Revisiting Directed Disjoint Paths on tournaments (and relatives)

Guilherme C. M. Gomes
LIRMM, Université de Montpellier, CNRS, Montpellier, France
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Raul Lopes
LIRMM, Université de Montpellier, CNRS, Montpellier, France
Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

Ignasi Sau
LIRMM, Université de Montpellier, CNRS, Montpellier, France

Abstract

In the Directed Disjoint Paths problem (k-DDP), we are given a digraph and k pairs of terminals, and the goal is to find k pairwise vertex-disjoint paths connecting each pair of terminals. Bang-Jensen and Thomassen [SIAM J. Discrete Math. 1992] claimed that k-DDP is NP-complete on tournaments, and this result triggered a very active line of research about the complexity of the problem on tournaments and natural superclasses. We identify a flaw in their proof, which has been acknowledged by the authors, and provide a new NP-completeness proof. From an algorithmic point of view, Fomin and Pilipczuk [J. Comb. Theory B 2019] provided an FPT algorithm for the edge-disjoint version of the problem on semicomplete digraphs, and showed that their technique cannot work for the vertex-disjoint version. We overcome this obstacle by showing that the version of k-DDP where we allow congestion c on the vertices is FPT on semicomplete digraphs provided that c is greater than $k/2$. This is based on a quite elaborate irrelevant vertex argument inspired by the edge-disjoint version, and we show that our choice of c is best possible for this technique, with a counterexample with no irrelevant vertices when $c \leq k/2$. We also prove that k-DDP on digraphs that can be partitioned into h semicomplete digraphs is W[1]-hard parameterized by $k + h$, which shows that the XP algorithm presented by Chudnovsky, Scott, and Seymour [J. Comb. Theory B 2019] is essentially optimal.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics → Graph theory → Graph algorithms.

Keywords and phrases directed graphs, tournaments, semicomplete digraphs, directed disjoint paths, congestion, parameterized complexity, directed pathwidth.

Related Version A conference version of this article will appear in the proceedings of *ICALP 2025*. We would like to thank the reviewers for their helpful remarks.

Funding *Guilherme C. M. Gomes*: Funded by the European Union, project PACKENUM, grant number 101109317. Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.
Raul Lopes: French project ELIT (ANR-20-CE48-0008-01) and HIDSS-0002 DASHH (Data Science in Hamburg - Helmholtz Graduate School for the Structure of Matter).
Ignasi Sau: French project ELIT (ANR-20-CE48-0008-01).

1 Introduction

The Disjoint Paths problem is one of the most well-studied classical NP-complete graph problems [29]. It consists in, given an undirected graph G and k *requests*, which are pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$ known as *terminals*, deciding whether G contains k

pairwise vertex-disjoint paths connecting each s_i to t_i, for each $i \in [k]$. As a crucial ingredient of their Graph Minors project, Robertson and Seymour [40] proved that Disjoint Paths is *fixed-parameter tractable* (FPT) parameterized by k, that is, it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ on n-vertex graphs for some computable function f. In this article we focus on its directed counterpart, defined analogously for an input digraph D and called Directed Disjoint Paths, which is known to be much harder from a computational point of view: it is already NP-complete for a fixed number $k = 2$ of terminals [25]. A number of approaches have been proposed to cope with this intractability, ranging from approximation algorithms [12, 43], heuristics [7, 38], parameterized algorithms [23, 41], restricting the input digraph D [2, 16, 25–27], or relaxing the problem by allowing congestion on vertices [1, 10, 20, 31, 33, 34, 37]. In this work we consider and combine the latter three approaches, which we proceed to discuss.

Let us start by restricting the graph class to which the input graph of Directed Disjoint Paths belongs. Two relevant classes of digraphs are typically considered when one seeks to improve the tractability of a problem: directed acyclic graphs (DAGs) and *tournaments* (that is, digraphs that can be obtained from a complete graph by orienting each edge). For the former, the (parameterized) complexity of Directed Disjoint Paths is well understood: the problem is NP-complete and solvable in time $n^{\mathcal{O}(k)}$ (hence, in the class XP) [25], and W[1]-hard [41] (even to approximate within a constant factor [43]), thus unlikely to be FPT. For the latter, the landscape is more murky, and the goal of this article is to contribute to understanding it a bit better.

Bang-Jensen and Thomassen [2] *claimed* that Directed Disjoint Paths is NP-complete on tournaments when k is part of the input, and this result triggered a long line of research. In the same paper, Bang-Jensen and Thomassen [2] showed that the problem can be solved in polynomial time for $k = 2$. In fact, their algorithm works on the larger class of *semicomplete digraphs*, where each pair of distinct vertices has at least one arc between them, instead of exactly one as in tournaments. Chudnovsky, Scott, and Seymour [16] showed that the problem can be solved in polynomial time for every fixed k on semicomplete digraphs, by providing an XP algorithm. Fradkin and Seymour [27] proved that the edge-disjoint version of the problem, for which all the results discussed above also apply, is also polynomial-time solvable on semicomplete digraphs for fixed k. It is worth mentioning that Chudnovsky, Fradkin, Kim, Scott, and Seymour [13, 15, 16, 26, 27, 35] built a containment theory on semicomplete digraphs for the study of minor-related problems, such as Directed Disjoint Paths (see also the work of Barbero, Paul, and Pilipczuk [5]). One of the key notions in this theory is a width measure of digraphs called *directed pathwidth* – a generalization of undirected pathwidth (see Section 3 for the definition) – that also plays a role in the current article.

Back to the Directed Disjoint Paths problem, the XP algorithm on tournaments [16] has been generalized by Chudnovsky, Scott, and Seymour [14] to the class $\mathcal{C}_h$ of digraphs whose vertex set can be partitioned into a bounded number h of semicomplete digraphs (thus, semicomplete digraphs correspond to the class $\mathcal{C}_1$). More precisely, the running time of their algorithm is $n^{\mathcal{O}\left((hk)^5\right)}$. The authors asked whether the result can be further generalized to the class $\mathcal{A}_h$ of digraphs whose underlying graph has independence number at most h, and this is still open. For the edge-disjoint version of the problem, an affirmative answer was given by Fradkin and Seymour [27].

Concerning the fixed-parameter tractability of the problem, it is open whether Directed Disjoint Paths on tournaments (or semicomplete digraphs) is FPT parameterized by k. Interestingly, the edge-disjoint version of the problem was shown to be FPT on semicomplete digraphs by Fomin and Pilipczuk [23], by solving a more general problem called Rooted

Immersion. In a nutshell (more details are given below when discussing our techniques in Section 2), theirs is a win/win approach based on the directed pathwidth of the input digraph. If the pathwidth is small (as a function of k), then a dynamic programming algorithm is used to solve the problem. Otherwise, it is shown that the digraph must contain a large obstruction to directed pathwidth called a *triple* [26], which is used to find an *irrelevant vertex* for the problem, that is, a vertex whose removal does not affect the existence of a solution.

Note that the above win/win approach needs, as a first step, to compute the directed pathwidth of the input digraph in FPT-time. While this problem is open on general digraphs, an FPT algorithm on semicomplete digraphs is given in the same article [23]. This FPT algorithm to compute directed pathwidth was generalized by Kitsunai, Kobayashi, and Tamaki [36] to another superclass of semicomplete digraphs called *h-semicomplete digraphs* and denoted by $\mathcal{S}_h$ for some fixed integer $h \geq 0$. Digraphs in this class satisfy the property that each vertex has at most h non-neighbors (in any direction). If we denote by $\mathcal{T}$ and $\mathcal{S}$ the classes of tournaments and semicomplete digraphs, respectively, it holds that $\mathcal{S} = \mathcal{S}_0$, and for every $h \geq 0$ (see the discussion in the beginning of Subsection 3.2),

$$\mathcal{T} \subseteq \mathcal{S} \subseteq \mathcal{S}_h \subseteq \mathcal{C}_{h+1} \subseteq \mathcal{A}_{h+1}. \tag{1}$$

Another transversal strategy to try to overcome the inherent hardness of Directed Disjoint Paths is to relax the problem by allowing every vertex of D to be used by at most c of the k paths of the solution, for some integer $c \geq 1$ that is also part of the input and is called the *congestion*; we call the corresponding problem Directed Disjoint Paths with Congestion. Note that if $c \geq k$ the problem is trivial, so we may assume that $c \leq k-1$. It is open whether the case $k = 3$ and $c = 2$ (which is the first non-trivial one with congestion greater than one) can be solved in polynomial time on general digraphs. There are, however, some positive results. For instance, Edwards, Muzi, and Wollan [20] showed that the problem for $c = 2$ can be solved in polynomial time if the input graph is sufficiently connected as a function of k, which is not the case for the truly disjoint version [42]. See [10] for recent improvements of this result. A popular variant of the congested problem is an *asymmetric* version, where the goal is to either find a congested solution or to provide a no-answer for the disjoint version. This problem has been proved to be XP parameterized by k on general digraphs for some small values of c [31, 33, 34], usually exploiting the celebrated Directed Grid Theorem [9, 32, 34]. On the other hand, other articles [1, 37] study the parameterized complexity of Directed Disjoint Paths with Congestion by considering parameters stronger than k.

