2}(a_2) = \cdots = f_{u_k}(a_k)$. We refer to this common value as $x = f_{u_i}(a_i)$. We will also utilize the notation from Definition 7.

Observe that in order to serve the request (s_{a_i}, t_{a_i}) in D' the path P_i starting at $s_{a_i} = J_i[s_{a_i}]$ must traverse the arc from $J_i[t_{a_i}]$ to $I_x[s_i]$ as every other arc leaving D_i leads to some I_y with $y \neq x$ having no connection to $t_{a_i} = I_x[t_i]$. Furthermore, the path P_i must contain a subpath connecting $I_x[s_i]$ to $I_x[t_i]$ in I_x. Since the same argument applies to every $i \in [k]$, we would have to serve all the k requests in I_x. But this is impossible because I_x is a copy of I which is a no-instance.

Now suppose that the collision does not occur at the root. Then it must occur in the subtree $T^{u_i}_{k,d}$ for some $i \in [k]$. For every $v \in A$ being a descendant of u_i, any path P_v serving the request (s_v, t_v) in D' must contain a subpath P'_v in D_i from $J_i[s_v]$ to $J_i[t_v]$ as again it must leave D_i through the vertex $J_i[t_v]$. By the inductive assumption, we know that we cannot simultaneously serve all the requests $(s_v, t_v)_{v \in A \cap L(T^{u_i}_{k,d})}$ in the smaller instance J_i. The lemma follows. ◀

We can now state our main technical theorem. Recall that a subset $A \subseteq L(T_{k,d})$ is called a q-subset if $|A| \geq |L(T_{k,d})|/q = k^d/q$.

▶ **Theorem 11.** *Let $k, d, q \in \mathbb{N}$ satisfy $d \geq k \cdot (4q)^{4k \log k}$. Then there exists $\beta \in \mathsf{Schemes}(k, d)$ such that for every q-subset $A \subseteq L(T_{k,d})$ there is a collision with respect to (A, β).*

The proof is postponed to Section 4 which abstracts from the Disjoint Paths problem and focuses on random permutations. With Theorem 11 at hand, the proof of the main result is easy.

▶ **Theorem 1.** *Let $q \in \mathbb{N}$ be a constant. It is W[1]-hard to distinguish whether for a given instance of k-Dag Disjoint Paths:*
1. *all the requests can be served simultaneously, or*
2. *no set of k/q requests can be served simultaneously.*

Proof. We are going to give an FPT-reduction from the exact variant of k-Dag Disjoint Paths, which is W[1]-hard [51], to the variant with a sufficiently large gap. To this end, we present an algorithm that, given an instance $I = (D, k, \mathcal{T})$, runs in time $f(k, q) \cdot |I|$ and outputs an instance $J = (D', k', \mathcal{T}')$ such that:
1. k' depends only on k and q,
2. if I is a yes-instance then J is a yes-instance, and
3. if I is a no-instance then no solution to J can simultaneously serve at least k'/q requests.

Obviously, being able to separate these two cases for J (all requests vs. at most $\frac{1}{q}$-fraction of requests) is sufficient to determine whether I is a yes-instance.

We set $d = k \cdot (4q)^{4k \log(k)}$ accordingly to Theorem 11. It guarantees that there exists a scheme $\beta \in \mathsf{Schemes}(k, d)$ such that for every q-subset $A \subseteq L(T_{k,d})$ there is a collision with respect to (A, β). Observe that such a scheme can be computed in time $f(k, q)$ because d is a function of (k, q) and the size of the family $\mathsf{Schemes}(k, d)$ is a function of (k, d). The same holds for the number of all q-subsets $A \subseteq L(T_{k,d})$. Therefore, we can simply iterate over all $\beta \in \mathsf{Schemes}(k, d)$ and check for each q-subset A whether there is a collision or not.

The instance J is defined as $J_{k,d}(I, \beta)$. A direct implementation of Definition 7 takes time $f(k, d) \cdot |I|$. Observation 8 says that if I is a yes-instance, then J is as well, whereas Lemma 10 ensures that if I is a no-instance, then for each set of k'/q requests (corresponding to some q-subset $A \subseteq L(T_{k,d})$ which must have a collision with β) no solution can simultaneously serve all of them. This concludes the correctness proof of the reduction. ◀

We remark that Theorem 1 works in a more general setting, where q is not necessarily a constant, but a function of k. This enables us to rule out not only an $\mathcal{O}(1)$-approximation in FPT time, but also an $\alpha(k)$-approximation for some slowly growing function $\alpha(k) \to \infty$. However, the value of the parameter k' becomes k^d for $d = \Omega(q^{k \log k})$ so q ends up very small compared to the new parameter k'. This is only sufficient to rule out approximation factors of the form $\alpha(k) = (\log k)^{o(1)}$. A detailed analysis of how to adjust such parameters is performed in [53].