Finally, it is worth mentioning that Cavallaro, Kawarabayashi, and Kreutzer [11] recently proved that the edge-disjoint version of Directed Disjoint Paths is FPT on Eulerian digraphs parameterized by k; this is one of the rare examples where (some variant of) the problem is FPT. We refer to the book of Bang-Jensen and Gutin [4] for a thorough introduction to algorithms on digraphs, in particular on tournaments and related superclasses.

In the sequel, for notational conciseness we may use the abbreviations k-DDP and (k, c)-DDP to refer to the Directed Disjoint Paths and Directed Disjoint Paths with Congestion problems, respectively. We permit ourselves to slightly abuse notation by including the integers k and c in the abbreviated names of the problems, even if they are part of the corresponding inputs.

Our contributions. As mentioned above, the NP-completeness proof of Bang-Jensen and Thomassen [2] of k-DDP on tournaments triggered intensive research in this area [14, 16, 23].

Unfortunately, we realized that their proof has a flaw that does not seem to be easily fixable, as acknowledged by the authors [3]; see Appendix A for an explanation of this flaw. Our first contribution is to provide a new (correct) proof of this result.

▶ **Theorem 1.** Directed Disjoint Paths *on tournaments is* NP*-complete.*

As mentioned above, it is open whether k-DDP on tournaments is FPT parameterized by k. Recall that the win/win approach of Fomin and Pilipczuk [23] for the edge-disjoint version has two main ingredients (other than computing the directed pathwidth): a dynamic programming algorithm and an irrelevant vertex argument. While the former is claimed to exist for the vertex-disjoint version [23], the latter one is doomed to fail: Fomin and Pilipczuk [23] provide a counterexample even for $k = 2$ consisting of a family of tournaments containing arbitrarily large triples (that are the structures where irrelevant vertices are found), but in which each vertex is relevant. Our next contribution is to prove that this obstacle disappears if we allow for a large congestion.

▶ **Theorem 2.** (k, c)-DDP *on semicomplete digraphs is* FPT *parameterized by* k *restricted to instances satisfying* $c > k/2$.

Note that since we can assume that $c \leq k-1$, the result of Theorem 2 covers roughly "half" of the range of values of the congestion c. It is natural to ask whether the problem remains NP-complete for this range of values of c, that is, when the congestion is lower-bounded by a *linear* function of k. This question is still open, but we provide the following hardness result, where the congestion c is *almost linear* in k (as ε approaches one).

▶ **Theorem 3.** (k, c)-DDP *remains* NP*-complete on tournaments even restricted to instances satisfying* $c = k^{\varepsilon}$*, for every* $\varepsilon \in [0, 1)$.

As discussed before, Chudnovsky, Scott, and Seymour [14] showed that Directed Disjoint Paths on the class $\mathcal{C}_h$ can be solved in time $n^{\mathcal{O}((hk)^5)}$. Our next result is to show that this algorithm is somehow optimal, in the sense that it is unlikely to get rid of both parameters in the exponent of n, even restricted to digraphs of bounded directed pathwidth.

▶ **Theorem 4.** *The* k-DDP *problem on* $\mathcal{C}_h$ *is* W[1]*-hard when parameterized by* $k + h$*, even if restricted to input digraphs of directed pathwidth two.*

Moreover, Theorem 4 can be generalized to show hardness for (k, c)-DDP when $c > k/2$ (cf. Theorem 19, which, alongside Equation 1, implies that the win/win strategy cannot be extended beyond h-semicomplete graphs). we summarize of our main contributions in Table 2.

Organization. In Section 2 we give an overview of the techniques used to obtain our results. In Section 3 we provide preliminaries about general digraphs, tournaments, related classes, directed pathwidth, and parameterized complexity. In Section 4 (resp. Section 5) we provide our negative (resp. positive) results. We conclude the article in Section 6 with some directions for further research.

2 Overview of our techniques

The reduction that we use to prove Theorem 1 is novel and versatile enough so that we can build upon it and modify it appropriately to prove the NP-completeness of several variants of the problem. Intuitively, in the proof of Theorem 1 we reduce from a variant of 3-SAT where each variable has a bounded number of occurrences (namely, exactly three positive and one negative). This allows us to build an instance of k-DDP where a variable's truth value is determined by two requests (cf. Figure 3), while the satisfaction of the clauses is encoded using one additional request per clause; interestingly, the paths that fulfill the requests have length at most five. By extending this construction using a single long path, named the *critical path* (cf. the black path in Figure 5), which is the unique way to fulfill several additional requests, we show how to prove NP-hardness for (k, c)-DDP for digraphs in $\mathcal{C}_2$ even if $c = \varepsilon k$ for $\varepsilon \in [0, 1)$ (cf. Theorem 11). With this approach, however, we are unable to prove hardness for (k, c)-DDP instances where $c = \varepsilon k$ and D is a tournament, only doing so for the weaker relation $c = k^{\varepsilon}$ (cf. Theorem 3).

For the FPT algorithm given in Theorem 2, the challenge is that now we deal with vertex-disjoint paths, instead of edge-disjoint paths as in the original work by Fomin and Pilipczuk [23]. Much like theirs, our proof is a win/win approach that makes extensive use of *k-triples* (cf. Definition 5), an obstacle for directed pathwidth on tournaments introduced by Fradkin and Seymour [26], and has two steps: (i) show that, in a minimum solution, almost all vertices (in their case, arcs) of a sufficiently large (i.e., with more than $f(k)$ vertices) triple can be used by other paths, and (ii) identify a vertex in the triple that can be safely removed from the instance. In the edge-disjoint setting, it was extensively used that, in a large enough triple, vertices had large in- or out-degree; consequently, finding available arcs to either shortcut (step i) or reroute (step ii) a solution was relatively simple. In the vertex-disjoint case, this does not happen; in fact, Fomin and Pilipczuk [23] present an infinite family of tournaments that have no irrelevant vertex for k-DDP. As we show in Subsection 3.3, their counterexamples are also valid when $c \leq k/2$ but, when $c > k/2$, we are able to easily find alternative paths, freeing up several vertices and thus making them irrelevant. In particular, we are able to implement step i using, in particular, the pigeonhole principle: any two fully congested vertices have at least one path in common. This is not enough by itself, and we must carefully construct two shortcuts simultaneously, instead of only one, to show that vertices occupied by exactly c paths only amount to $\mathcal{O}\left(\binom{k}{c}\right)$ (see Lemma 25 elements of the triple. Step ii also offers additional challenges; in particular, when dealing with the exterior neighborhood of the triple, we must find large matchings entering and leaving the triple to properly reroute the paths using it. As such, instead of the polynomial on k used by Fomin and Pilipczuk, we now require that the triple has size of the order of $2^{\mathcal{O}(k \log k)}$.

Given the success of the win/win algorithms based on directed pathwidth, it is natural to see how far one can push this approach. The results of Kitsunai, Kobayashi, and Tamaki [36] that triples are also directed pathwidth obstacles for h-semicomplete digraphs is a further step in this direction. By Theorem 4 and Equation 1 however, the class of h-semicomplete digraphs is essentially as far as it goes, even if we allow for congestion (cf. Theorem 19). To prove Theorem 4, we take inspiration from Slivkins' [41] proof that k-Edge Disjoint Paths is W[1]-hard parameterized by k on DAGs. In his reduction from the Clique instance (G, ℓ), a DAG in a matrix-like format is built in a way that each row corresponds to a copy of G and each column to a vertex. Moreover, only one cell may be available in each row, and this must correspond to a vertex of G in the clique. Requests are added so that ℓ of them are used to enforce the uniqueness, while $\binom{\ell}{2}$ are used to encode the edges of the clique (cf. Figure 8). The uniqueness of the available cell is the main obstacle to obtain Theorem 4. In Slivkins' proof, it is obtained by placing two parallel paths, named a and b, with a request from the first vertex of a to the last of b, which may only be fulfilled if we perform a jump at the appropriate point: from where the segment of a corresponding to the desired vertex begins to where the corresponding segment in b ends. We adapt his proof by making each of the paths a and b a tournament (cf. Figure 6). This, however, opens us to cheating: an edge-encoding path could randomly walk along a suffix of a or a prefix of b and break the proof. To overcome this, we require that a and b are completely occupied, except at the segments corresponding to the desired vertex of G; this is achieved by introducing several structures and requests in order to enforce a synchronous behavior in each row of the matrix (cf. **??**). Using a long path strategy, similarly to the one used in Theorem 11, we are able to force that every vertex participates in the fulfillment of several requests while keeping both the congestion high ($c = k/d$ for some constant $1 < d \leq k$) and the directed pathwidth equal to two, proving that a statement analogous to Theorem 4 (i.e., Theorem 19) also holds for (k, c)-DDP.

3 Definitions and preliminaries

In this section we provide basic preliminaries about digraphs (including the definition of directed pathwidth), parametererized complexity, and tournaments and related classes along with the notion of k-triple. In Subsection 3.3 we present and analyze the counterexample given by Fomin and Pilipczuk [23] that shows that the irrelevant vertex technique is not extensible to k-DDP.

3.1 Digraphs, directed pathwidth, and parametrized complexity

For basic background on graph theory we refer the reader to [6]. Since in this article we mainly work with digraphs, we focus on basic definitions of digraphs, often skipping their undirected counterparts. Given a digraph G we denote by $V(G)$ and $E(G)$ the sets of vertices and arcs of G, respectively. All involved digraphs are simple, i.e., they have neither loops nor parallel arcs. Given $X \subseteq V(G)$, we denote by $G \setminus X$ the digraph resulting from removing every vertex of X from G. We denote by $G[X]$ the subgraph of G *induced* by X.

If e is an arc of a digraph from a vertex u to a vertex v, we say that e has *endpoints* u and v, that e is *incident* to u and v, and that e is *oriented* from u to v. We may refer to e as the ordered pair (u, v). In this case, u is the *tail* of e and v is the *head* of e. We also say that e is *leaving* u and *reaching* v, and that u and v are *adjacent*. A *clique* in a (di)graph G is a set of pairwise adjacent vertices of G, and an *independent set* is a set of pairwise non-adjacent vertices of G. A pair of arcs is adjacent if they share an endpoint. A *matching* is a set of pairwise non-adjacent arcs. For $A, B \subseteq V(G)$, we say that M is a matching *from A to B* if every arc of M has tail in A and head in B. The *in-degree* (resp. *out-degree*) of a vertex v in a digraph G is the number of arcs with head (resp. tail) v. We denote the in-degree and out-degree of v by $\deg^-_G(v)$ and $\deg^+_G(v)$, respectively.

The *in-neighborhood* $N^-_D(v)$ of v is the set $\{u \in V(D) \mid (u, v) \in E(G)\}$, and the *out-neighborhood* $N^+_D(v)$ is the set $\{u \in V(D) \mid (v, u) \in E(G)\}$. We say that u is an *in-neighbor* of v if $u \in N^-_D(v)$ and that u is an *out-neighbor* of v if $u \in N^+_D(v)$. We extend these notations to sets of vertices: given $X \subseteq V(D)$, we define $N^-_D(X) = (\bigcup_{v \in X} N^-_D(v)) \setminus X$ and $N^+_D(X) = (\bigcup_{v \in X} N^+_D(v)) \setminus X$.

A *walk* in a digraph G is an alternating sequence W of vertices and arcs that starts and ends with a vertex, and such that for every arc (u, v) in the walk, vertex u (resp. vertex v) is the element right before (resp. right after) arc (u, v) in W. If the first vertex in a walk is u and the last one is v, then we say this is a *walk from u to v*. A *path* on p vertices is a digraph formed by p pairwise distinct vertices; to avoid confusion with sets of vertices, $P = \langle v_1, \ldots, v_p \rangle$ always denotes the path P *starting* at v_1, *ending* at v_p, following the arcs in the given order of the vertices. We denote by $\prec_P$ the *comes before than in P* relation, i.e.,

$v_i \prec_P v_j$ if and only if $i < j$. When the ordering itself is unimportant, we use $V(P)$ to refer to its vertex set.

Given $X, Y \subseteq V(D)$, we say that X is *complete to* Y if $(u, v) \in E(D)$ whenever $u \in X$ and $v \in Y$.

For a positive integer k, we denote by $[k]$ the set of integers $\{1, \ldots, k\}$.

Directed pathwidth and k-triples. A *path decomposition* of a digraph D is a sequence $(X_1, \ldots, X_p)$ of vertex subsets of D, called *bags*, such that the following three conditions are satisfied:

1. $\bigcup_{i\in[p]} X_i = V(D)$,
2. for every arc $(u, v) \in E(D)$, we either have $u, v \in X_i$ for some X_i, or $u \in X_i$ and $v \in X_j$ for some $i \geq j$, and
3. for every vertex $v \in V(D)$, the set $\{i \mid v \in X_i\}$ of indices of the bags containing v forms a single integer interval.

Similarly to undirected pathwidth, the *width* of a path decomposition $(X_1, \ldots, X_p)$ of a digraph D is defined as $\max_{i\in[p]} |X_i| - 1$, and the *directed pathwidth* of D, denoted by $\mathsf{dpw}(D)$, is the smallest integer ℓ such that there exists a path decomposition of D of width ℓ.

▶ **Definition 5** (k-triple). *Let D be a digraph. For an integer $k \geq 1$, a k*-triple *$\mathcal{K}$ of D is formed by an ordered triple (A, B, C) of disjoint subsets of $V(D)$ and*

- *$|A| = |B| = |C| = k$; and*
- *there are orderings $(a_1, \ldots, a_k)$, $(b_1, \ldots, b_k)$, and $(c_1, \ldots, c_k)$ of A, B, and C, respectively, such that*
  - *for all $i, j \in [k]$ we have $(a_i, b_j), (b_i, c_j) \in E(D)$, and*
  - *for all $i \in [k]$ we have $(c_i, a_i) \in E(T)$.*

When working with a k-triple (A, B, C), it is useful to have an easy way to refer the associated endpoints of the matching from C to A. Thus we sometimes refer to this matching as a bijective mapping M from C to A. That is, for $i \in [k]$ we have $\mathsf{M}(c_i) = a_i$ and $\mathsf{M}^{-1}(a_i) = c_i$. In addition, we extend graph-theoretical notation to k-triples in the following way. If $\mathcal{K}$ is a k-triple, we denote by $V(\mathcal{K})$ the set $A \cup B \cup C$ and by $E(\mathcal{K})$ the set of all arcs between pairs of vertices of $V(\mathcal{K})$. See Figure 1 for an example of a 4-triple.

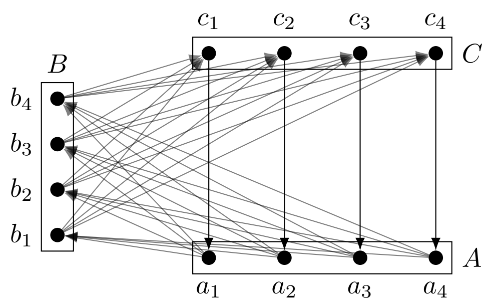


Figure 1 A 4-triple (A, B, C). We remark that nothing is known about the arcs inside A, B, or C nor about arcs between C and A other than the ones in the matching.

Parameterized complexity. A *parameterized problem* is a language $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, the value k is called the *parameter*. Such a problem is *fixed-parameter tractable* (FPT for short) if there is an algorithm that decides membership in L of an instance (x, k) in time $f(k) \cdot |x|^{\mathcal{O}(1)}$ for some computable

	$\mathcal{T}$	$\mathcal{S}$	$\mathcal{C}_h$
k-DDP	NP-c. [Theorem 1]	–	–
(k,c)-DDP	NP-c. for $c = k^{\varepsilon}$, $\varepsilon \in [0,1)$ [Theorem 3]	FPT by $k+h$ when $2c > k$ [Theorem 2]	W[1]-h. by $k+h$ with dpw$(T) \leq 2$ [Theorem 4]

Table 2 Summary of our main results.

function f, and it is in the class XP if there is an algorithm that decides membership in L of an instance (x,k) in time $f(k) \cdot |x|^{g(k)}$ for some computable functions f and g. Consult the monographs [17, 19, 21, 22, 39] for background on parameterized complexity.

3.2 Tournaments and relatives

As already said in the introduction, we denote by $\mathcal{T}$ and $\mathcal{S}$ the classes of tournaments and semicomplete digraphs, respectively. For every non-negative integer h, we consider three classes of digraphs. The class $\mathcal{S}_h$ contains every digraph D such that every $v \in V(D)$ has at most h non-neighbors. The class $\mathcal{C}_h$ contains every digraph whose vertex set can be partitioned into at most h cliques. The class $\mathcal{A}_h$ contains every digraph D with $\alpha(D) \leq h$.