4 Constructing the scheme

This section is devoted to the proof of Theorem 11. Before delving into the rigorous analysis, we sketch the main ideas behind the proof.

Outline. We use the probabilistic method to prove the existence of a scheme having a collision with every q-subset of leaves in $T_{k,d}$. We will show that for a sufficiently large d choosing each bijection at random yields a very high probability of a collision with any fixed q-subset. Specifically, the probability that a collision does not occur should be less than 2^{-k^d}. Since the number of all q-subsets of a k^d-size set is bounded by 2^{k^d}, the union bound will imply that the probability that a collision does not occur for at least one q-subset is strictly less than one, implying the existence of the desired scheme.

Let us fix a q-subset $A \subseteq L(T_{k,d})$. Suppose there is a vertex $u \in V(T_{k,d})$ such that for every child y of u the fraction of leaves in $T^y_{k,d}$ belonging to A is at least $1/q$. Let ℓ denote $[|L(T^y_{k,d})|]$. For each such child we choose a random bijection from $L(T^y_{k,d})$ to $[\ell]$. The probability that each of these k bijections maps an element of A to a fixed index $x \in [\ell]$ is at least q^{-k}. Such events are not independent for distinct x but we will see that they are negatively associated, which still allows us to upper bound the probability of no such event happening by $(1 - q^{-k})^\ell$ (see Lemma 12).

How to identify such a vertex u? First, it is sufficient for us to relax the bound $1/q$ assumed above to $1/(4q)$. Observe that for each layer in $T_{k,d}$ there must be many vertices v satisfying $|A \cap L(T^v_{k,d})| \ge \frac{1}{2q}|L(T^v_{k,d})|$. Suppose that v does not meet our criterion: this means that it has a child v' with less than $1/(4q)$-fraction of the A-leaves in its subtree. But then the average fraction of the A-leaves among the remaining children is higher than the fraction for v. Consequently, we can choose a child of v with a higher fraction and repeat this argument inductively. We show that after $\mathcal{O}(k \log(q))$ many steps this process must terminate so we are guaranteed to find a vertex for which every child has at least a $1/(4q)$-fraction of the A-leaves. This is proven in Lemma 13.

Finally, to obtain a large probability of a collision we must show that there many such vertices u with a large sum of their subtrees' sizes. This will allows us to multiply the aforementioned bounds of the form $(1 - q^{-k})^\ell$ with a large sum of the exponents ℓ. By applying the argument above to a single layer in $T_{k,d}$ we can find such a collection with the sum of their subtrees' sizes being k^d divided by some function of k and q. But we can also apply it to multiple layers as long as they are sufficiently far from each other (so that the vertices found by the inductive procedure are all distinct). Therefore, it suffices to take d large enough so that the number of available layers surpasses the factors in the denominator, which depend only on k and q. This is analyzed in Lemma 15.

We begin with a probabilistic lemma stating that randomly permuting k large subsets of a common universe yields a large chance of creating a non-empty intersection of these sets.

▶ **Lemma 12.** *Let $k, z, \ell \in \mathbb{N}$ and $X_1, \dots, X_k$ be subsets of $[\ell]$ of size at least ℓ/z each. Next, let $\Pi_1, \dots, \Pi_k \colon [\ell] \to [\ell]$ be independent random variables with a uniform distribution on the family of all permutations over the set $[\ell]$. Then*

$$\mathbb{P}\left[\Pi_1(X_1) \cap \Pi_2(X_2) \cap \dots \cap \Pi_k(X_k) = \emptyset\right] \le \exp(-\ell/z^k).$$

Proof. For $i \in [k]$ and $j \in [\ell]$ let $Y^i_j = 1$ if $j \in \Pi_i(X_i)$ and $Y^i_j = 0$ otherwise. By Theorem 5 the variables $(Y^i_1, \dots, Y^i_\ell)$ have negative association for each $i \in [k]$. Note that $\mathbb{E}Y^i_j \ge 1/z$. Next, let $Z_j = \sum_{i=1}^k Y^i_j$ for $j \in [\ell]$. Lemma 4 ensures that the variables $Z_1, \dots, Z_\ell$ also enjoy negative association. Condition $\Pi_1(X_1) \cap \Pi_2(X_2) \cap \dots \cap \Pi_k(X_k) = \emptyset$ is equivalent to $\max(Z_j)_{j=1}^\ell \le k - 1$. We have

$$\mathbb{P}\left[Z_j \le k-1\right] = 1 - \mathbb{P}\left[Z_j = k\right] = 1 - \prod_{i=1}^k \mathbb{P}\left[j \in \Pi_i(X_i)\right] \le 1 - 1/z^k.$$

$$\mathbb{P}\left[\max(Z_j)_{j=1}^\ell \le k-1\right] \le \prod_{j=1}^\ell \mathbb{P}\left[Z_j \le k-1\right] \le (1 - 1/z^k)^\ell = (1 - 1/z^k)^{z^k \cdot (\ell/z^k)} \le \exp(-\ell/z^k).$$

In the first inequality we used Lemma 3(3). The last one holds because $(1 - \frac{1}{m})^m < \frac{1}{e}$ for all $m \ge 2$. ◀

Notation. We introduce some additional notation to work with the tree $T_{k,d}$. For a vertex $v \in V(T_{k,d})$ let $\mathsf{Leaves}(v)$ denote the size of the set $L(T^v_{k,d})$. Note that $\mathsf{Leaves}(v) = k^{d-h}$ where h is the depth of v. Next, for a set $A \subseteq L(T_{k,d})$, we will write $\mathsf{Frac}_A(v) = |A \cap L(T^v_{k,d})| \,/\, \mathsf{Leaves}(v)$. When A is clear from the context, we will omit the subscript.

▶ **Lemma 13.** *Let $k, d, q, \tau \in \mathbb{N}$ satisfy $k, q \geq 2$ and $d \geq \tau \geq 2k \cdot \log(q)$. Next, let $v \in V(T_{k,d})$ be of depth at most $d - \tau$ and $A \subseteq L(T_{k,d})$ satisfy $\mathsf{Frac}_A(v) \geq \frac{1}{q}$. Then there exists a vertex $u \in V(T^{v,\tau}_{k,d})$ such that for each $y \in \mathsf{Children}(u)$ it holds that $\mathsf{Frac}_A(y) \geq \frac{1}{2q}$.*

Proof. Suppose the claim does not hold. We will show that under this assumption for each $i \in [\tau]$ there exists $v_i \in V(T^{v,i}_{k,d})$ with $\mathsf{Frac}(v_i) \geq \frac{1}{q} \cdot (1 + \frac{1}{2k-2})^{i-1}$. Then by substituting $i = \tau > (2k-2) \cdot \log(q)$ and estimating $(1 + \frac{1}{m})^m > 2$ (for all $m \geq 2$) we will arrive at a contradiction:

$$\mathsf{Frac}(v_\tau) \geq \frac{1}{q} \cdot \left(1 + \frac{1}{2k-2}\right)^{(2k-2)\cdot\log(q)} > \frac{1}{q} \cdot 2^{\log(q)} \geq 1.$$

We now construct the promised sequence (v_i) inductively. For $i = 1$ we set $v_1 = v$ which obviously belongs to $T^{v,1}_{k,d}$ and satisfies $\mathsf{Frac}(v) \geq \frac{1}{q}$. To identify v_{i+1} we consider $\mathsf{Children}(v_i) = u_1, u_2, \ldots, u_k$. We have $\mathsf{Frac}(v_i) = \frac{1}{k} \cdot \sum_{j=1}^k \mathsf{Frac}(u_j)$. We define v_{i+1} as the child of v_i that maximizes the value of Frac (see Figure 2). By the assumption, one of the children satisfies $\mathsf{Frac}(u_j) < \frac{1}{2q}$. Then $\mathsf{Frac}(v_{i+1})$ is lower bounded by the average value of Frac among the remaining $k-1$ children, which is at least $\frac{1}{k-1}\left(\mathsf{Frac}(v_i) \cdot k - \frac{1}{2q}\right)$. We have $\mathsf{Frac}(v_i) \geq \mathsf{Frac}(v_1) \geq \frac{1}{q}$ so $\left(\mathsf{Frac}(v_i) \cdot k - \frac{1}{2q}\right) \geq \left(\mathsf{Frac}(v_i) \cdot k - \frac{\mathsf{Frac}(v_i)}{2}\right)$. We check that v_{i+1} meets the specification:

$$\mathsf{Frac}(v_{i+1}) \geq \frac{\mathsf{Frac}(v_i)}{k-1} \cdot \left(k - \frac{1}{2}\right) = \mathsf{Frac}(v_i) \cdot \left(1 + \frac{1}{2k-2}\right) \geq \frac{1}{q} \cdot \left(1 + \frac{1}{2k-2}\right)^i$$

In the last inequality we have plugged in the inductive assumption. The lemma follows. ◀

To apply Lemma 13 we need to identify many vertices satisfying $\mathsf{Frac}_A(v) \geq \frac{1}{2q}$. To this end, we will utilize the following simple fact.