Brook's Theorem [8] states that the vertex set of every graph G with maximum degree $\Delta(G) \leq k$ can be partitioned into $k+1$ color classes $X_1, \ldots, X_{k+1}$ such that each X_i induces an independent set in G. Let $D \in \mathcal{S}_h$. Applying Brook's Theorem to the complement $\overline{D}$ of D, we conclude that $V(D)$ can be partitioned into $h+1$ cliques and thus $\mathcal{S}_h \subseteq \mathcal{C}_{h+1}$. Since every independent set of a digraph intersects any clique in at most one vertex, it also holds that $\mathcal{C}_h \subseteq \mathcal{A}_h$. In Table 2 we give a summary of our main results.

3.3 Analyzing the Fomin-Pilipczuk counterexamples

Let us generalize the counterexamples given in [23] to other values of the congestion; this will show that the irrelevant vertex technique is not extensible to (k,c)-DDP for appropriate values of c. For each $n \geq 1$ and $c \geq 1$, we build an instance (T_n, K, c) of (k,c)-DDP in tournaments such that:

1. The digraph T_n has $4(n+1)$ vertices.
2. The vertex set of T_n is partitioned into two sets U, V of the same size, with $U = \{u_i \mid i \in [2n+2]\}$ and $V = \{v_i \mid i \in [2n+2]\}$.
3. The tournament $T_n[U]$ has a path $P_u = \langle u_1, \ldots, u_{2n+2} \rangle$ and all other arcs going in the opposite direction. Similarly, $T_n[V]$ has a path $P_v = \langle v_{2n+2}, \ldots, v_1 \rangle$ and all other arcs going in the opposite direction.
4. For every pair $(u_i, v_j) \in U \times V$, arc (u_i, v_j) exists if $i = j$, otherwise we have the arc (v_j, u_i).
5. Digraph T_n has an n-triple where $A = \{v_1, v_n\}$, $B = \{u_{n+3}, \ldots, u_{2n+2}\}$, and $C = \{u_1, \ldots, u_n\}$.
6. There are c requests (u_1, u_{2n+2}) and c requests (v_{2n+2}, v_1).

We refer to Figure 2 for the case $n = 2$. Observe that the requests can only be satisfied by using P_u and P_v, c times each. This statement follows by induction on n: it suffices to observe that u_2 is the unique vertex that may be the second vertex in the (u_1, u_{2n+2})-satisfying paths, while v_2 is the unique vertex that may be the penultimate vertex in the paths that

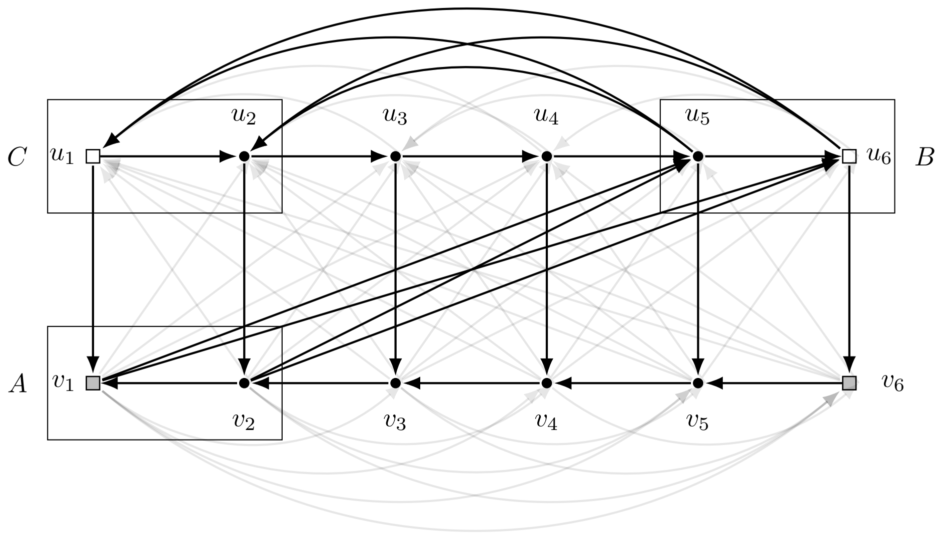


Figure 2 Counterexample for $n = 2$, where non-black vertices of the same color correspond to endpoints of a same request. The rectangles indicate the three sets that make up the n-triple.

satisfy (v_{2n+2}, v_1); at this point we can discard u_1, v_1 and repeat the analysis. We can further restrict ourselves to congestion values smaller than $k/2$. For example, if we add τ copies of T_n and add all arcs from the i-th copy to the j-th whenever $i < j$, then we can get $c = k/(2\tau)$; it is not hard to come up with other strategies that would yield $c = k/d$ for every $d \in [2, k]$. At this point, it is natural to think about the interval $(1, 2)$? Interestingly, things break down in this case. Suppose that, instead of having the same number of requests between the endpoints of P_u and P_v, we had two requests (u_1, u_{2n+2}), only *one* request for (v_{2n+2}, v_1), and congestion $c = 2$. We could use P_u twice and P_v once, but we could also do the following: begin by satisfying (v_{2n+2}, v_1) using P_v; now, add the paths $\langle u_1, v_1, u_6 \rangle$ and $\langle u_1, u_2, v_2, u_{2n+2} \rangle$ to satisfy the (u_1, u_{2n+2}) requests. With this, we have avoided the usage of every vertex in $\{u_3, \dots, u_{2n+1}\}$, making them *irrelevant* to the instance; we could also have avoided using vertices in V using extra steps.

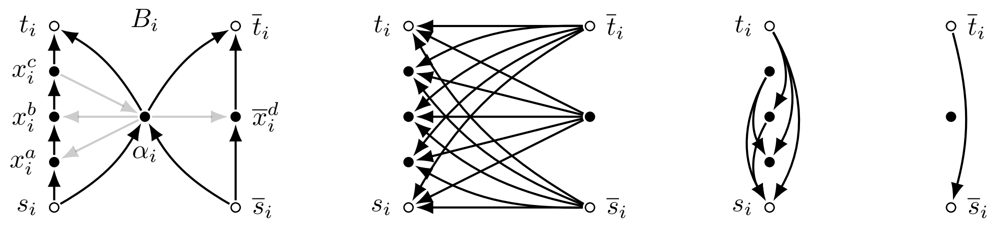
4 Hardness results

While it had been assumed for a long time that Directed Disjoint Paths was an NP-hard problem on tournaments [1, 14, 33, 41], we verified that there is a mistake in the proof given in [2]. We have reached out to its authors, which have confirmed the flaw in their proof in a personal communication [3]. In this section, we provide a direct NP-hardness proof for tournaments as well as other lower bounds the C_h superclasses.

4.1 Tournaments

Fortunately, as we show in the following theorem, k-DDP is indeed NP-hard on tournaments. We highlight that our reduction is completely different from the approach in [2]. In particular, we reduce from (3,1)-3-SAT, a variant of 3-SAT where each variable appears exactly four times – once negated and three times unnegated – and that was shown to be NP-complete in [18].

Construction. Let $(X, \mathcal{C})$ be the input instance to (3,1)-3-SAT, where X is the set of variables and $\mathcal{C}$ is the set of clauses, and (T, K) be the k-DDP instance we are going to build, with T denoting the digraph, and K denoting the set of requests. We begin by picking an arbitrary but fixed order $\langle x_1, \dots, x_n \rangle$ of X. Now, for each variable $x_i \in X$, add a copy of the *directed butterfly gadget* shown in Figure 3 to T, while (s_i, t_i) and $(\bar{s}_i, \bar{t}_i)$ are added to K; we denote this gadget by B_i. Intuitively, the paths between the white vertices of B_i must be contained in it and, since the gadget has only once center (namely, α_i in Figure 3), at most one of them may avoid the "wings", as shown in Observation 7. This allows us to encode the assignment of a variable and only satisfy clauses for which the appropriate literal is true, i.e., adding $\langle s_i, \alpha_i, t_i \rangle$ to the solution is equivalent to setting $x_i = \mathsf{true}$.



(a) Structural arcs. **(b)** Right-to-left arcs. **(c)** Top-to-bottom arcs.

Figure 3 Directed butterfly gadget for variable x_i, occurring negated in clause C_d and unnegated in clauses C_a, C_b, and C_c. White vertices represent terminals. The given orientations of the gray arcs will be important when talking about congested versions of the problem. Vertex α_i is the *center* of B_i, while the two disjoint paths from the s vertices to the t vertices that do not use α_i are its *wings*.