▶ **Lemma 14.** *Let $a_1, a_2, \ldots, a_\ell \in [0,1]$ be a sequence with mean at least x for some $x \in [0,1]$. Then at least $\frac{x\ell}{2}$ elements in the sequence are lower bounded by $\frac{x}{2}$.*

Proof. Suppose that $|\{a_i \geq \frac{x}{2} \mid i \in [\ell]\}| < \frac{x\ell}{2}$. This leads to a contradiction:

$$\sum_{i=1}^{\ell} a_i < 1 \cdot \frac{x\ell}{2} + \frac{x}{2} \cdot \ell = x\ell.$$ ◀

We will use the lemmas above for a fixed layer in the tree $T_{k,d}$ to identify multiple vertices v meeting the requirements of Lemma 13. For each such v we can find a close descendant u of v for which we are likely to observe a collision. The value ℓ in Lemma 12, governing the probability of a collision, corresponds to the number of leaves in the subtree of u, i.e., $\mathsf{Leaves}(u)$. Since this value appears in the exponent of the formula, we need a collection of such vertices u in which the total sum of $\mathsf{Leaves}(u)$ is large.

▶ **Lemma 15.** *Let $k, d, q \in \mathbb{N}$ satisfy $k, q \geq 2$, $d \geq 4kq$. If $A \subseteq L(T_{k,d})$ is a q-subset then there exists a set $F \subseteq V(T_{k,d})$ with the following properties.*
1. *For each $v \in F$ and $u \in \mathsf{Children}(v)$ it holds that $\mathsf{Frac}_A(u) \geq \frac{1}{4q}$.*
2. *The sum $\sum_{v \in F} \mathsf{Leaves}(v)$ equals at least $d \cdot k^d \cdot (4q)^{-3k\log(k)}$.*

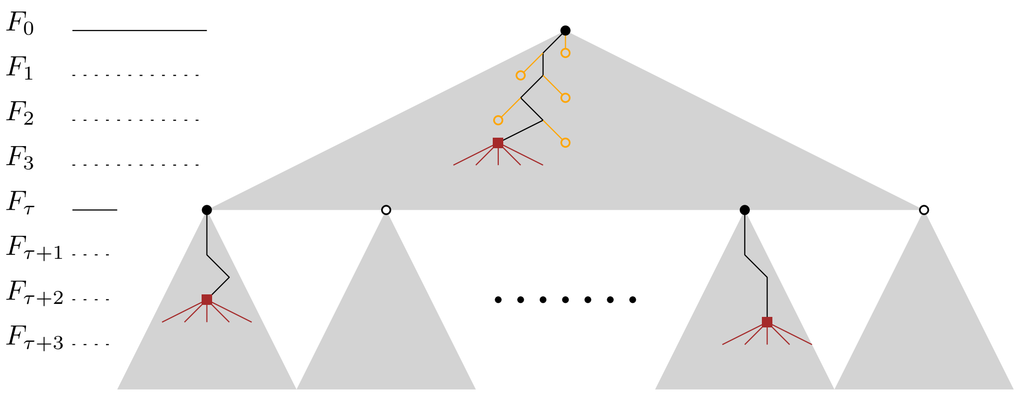


Figure 2 An illustration for Lemma 15. We consider layers $F_0, F_\tau, F_{2\tau}, \dots$ The vertices from F_0^+ and F_τ^+ are marked by black disks and their subtrees $F_{k,d}^{v,\tau}$ are depicted as gray triangles. For each vertex $v \in F^+$ we apply Lemma 13 to identify a vertex $\gamma(v) \in F$: the red square inside the corresponding triangle. The root also illustrates the argument from Lemma 13. We start with a vertex v satisfying $\mathsf{Frac}(v) \ge \frac{1}{2q}$ and while one of its children v' has $\mathsf{Frac}(v') < \frac{1}{4q}$ we can find another child v'' of v with $\mathsf{Frac}(v'') > \mathsf{Frac}(v)$. This process terminates within τ steps.

Proof. Let $F_i \subseteq V(T_{k,d})$ be i-th layer of $T_{k,d}$, i.e., the set of vertices of depth i; we have $|F_i| = k^i$ and $\mathsf{Leaves}(v) = k^{d-i}$ for each $v \in F_i$. Since their subtrees are disjoint, we can see that $\sum_{v\in F_i} |A \cap L(T_{k,d}^v)| = |A|$. Therefore $\sum_{v\in F_i} \mathsf{Frac}(v)/|F_i| \ge \frac{1}{q}$. By Lemma 14 at least $\frac{1}{2q}$ fraction of the vertices in F_i must satisfy $\mathsf{Frac}(v) \ge \frac{1}{2q}$. Let us denote this subset as F_i^+.

Let $\tau = \lceil 2k \cdot \log(2q) \rceil$ and $M = \lfloor d/\tau \rfloor$. We define $F^+ = F_0^+ \cup F_\tau^+ \cup F_{2\tau}^+ \cup \dots \cup F_{(M-1)\tau}^+$. Observe that for each pair $u, v \in F^+$ the trees $T_{k,d}^{u,\tau}$, $T_{k,d}^{v,\tau}$ are disjoint. We apply Lemma 13 with $q' = 2q$ to each $v \in F^+$ to obtain a vertex $\gamma(v) \in V(T_{k,d}^{v,\tau})$ satisfying condition (1). The disjointedness of these subtrees ensures that the vertices $\gamma(v)_{v\in F^+}$ are distinct. We define $F = \{\gamma(v) \mid v \in F^+\}$.

Now we take care of condition (2). Let us fix $j \in [0, M-1]$. Since $\gamma(v) \in V(T_{k,d}^{v,\tau})$ for v with the depth $j\tau$, we infer that the depth of $\gamma(v)$ is at most $(j+1)\tau - 1$ so $|\mathsf{Leaves}(\gamma(v))| \ge k^{d+1-(j+1)\tau}$. We have established already that $|F_{j\tau}^+| \ge \frac{|F_{j\tau}|}{2q} = \frac{k^{j\tau}}{2q}$. The assumption $d \ge 4kq$ implies $d \ge \tau$ so we can simplify $M = \lfloor d/\tau \rfloor \ge d/(2\tau)$. We estimate the sum within each layer $F_{j\tau}^+$ and then multiply it by M.

$$\sum_{v\in F_{j\tau}^+} \mathsf{Leaves}(\gamma(v)) \ge \frac{k^{j\tau}}{2q} \cdot k^{d+1-(j+1)\tau} = \frac{k^{d+1-\tau}}{2q}$$

$$\sum_{v\in F} \mathsf{Leaves}(v) = \sum_{j=0}^{M-1} \sum_{v\in F_{j\tau}^+} \mathsf{Leaves}(\gamma(v)) \ge \frac{d \cdot k^{d+1-\tau}}{2\tau \cdot 2q}$$

To get rid of the ceiling, we estimate $\tau \le 2k \cdot \log(4q)$. Then $k^\tau \le k^{2k\log(4q)} = (4q)^{2k\log(k)}$. We also use a trivial bound $\tau \le 4kq$. We can summarize the analysis by

$$\sum_{v\in F} \mathsf{Leaves}(v) \ge \frac{d \cdot k^{d+1-\tau}}{2\tau \cdot 2q} = \frac{d \cdot k^{d+1}}{k^\tau \cdot 4q\tau} \ge \frac{d \cdot k^{d+1}}{(4q)^{2k\log(k)} \cdot 16kq^2} \ge \frac{d \cdot k^d}{(4q)^{3k\log(k)}} \qquad \blacktriangleleft$$

Now we combine the gathered ingredients to show that a random scheme yields a high probability of a collision with any fixed q-subset. At this point we also adjust d to be larger then the factors depending on k and q.