After building all n butterflies, add an arc from every $u \in B_j$ to every $v \in B_i$ whenever $j > i$. To encode our clauses, take each $C_a \in \mathcal{C}$, add new vertices p_a, q_a to T, the pair (p_a, q_a) to K, and the arc (q_a, p_a); arcs between each p_a and q_b, for $a \neq b$, can be added arbitrarily. Now, take $a \in \{1, \dots, m\}$ as an example, and suppose that the a-th clause of $\mathcal{C}$ is $C_a = (x_1 \vee x_2 \vee \bar{x}_3)$; in this case, we add arcs from p_a to the vertices x_1^a, x_2^a, and $\bar{x}_3^a$, and from these to q_a. For simplicity, the superscript j of each x_i^j and $\bar{x}_i^j$ is the same as the subscript of the corresponding clause C_j. At this point, the only missing arcs to ensure we have a tournament are between vertices of butterfly gadgets and clause gadgets; we add them such that p_a has no other outgoing arc and q_a has no other incoming arc. This concludes the construction of (T, K), with $K = \{(s_i, t_i), (\bar{s}_i, \bar{t}_i) \mid x_i \in X\} \cup \{(p_a, q_a) \mid C_a \in \mathcal{C}\}$. For the above example, our goal is to witness the satisfiability of C_a with a path $\langle p_a, y, q_a \rangle$, where $y \in \{x_1^a, x_2^a, \bar{x}_3^a\}$, which will have to be the case as $N^+(p_a) = N^-(q_a) = \{x_1^a, x_2^a, \bar{x}_3^a\}$.

▶ **Lemma 6.** *If there is a satisfying assignment π for $(X, \mathcal{C})$, there is a family of disjoint paths $\mathcal{P}$ satisfying (T, K).*

Proof. TOPROVE 0 ◀

▶ **Observation 7.** *Let B_i be one of the butterfly gadgets of (T, K). If (T, K) admits a solution $\mathcal{P}$, then (i) the paths that satisfy $(\bar{s}_i, \bar{t}_i)$ and (s_i, t_i) are internal to B_i, and (ii) at least one of the wings of B_i is entirely occupied by such a path.*

Proof. TOPROVE 1 ◀

▶ **Lemma 8.** *If there is a family of disjoint paths $\mathcal{P}$ satisfying (T, K), there is a satisfying assignment π for $(X, \mathcal{C})$.*

Proof. TOPROVE 2 ◀

▶ **Theorem 1.** Directed Disjoint Paths *on tournaments is* NP*-complete.*

Proof. TOPROVE 3 ◀

4.2 Tournaments with congestion

As our main focus for this work is on congested variants of Directed Disjoint Paths, it is natural to extend the proof of Theorem 1 to these variants as well. In particular, we are interested in the case where the congestion $c = |K|/d$.

4.2.1 Restricted K

We first consider a variant that we call Restricted (k, c)-DDP, where we are additionally forbidden from using vertices of K as inner vertices of other paths; that is, if $(u, v) \in K$, then a path P has $u, v \in P$ if and only if u is the first vertex of P and v is the last. Note that if $c = 1$, this is precisely what we have in the standard k-DDP problem.
Meta-construction. We reduce from the instance (T, K) of k-DDP built in the proof of Theorem 1 to construct the instance (T', K', c) of Restricted (k', c)-DDP. Let $\langle B_1, \ldots, B_n \rangle$ be the ordering of the butterfly gadgets, that is, B_j is complete to B_i for all $j > i$. We perform the following modification to T: for each $i \in [n-1]$, reverse the arc $(x^a_{i+1}, \overline{x}^d_i)$, where x^a_{i+1} is the unique out-neighbor of s_{i+1} in the left wing of B_{i+1}. To conclude the construction of T', we add two vertices s^*, t^* and arcs such that x^a_1 is the unique out-neighbor of s^* and $\overline{x}^d_n$ is the unique in-neighbor of t^*. Finally, we add $c - 1$ requests (s^*, t^*) and one request (t^*, s^*) to K, thus obtaining K'. Note that c is *almost* arbitrary, as it can be set to a constant or whichever *fraction* of $|K'|$ that we wish. Not every element of $o(|K'|)$, however, is assignable to c; for example, $c = |K'| - \log |K'|$ is not a viable candidate, as this would imply that $|K'|$ is exponential in $|K|$ since $|K'| = |K| + c$. We illustrate the resulting tournament T' in Figure 4.

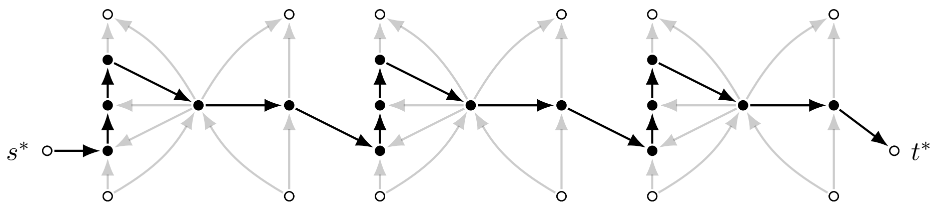


Figure 4 Queuing of the butterfly gadgets performed for the construction of T'. Gray arcs do not participate in the critical path of T', while black arcs do. Again, within the butterfly gadgets, we add the missing arcs from top-to-bottom, right-to-left. All other arcs are also right-to-left.

Observe that the interior vertices of the butterflies' wings are the only vertices of T' that do not participate in any element of K'. Moreover, by construction, they all participate in the unique s^*-t^* path of $T' \setminus V(K' \setminus \{(s^*, t^*)\})$ in the same order specified by $\langle B_1, \ldots, B_n \rangle$. We call this path the *critical path* of T', and in any solution to (T', K', c), all of its inner vertices participate in at least $c - 1$ paths; cf. the black arcs in Figure 4.

▶ **Theorem 9.** Restricted (k', c)-DDP *remains* NP*-complete even when restricted to tournaments and when $c = \varepsilon k$ for every $\varepsilon \in (0, 1)$.*

Proof. TOPROVE 4 ◀

4.2.2 Unrestricted K

While we are unable to prove a statement as strong as Theorem 9 for (k,c)-DDP, we are able to (conditionally) rule out polynomial-time algorithms for some choices of c; in particular, if $c \leq |K|^{\varepsilon}$ for $\varepsilon \in [0,1)$ we can show such a lower bound.

▶ **Theorem 3.** (k,c)-DDP *remains* NP*-complete on tournaments even restricted to instances satisfying* $c = k^{\varepsilon}$*, for every* $\varepsilon \in [0,1)$.

Proof. TOPROVE 5 ◀

We would like to point out that overcoming the $|K|^{\varepsilon}$ barrier has proved extremely challenging for us, and seems to require a very different approach to the one we employed so far. In particular, we can rearrange the butterfly gadgets to be *stacked* instead of queued, i.e., we glue the terminals of two butterflies, avoiding the need of too many paths to saturate them. This, however, is not enough, as we are unable to saturate the clause gadgets with few paths, instead always requiring $\mathcal{O}(|\mathcal{C}|)$ requests, as done in Theorem 3. Interestingly, we can prove a weak NP-hardness result for larger values of the congestion, namely if $c = |K| \log^{-\varepsilon} |K|$.

▶ **Theorem 10.** (k,c)-DDP *is weakly* NP*-hard on tournaments if the congestion is of the form* $|K| \log^{-\varepsilon} |K|$*, for every* $\varepsilon > 0$.

Proof. TOPROVE 6 ◀

4.2.3 NP-hardness for graphs in $\mathcal{C}_2$

Before proceeding, recall that $G \in \mathcal{C}_h$ if and only if the vertices of its underlying graph can be partitioned by at most h cliques, i.e., its complement has chromatic number at most h. Our construction is similar to the one employed on Theorem 9 and yields an equivalent statement, however there are some key differences, so we do not use it as directly as in Theorem 3. As such, we describe our reduction directly from (3,1)-3-SAT. Intuitively, we will use the flexibility of some non-arcs to extend the critical path with some sinks of the clause requests; at the same time, we can replace the $\mathcal{O}(|T|)$ sources with a single vertex.
Construction Let once again $(X, \mathcal{C})$ be the given (3,1)-3-SAT instance and take an arbitrary ordering $\langle C_1, \ldots, C_m \rangle$ of $\mathcal{C}$. We first show how to construct an instance (G, K, c) of Directed Congested Disjoint Paths with $c = |\mathcal{C}|$, and subsequently show how to extend it to so c is an arbitrary ratio of K. Initially, add the queued structure of butterflies shown in Figure 4 to G and the corresponding requests to K, but now we perform some modifications to it: (i) take the unique out-neighbor v of s^* and subdivide the arc (s^*, v) exactly $|\mathcal{C}|$ times, assigning to each generated q_a a unique clause C_a; (ii) for each C_a, add an arc from x_i^a to q_a if $x_i \in C_a$ and from $\overline{x}_i^a$ to q_a if $\overline{x}_i \in C_a$; (iii) for each pair $a, b \in [m]$, add the arc (q_b, q_a) if and only if $b > a$; (iv) now, add a unique vertex p^* to G, which has all vertices in the butterfly gadgets as out-neighbors and t^* as unique in-neighbor. Finally, add the arc and the request (t^*, s^*), $c - 1$ requests of the form (s^*, t^*) and, for each $C_a \in \mathcal{C}$, add the request (p^*, q_a) to K. We exemplify the above changes in Figure 5, where the rectangles with the thick vertical line as one of the sides correspond to the cliques partitioning G.

▶ **Theorem 11.** (k,c)-DDP *is* NP*-complete when restricted to graphs of* $\mathcal{C}_2$.

Proof. TOPROVE 7 ◀

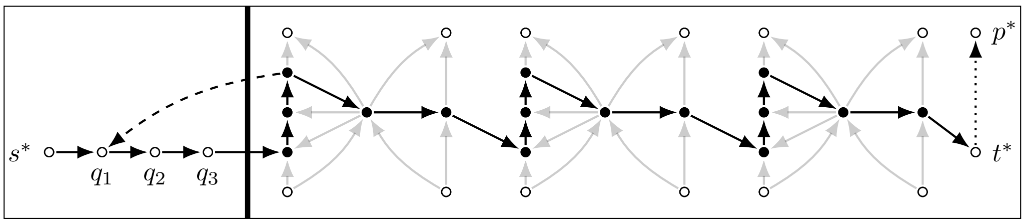


Figure 5 Queuing of the butterfly gadgets performed for the construction of G with the extended critical path on the left of the figure. Each of the two smaller rectangles are cliques in the host graph. Within them, missing arcs are from top-to-bottom, right-to-left. Across the thick vertical line, the only arcs are from vertices of the form $\ell_i^a \in B_i$ to q_a, which exists if and only if $\ell_i \in C_a$; these are shown by the dashed arc from the first butterfly to q_1. The dotted arc is not considered part of the critical path. Arc $(t^*, s^*) \in E(G)$ is omitted for simplicity.

It is also possible to obtain similar proofs when setting $c < |\mathcal{C}|$. The proof follows the same steps as the one of Theorem 11, but requires splitting p^* into sufficiently many vertices to accommodate the fewer requests that can touch p^*. For example, if we want $c = |\mathcal{C}|/2$ and this is an integer value, then we replace p^* with p_1^*, p_2^* and evenly divide the requests of the form (p^*, q_a) between them. We omit the more technical details for brevity.

▶ **Theorem 12.** (k, c)-DDP *is* NP*-complete when restricted to graphs of* $\mathcal{C}_2$ *for every constant congestion value* $c \geq 1$*.*

We conclude our polynomial-time lower bounds with the following theorem.

▶ **Theorem 13.** (k, c)-DDP *is* NP*-complete when restricted to graphs of* $\mathcal{C}_2$ *even if the congestion satisfies* $c = |K|/d$ *for* $d > 1$*.*

Proof. TOPROVE 8 ◀

4.3 W[1]-hardness on $k + h$ on graphs of bounded directed pathwidth

While our previous lower bounds were all in the polynomial-time world, a natural question is what happens if the numbers $k = |K|$ of requests and h of partitioning cliques are now taken as parameters. In [14], Chudnosvky, Scott, and Seymour showed a $|V(G)|^{\mathcal{O}((hk)^5)}$-time algorithm for Directed Disjoint Path on $\mathcal{C}_h$. We show that their XP algorithm cannot be significantly improved by proving that Directed Congested Disjoint Paths remains W[1]-hard when parameterized by $k + h$ on graphs of bounded directed pathwidth. Our reduction is heavily inspired by, but requires significant increments to, the proof of Slivkins [41] of the W[1]-hardness of Edge Directed Disjoint Paths on DAGs, where the source problem was Clique parameterized by the solution size; to simplify a bit our arguments, we opt to reduce from Multicolored Clique parameterized by the solution size. As in Section 4.1, we first deal with the completely disjoint version of the problem and then show how to extend to the congested case.

Construction. Let $(G, \mathcal{V})$ be the input instance to Multicolored Clique where $\mathcal{V} = \{V_1, \dots, V_q\}$ are the color classes and (D, K) denotes the Directed Disjoint Paths instance we are going to build. For simplicity, assume that every $V_i \in \mathcal{V}$ has the same size n. The vertices of D are partitioned in a $q \times (2 + n)$ matrix-like fashion, with $D_{i,j} \subseteq V(D)$ corresponding to the j-th vertex of $V_i \in \mathcal{V}$ and $D_{i,0}, D_{i,n+1}$ with no correspondence. For every $i \in [q]$, let us define and add the following sets of vertices to G, where each $D_{i,j}$ is a set of $2q + 3$ vertices that will be specified later:

$$\begin{aligned} D_i = &\{\alpha_{i,s}, \alpha_{i,t}, a_{i,s}, a_{i,t}, b_{i,s}, b_{i,t}\} \\ &\cup \{g^1_{i,s}, g^1_{i,t}, g^2_{i,s}, g^2_{i,t}\} \\ &\cup \bigcup_{j \in \{0\} \cup [n+1]} D_{i,j}. \end{aligned}$$

For this proof, we will stick to the convention that vertices with an s subscript or t subscript are sources or targets of a request of K, respectively. In particular, K now contains the pairs $(\alpha_{i,s}, \alpha_{i,t})$, $(a_{i,s}, a_{i,t})$, $(b_{i,s}, b_{i,t})$, $(g^1_{i,s}, g^1_{i,t})$, and $(g^2_{i,s}, g^2_{i,t})$ for every $i \in [q]$. Each $D_{i,j}$ is further broken down into four tournaments: $D^\alpha_{i,j}, D^a_{i,j}, D^\beta_{i,j}, D^b_{i,j}$. We set $D^\alpha_{i,j} = \{\alpha_{i,j}\}$, $D^\beta_{i,j} = \{\beta^1_{i,j}, \beta^2_{i,j}\}$, and, for $x \in \{a, b\}$, we set $D^x_{i,j} = \{x^0_{i,j}, x^1_{i,j}, \dots, x^q_{i,j}\}$; intuitively, $x^\ell_{i,j}$ is the only vertex that will connect $D_{i,j}$ to vertices of D_ℓ, while $x^0_{i,j}$ is used to synchronize the internal behavior of D_i. Let us divide our construction in steps:

(i) The arcs of $D_{i,j}$ including $D_{i,0}$ and $D_{i,n}$ are built as follows: each $D^x_{i,j}$, $x \in \{\alpha, a, \beta, b\}$ has exactly one Hamiltonian path – known as the *x-path* of $D_{i,j}$ – and all other arcs are in the opposite direction; i.e., if $\langle x^0_{i,j}, \dots, x^\ell_{i,j} \rangle$ is the Hamiltonian path of $D^x_{i,j}$, then $(x^z_{i,j}, x^y_{i,j}) \in E(D)$ for every non-consecutive pair z, y with $z > y$; w.l.o.g. we assume the visitation order of the Hamiltonian path is the same as the one we specified in the definition of $D^x_{i,j}$. The only other arcs incident to two vertices of $D_{i,j}$ are a perfect matching from $D^a_{i,j}$ to $D^b_{i,j}$, such that $a^\ell_{i,j}$ is matched to $b^\ell_{i,j}$ for $\ell \in [n]$. We present an example in Figure 6.