▶ **Lemma 16.** *Let $k, d, q \in \mathbb{N}$ satisfy $d \geq k \cdot (4q)^{4k \log k}$. Consider some q-subset $A \subseteq L(T_{k,d})$. Suppose that we choose the scheme $\beta = (f_v)_{v \in V(T_{k,d})} \in \mathsf{Schemes}(k,d)$ by picking each bijection $f_v \colon L(T^v_{k,d}) \to [|L(T^v_{k,d})|]$ uniformly and independently at random. Then the probability that (A, β) has no collision is at most $\exp(-k^d)$.*

Proof. We apply Lemma 15 and use the obtained set $F \subseteq V(T_{k,d})$ to analyze the probability of getting a collision. Consider $u \in F$ with $\mathsf{Children}(u) = \{u_1, \dots, u_k\}$ and let C_u denote the event that (A, β) has a collision at u. For each $i \in [k]$ we have $\mathsf{Leaves}(u_i) = \mathsf{Leaves}(u)/k$ and we know from Lemma 15(1) that $\mathsf{Frac}_A(u_i) \geq 1/(4q)$. For each $i \in [k]$ a random bijection f_{u_i} is chosen between $L(T^{u_i}_{k,d})$ and $[\mathsf{Leaves}(u_i)]$. This can be interpreted as first picking an arbitrary bijection to $[\mathsf{Leaves}(u_i)]$ and then combining it with a random permutation over $[\mathsf{Leaves}(u_i)]$. We apply Lemma 12 with $z = 4q$ to infer that the probability of getting no collision at u is upper bounded by

$$\mathbb{P}\left[\neg C_u\right] \leq \exp\left(\frac{-\mathsf{Leaves}(u_i)}{z^k}\right) = \exp\left(\frac{-\mathsf{Leaves}(u)}{k \cdot (4q)^k}\right).$$

Since the sets $(\mathsf{Children}(u))_{u \in F}$ are pairwise disjoint, the corresponding events C_u are independent. We can thus upper bound the probability of getting no collision at all by the product $\prod_{u \in F} \mathbb{P}\left[\neg C_u\right]$. Next, by Lemma 15(2) and the assumption on d we know that

$$\sum_{u \in F} \mathsf{Leaves}(u) \geq d \cdot k^d \cdot (4q)^{-3k \log k} \geq k^{d+1} \cdot (4q)^k.$$

We combine this with the previous formula to obtain

$$\mathbb{P}\left[\neg \bigcup_{u \in F} C_u\right] = \mathbb{P}\left[\bigcap_{u \in F} \neg C_u\right] = \prod_{u \in F} \mathbb{P}\left[\neg C_u\right] \leq \exp\left(\frac{-\sum_{u \in F} \mathsf{Leaves}(u)}{k \cdot (4q)^k}\right) \leq \exp(-k^d). \blacktriangleleft$$

We are ready to prove Theorem 11 (restated below) and thus finish the proof of the reduction.

▶ **Theorem 11.** *Let $k, d, q \in \mathbb{N}$ satisfy $d \geq k \cdot (4q)^{4k \log k}$. Then there exists $\beta \in \mathsf{Schemes}(k,d)$ such that for every q-subset $A \subseteq L(T_{k,d})$ there is a collision with respect to (A, β).*

Proof. We choose the scheme β by picking each bijection uniformly and independently at random. For a fixed q-subset A let C_A denote the event that (A, β) witnesses a collision. In these terms, Lemma 16 says that $\mathbb{P}\left[\neg C_A\right] \leq \exp(-k^d)$. Let $\mathcal{A}$ be the family of all q-subsets $A \subseteq L(T_{k,d})$; we have $|\mathcal{A}| \leq 2^{k^d}$. By the union bound, the probability that there exists a q-subset with no collision with β is

$$\mathbb{P}\left[\bigcup_{A \in \mathcal{A}} \neg C_A\right] \leq \sum_{A \in \mathcal{A}} \mathbb{P}\left[\neg C_A\right] \leq 2^{k^d} \cdot (1/e)^{k^d} < 1.$$

Consequently, there is a positive probability of choosing a scheme β having a collision with every q-subset. In particular, this means that such a scheme exists. ◀

5 Conclusion

We have shown that no FPT algorithm can achieve an $\mathcal{O}(1)$-approximation for Max Disjoint Paths on acyclic digraphs. However, our reduction blows up the parameter significantly so it does not preserve a running time of the form $f(k)n^{o(k)}$. It is known that such a running time is unlikely for the exact variant of the problem [12]. This leads to a question whether Max Dag Disjoint Paths admits an $\mathcal{O}(1)$-approximation that is faster than $n^{\mathcal{O}(k)}$.

Our proof yields an alternative technique for gap amplification in a parameterized reduction based on the probabilistic method (extending the restricted version appearing in [53]), compared to reductions relying on coding theory [39, 25] or communication complexity [30]. Can this approach come in useful for proving that Parameterized Inapproximability Hypothesis (PIH) follows from FPT≠W[1]?

References

1 Isolde Adler, Stavros G. Kolliopoulos, Philipp Klaus Krause, Daniel Lokshtanov, Saket Saurabh, and Dimitrios M. Thilikos. Irrelevant vertices for the planar disjoint paths problem. *J. Comb. Theory, Ser. B*, 122:815–843, 2017. doi:10.1016/j.jctb.2016.10.001.

2 Saeed Akhoondian Amiri, Stephan Kreutzer, Dániel Marx, and Roman Rabinovich. Routing with Congestion in Acyclic Digraphs. In Piotr Faliszewski, Anca Muscholl, and Rolf Niedermeier, editors, *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:11, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2016.7.

3 Matthias Bentert, Fedor V. Fomin, and Petr A. Golovach. Tight approximation and kernelization bounds for vertex-disjoint shortest paths. *arXiv*, abs/2402.15348, 2024. doi:10.48550/arXiv.2402.15348.

4 Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *Journal of the ACM (JACM)*, 68(3):1–40, 2021. doi:10.1145/3444942.

5 Dario Giuliano Cavallaro, Ken-ichi Kawarabayashi, and Stephan Kreutzer. Edge-disjoint paths in eulerian digraphs. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 704–715. ACM, 2024. doi:10.1145/3618260.3649758.

6 Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-Exponential Time Hypothesis to fixed parameter tractable inapproximability: Clique, dominating set, and more. *SIAM J. Comput.*, 49(4):772–810, 2020. doi:10.1137/18M1166869.

7 Parinya Chalermsook, Bundit Laekhanukit, and Danupon Nanongkai. Pre-reduction graph products: Hardnesses of properly learning DFAs and approximating EDP on dags. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 444–453. IEEE, 2014. doi:10.1109/FOCS.2014.54.

8 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Edge-disjoint paths in planar graphs. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 71–80. IEEE, 2004. doi:10.1109/FOCS.2004.27.

9 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of computing*, 2(1):137–146, 2006. doi:10.4086/TOC.2006.V002A007.

10 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Edge-disjoint paths in planar graphs with constant congestion. *SIAM J. Comput.*, 39(1):281–301, 2009. doi:10.1137/060674442.

11 Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. A note on multiflows and treewidth. *Algorithmica*, 54(3):400–412, 2009. doi:10.1007/S00453-007-9129-Z.

12 Rajesh Chitnis. A tight lower bound for edge-disjoint paths on planar dags. *SIAM Journal on Discrete Mathematics*, 37(2):556–572, 2023. doi:10.1137/21M1395089.

13 Kyungjin Cho, Eunjin Oh, and Seunghyeok Oh. Parameterized algorithm for the disjoint path problem on planar graphs: Exponential in k^2 and linear in n. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3734–3758. SIAM, 2023. doi:10.1137/1.9781611977554.CH144.

14 Julia Chuzhoy, David H. K. Kim, and Rachit Nimavat. New hardness results for routing on disjoint paths. *SIAM J. Comput.*, 51(2):17–189, 2022. doi:10.1137/17M1146580.

15 Julia Chuzhoy, David HK Kim, and Shi Li. Improved approximation for node-disjoint paths in planar graphs. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 556–569, 2016. doi:10.1145/2897518.2897538.

16 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.

17 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 197–206, 2013. doi:10.1109/FOCS.2013.29.

18 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

19 Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.

20 Alina Ene, Matthias Mnich, Marcin Pilipczuk, and Andrej Risteski. On routing disjoint paths in bounded treewidth graphs. In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 15:1–15:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.SWAT.2016.15.

21 Andreas Emil Feldmann, Karthik C S, Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. doi:10.3390/A13060146.

22 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

23 András Frank. Packing paths, cuts, and circuits - a survey. *Paths, Flows and VLSI-Layout*, 49:100, 1990.

24 Archontia C. Giannopoulou, Ken-ichi Kawarabayashi, Stephan Kreutzer, and O-joung Kwon. Directed tangle tree-decompositions and applications. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 377–405. SIAM, 2022. doi:10.1137/1.9781611977073.19.

25 Venkatesan Guruswami, Bingkai Lin, Xuandi Ren, Yican Sun, and Kewen Wu. Parameterized inapproximability hypothesis under exponential time hypothesis. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 24–35. ACM, 2024. doi:10.1145/3618260.3649771.