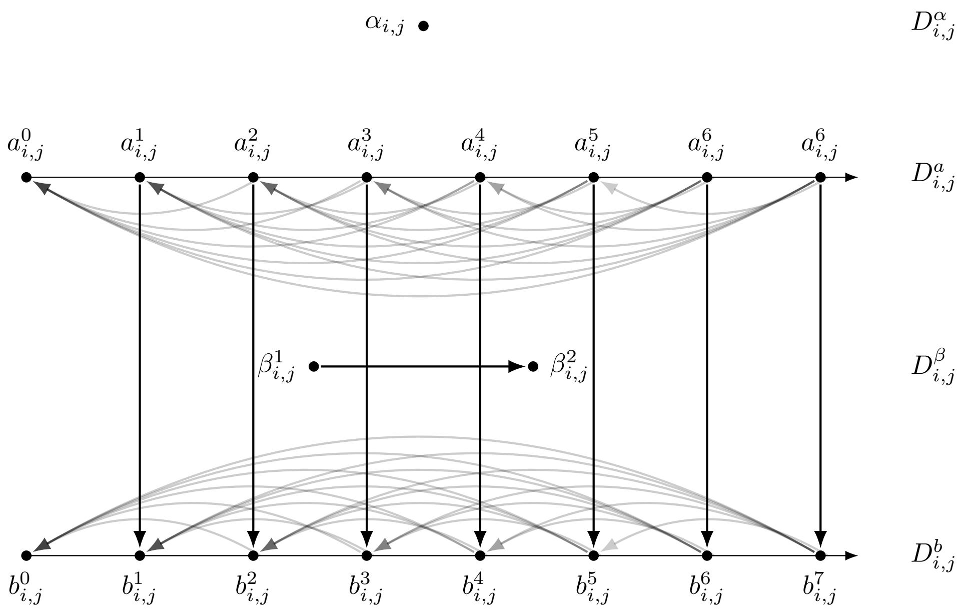


Figure 6 Gadget $D_{i,j}$ and the arcs built in Step (i) of the reduction. Grey arcs are only used to force that their incident vertices form a tournament. Horizontal arrows denote the three x-paths of $D_{i,j}$ with at least one edge, i.e., $x \in \{a, b, \beta\}$.

Intuitively, the a- and b-paths will be used in the encoding of the MULTICOLORED CLIQUE instance, while the α- and β-paths are used to synchronize the decisions performed in the a- and b-paths.

(ii) The next step in the construction is the connection of the different elements of D_i. Our goal is to add arcs so that exactly one index $j \in [n]$ has both $D^a_{i,j}$ and $D^b_{i,j}$ unoccupied by paths satisfying requests internal to gadget D_i, while all other vertices of D^a_i and D^i_b are occupied; that is, exactly one vertex of each color class, corresponding to that index j, may be picked in the Multicolored Clique instance. This can be accomplished as follows:

a. For each $j \in [n+1]$ and $x \in \{\alpha, a, \beta, b\}$, take the last vertex of the x-path of $D^x_{i,j-1}$ and add an arc to the first vertex of the x-path of $D^x_{i,j}$, with all other arcs going from $D^x_{i,j}$ to $D^x_{i,j-1}$. Doing this, $D^x_i = D_{i,0} \cup \bigcup_{j\in[n]} D_{i,j}$ is a tournament with a unique Hamiltonian path obtained by stitching the x-paths of $D^x_{i,j}$'s to each other.

b. Add the arcs $(\alpha_{i,s}, \alpha_{i,0})$, $(a_{i,s}, a^0_{i,0})$, $(b_{i,s}, b^0_{i,0})$, the arcs $(a^q_{i,n+1}, \alpha_{i,t})$, $(\beta^2_{i,n+1}, b_{i,t})$, $(b^q_{i,n+1}, a_{i,t})$, and all remaining right-to-left arcs so that the vertices in each long horizontal arrow in Figure 7 form a tournament; e.g. $a_{i,s}$ has $D^a_i \setminus \{a^0_{i,0}\}$ as its in-neighbors while $\alpha_{i,t}$ has $D^a_i \setminus \{a^q_{i,n+1}\}$ as its out-neighbors.

c. For each $j \in \{0\} \cup [n-1]$, we add the *forward jumping arcs* $(\alpha_{i,j}, a^0_{i,j+2})$, $(a^q_{i,j}, b^0_{i,j+2})$, and $(b^q_{i,j}, \beta^1_{i,j})$.

d. For each $j \in [n]$, we add the *backward jumping arcs* $(\alpha_{i,j}, \beta^2_{i,j-1})$ and $(a^0_{i,j}, \beta^1_{i,j-1})$

e. Finally, for each $j \in \{0\} \cup [n]$, we add every arc from $g^1_{i,s}$ to D^a_i, from $g^2_{i,s}$ to D^α_i, from D^β_i to $g^1_{i,t}$ and $g^2_{i,t}$, and between the latter two arbitrarily. Consequently, $\{g^1_{i,s}\} \cup D^a_i$, $\{g^2_{i,s}\} \cup D^\alpha_i$, and $\{g^1_{i,t}, g^2_{i,t}\} \cup D^\beta_i$ are tournaments.

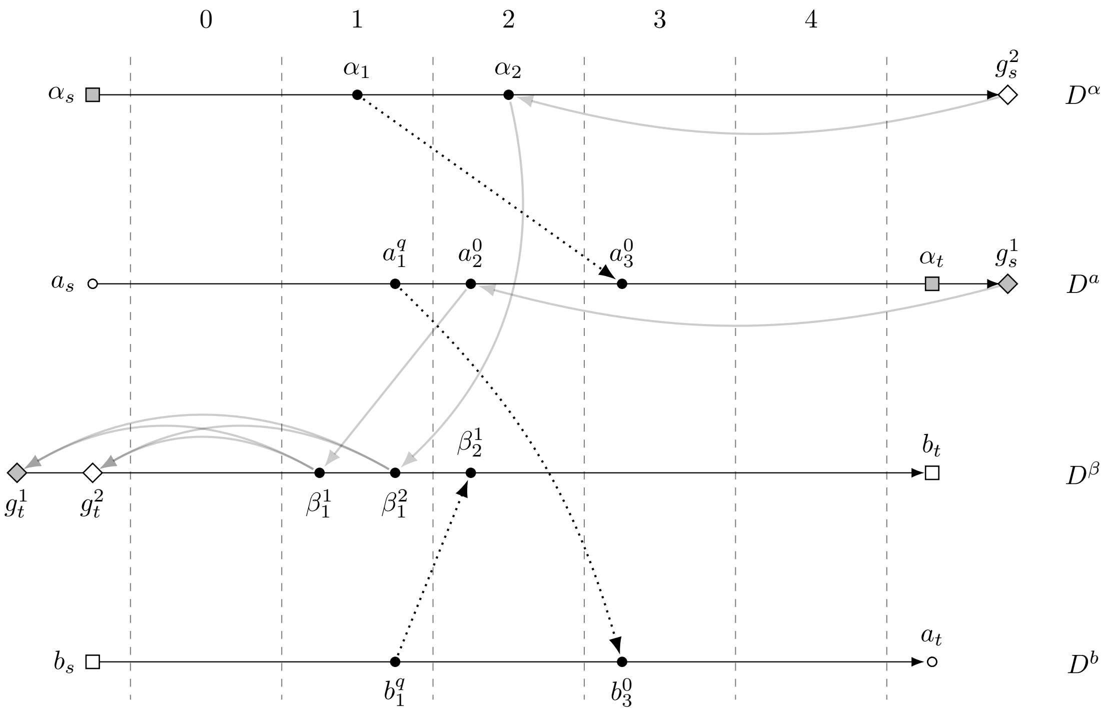


Figure 7 Gadget D_i and the arcs built in Step (ii) of the reduction. Dotted arcs correspond to the forward jumping arcs of Step (ii.c), while the gray arcs represent the arcs of Steps (ii.d) and (ii.e). We do not illustrate the arcs added in Steps (ii.a), (ii.b), (ii.c) and omit the i in the subscripts to improve readability. Differently shaped/colored vertices correspond to different requests of K.

(iii) While Step (ii) essentially encodes that only one vertex of each V_i may be picked, we now must make sure that they collectively indeed form a tournament of G. Our final set of vertices are obtained by adding the following requests and their corresponding vertices $(\delta_{i,\ell,s}, \delta_{i,\ell,t})$, where $i \in [q]$ and $\ell \in [q] \setminus [i-1]$; this implies that $|K| = 5q + \binom{q}{2} + q$. We proceed as follows, which is illustrated in Figure 8.

a. For each triple $i, \ell \in [q], j \in [n]$ with $i \leq \ell$: add the arcs $(\delta_{i,\ell,s}, a^{\ell}_{i,j}), (b^{i}_{\ell,j}, \delta_{i,\ell,t})$.

b. Finally, let $u \in V_i$ and $v \in V_\ell$ be adjacent vertices of G with $i < \ell$: add the arc $(b^{\ell}_{i,u}, a^{i}_{\ell,v})$.