26 Venkatesan Guruswami, Xuandi Ren, and Sai Sandeep. Baby PIH: Parameterized Inapproximability of Min CSP. In Rahul Santhanam, editor, *39th Computational Complexity Conference (CCC 2024)*, volume 300 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:17, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CCC.2024.27.

27 Pinar Heggernes, Pim van't Hof, Erik Jan van Leeuwen, and Reza Saei. Finding disjoint paths in split graphs. *Theory of Computing Systems*, 57:140–159, 2015. doi:10.1007/S00224-014-9580-6.

28 Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983.

29 Karthik C. S. and Subhash Khot. Almost polynomial factor inapproximability for parameterized k-Clique. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.CCC.2022.6.

30 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019. doi:10.1145/3325116.

31 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Stephan Kreutzer. An excluded half-integral grid theorem for digraphs and the directed disjoint paths problem. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 70–78, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2591796.2591876.

32 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/J.JCTB.2011.07.004.

33 Ken-ichi Kawarabayashi and Stephan Kreutzer. The directed grid theorem. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 655–664, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2746539.2746586.

34 Jon M. Kleinberg and Éva Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *J. Comput. Syst. Sci.*, 57(1):61–73, 1998. doi:10.1006/JCSS.1998.1579.

35 Stavros G Kolliopoulos and Clifford Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004. doi:10.1007/S10107-002-0370-6.

36 Tuukka Korhonen, Michał Pilipczuk, and Giannos Stamoulis. Minor containment and disjoint paths in almost-linear time. *arXiv*, abs/2404.03958, 2024 (to appear at FOCS 2024). doi:10.48550/arXiv.2404.03958.

37 Mark R. Kramer and Jan van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. *Advances in Computing Research*, 2:129–146, 1984.

38 Michael Lampis and Manolis Vasilakis. Parameterized maximum node-disjoint paths. *arXiv*, abs/2404.14849, 2024. doi:10.48550/arXiv.2404.14849.

39 Bingkai Lin. Constant approximating k-Clique is W[1]-hard. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1749–1756. ACM, 2021. doi:10.1145/3406325.3451016.

40 Bingkai Lin, Xuandi Ren, Yican Sun, and Xiuhan Wang. Constant approximating parameterized k-SetCover is W[2]-hard. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 3305–3316. SIAM, 2023. doi:10.1137/1.9781611977554.CH126.

41 Daniel Lokshtanov, Pranabendu Misra, Michał Pilipczuk, Saket Saurabh, and Meirav Zehavi. An exponential time parameterized algorithm for planar disjoint paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 1307–1316, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3357713.3384250.

42 Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Meirav Zehavi. Parameterized complexity and approximability of directed odd cycle transversal. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2181–2200, 2020. doi:10.1137/1.9781611975994.134.

43 Daniel Lokshtanov, Saket Saurabh, and Vaishali Surianarayanan. A parameterized approximation scheme for min k-Cut. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 798–809. IEEE, 2020. doi:10.1109/FOCS46700.2020.00079.

44 James F Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3):31–36, 1975.

45 Sridhar Natarajan and Alan P Sprague. Disjoint paths in circular arc graphs. *Nordic Journal of Computing*, 3(3):256–270, 1996.

46 Richard G. Ogier, Vladislav Rutenburg, and Nachum Shacham. Distributed algorithms for computing shortest pairs of disjoint paths. *IEEE Trans. Inf. Theory*, 39(2):443–455, 1993. doi:10.1109/18.212275.

47 Naoto Ohsaka. On the parameterized intractability of determinant maximization. *Algorithmica*, pages 1–33, 2024. doi:10.1007/S00453-023-01205-0.

48 Bruce Reed. Rooted routing in the plane. *Discrete Applied Mathematics*, 57(2-3):213–227, 1995. doi:10.1016/0166-218X(94)00104-L.

49 Neil Robertson and Paul D Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of combinatorial theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

50 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

51 Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM J. Discret. Math.*, 24(1):146–157, 2010. doi:10.1137/070697781.

52 Anand Srinivas and Eytan H. Modiano. Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wirel. Networks*, 11(4):401–417, 2005. doi:10.1007/S11276-005-1765-0.

53 Michał Włodarczyk. Parameterized inapproximability for steiner orientation by gap amplification. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 104:1–104:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPICS.ICALP.2020.104.

54 Michał Włodarczyk and Meirav Zehavi. Planar disjoint paths, treewidth, and kernels. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 649–662. IEEE, 2023. doi:10.1109/FOCS57990.2023.00044.