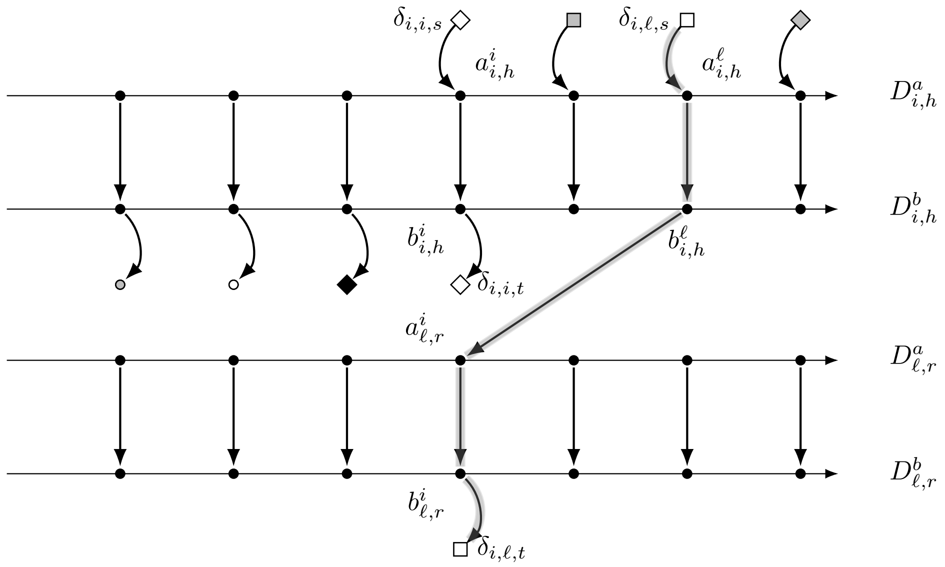


Figure 8 Partial representation of the gadgets $D_{i,h}$ and $D_{\ell,r}$ with the thicker arcs being those added in Step (iii) of the reduction. The grayed path encodes one edge of the multicolored clique of G. Differently shaped/colored vertices correspond to different requests of K.

Note that, if we can guarantee that each $i \in [q]$ has exactly one $D_{i,j}$ unoccupied by the paths discussed in Step (ii), then the paths used to satisfy each $\delta_{i,\ell}$ pair must, necessarily, go through arcs of D that encode adjacency in G, and all such arcs must be incident to the same vertex of V_i. We now formalize our intuition in the lemmas required to prove our main result for this section.

▶ **Observation 14.** *Instance (D, K) of* Directed Disjoint Paths *has $|K| = 6q + \binom{q}{2}$, D can be partitioned into $6q + 2\binom{q}{2}$ tournaments and has directed pathwidth equal to 2.*

Proof. TOPROVE 9 ◀

We could modify our reduction to get a smaller number of cliques, but breaking the $\Theta(q^2)$ bound for either $|K|$ or the number of cliques seems very hard with this approach. As such, we do not further trim down either parameter, opting for the current version for ease of exposition.

▶ **Lemma 15.** *If there is a solution Q to the* Multicolored Clique *instance $(G, \mathcal{V})$, then the* Directed Disjoint Path *instance (D, K) admits a solution $\mathcal{P}$.*

Proof. TOPROVE 10 ◀

The converse direction, as usual, requires additional care in its proof. To this end, we first show that minimal solutions in (D, K) must adhere to a specific format, i.e., the jumps between the x-paths of each D_i happen *synchronously* as in **??**: all forward jumps leave a gap at precisely one index j of each D_i, and this happens due to how the backward jumps were defined.

▶ **Lemma 16.** *Let $\mathcal{P}$ be a solution of (D, K) where every $P \in \mathcal{P}$ is induced (i.e., P is minimal), $D_i \in D$, $P_{i,i} \in \mathcal{P}$ be the path satisfying $(\delta_{i,i,s}, \delta_{i,i,t}) \in K$, and $\mathcal{I}_i$ be the set of paths satisfying internal requests of D_i. It holds that (1) $V(\mathcal{I}_i) \subseteq D_i$ and there is some $u \in [n]$ such that: (2) all forward jumps performed by paths of $\mathcal{I}_i$ originate in $D_{i,u-1}$, and (3) all backward jumps go from $D_{i,u} \setminus D^{\beta}_{i,u-1}$ to $D^{\beta}_{i,u-1}$. Moreover, (4) for every $v \in \{0, \dots, n+1\} \setminus \{u\}$, we have $D^a_{i,v} \cup D^b_{i,v} \subseteq V(\mathcal{I}_i)$ and $(D^a_{i,u} \cup D^b_{i,u}) \cap V(\mathcal{I}_i) = \{a^0_{i,u}, b^0_{i,u}\}$.*

Proof. TOPROVE 11 ◀

Property (4) of Lemma 16 is the key ingredient of our proof and the main complication to generalize Slivkins' result. In particular, we need to fully use D^a_i and D^b_i to avoid cheating by the δ paths: without this guarantee, we could have a δ path hitting an a vertex at some index u, going to an a vertex of index v using an arc added to force the appearance of a tournament, moving to its matched b vertex and then moving on to any D_j, completely breaking down the desired behavior. With these bizarre paths safely ruled out, proving the converse becomes quite direct, as we show in the following lemma.

▶ **Lemma 17.** *If (D, K) admits a solution $\mathcal{P}$, then $(G, \mathcal{V})$ contains a multicolored clique Q of appropriate size.*

Proof. TOPROVE 12 ◀

▶ **Theorem 4.** *The k-DDP problem on $\mathcal{C}_h$ is $\mathsf{W}[1]$-hard when parameterized by $k + h$, even if restricted to input digraphs of directed pathwidth two.*

Proof. TOPROVE 13 ◀

4.3.1 Into the congestionverse

We employ a trick very similar to the one used in Subsubsection 4.2.3. Essentially, we introduce a path that is *almost* Hamiltonian, missing only the δ vertices of Theorem 4. It is also possible to include them in this new path, but for simplicity's sake we do not. Formally, we proceed as follows: let (D', K') be the instance of Directed Disjoint Paths built in the proof of Theorem 4 and $c > 1$ be an integer; to obtain (D, K, c), we add $c - 1$ requests of the form (z_s, z_t) to K' and the corresponding vertices to D'. Now, add the arcs $(z_s, g^1_{q,t})$ and $(g^2_{1,s}, z_t)$ to D. Then, for each $i \in [q]$, we add the arcs $(b_{i,t}, b_{i,s})$, $(a_{i,t}, a_{i,s})$, $(g^1_{i,s}, \alpha_{i,s})$ and, if $i > 1$, $(g^2_{i,s}, g^1_{i-1,t})$. This completes the construction of (D, K); note that $|K| = |K'| + (c - 1)$ and that the directed pathwidth of D remains 2.

▶ **Observation 18.** *Let $\mathcal{P}$ be a solution to (D, K). There exists a unique path P_z that can satisfy (z_s, z_t) and, for every $i \in [q]$, this path is obtained by concatenating the β-, b-, a- and α-paths, in this order.*

Proof. TOPROVE 14 ◀

As in the proof of Theorem 12, we are now free to choose c to get whichever fraction of K we desire, and so we obtain the following theorem.

▶ **Theorem 19.** *(k, c)-DDP jointly parameterized by the number of requests $|K|$ and the minimum clique cover of the host graph is* W*[1]-hard even if the input instance has directed pathwidth 2 and the congestion c is a fraction of, but not equal to, $|K|$.*

5 Algorithmic results

In this section we prove our main algorithmic result and discuss its applications.

▶ **Theorem 20.** *For all integers $k \geq 1$ and $h \geq 0$, there is a function $f(k,h)$ such that every instance of (k,c)-DDP with $2c > k$ on an h-semicomplete digraph D containing a $f(k,h)$-triple has an irrelevant vertex which can be found in $\mathcal{O}(f^2(k) \cdot n^2)$ time. In particular, $f(k,0) = \mathcal{O}(2^{k \cdot \log k})$.*

Fomin and Pilipczuk [23, Theorem 6.3] showed that Directed Edge-Disjoint Paths is FPT on tournaments parameterized by the number of paths. They solve the problem with a win-win approach. First they show that the problem is FPT parameterized by the directed pathwidth $\mathsf{dpw}(T)$ of the input semicomplete digraph T plus the number k of paths. Then, through a series of results and constructions, they show that there is a computable function $g(k)$ such that every semicomplete digraph T with $\mathsf{dpw}(T) \geq g(k)$ contains a k-triple, and that every such triple contains a vertex v that is irrelevant for the given instance. This second proof essentially shows that only the instances with $\mathsf{dpw}(T)$ bounded by some function of k need to be solved.

Although they leave open whether Directed Disjoint Paths is FPT on semicomplete digraphs, they mention that an FPT dynamic programming algorithm, parameterized by the number of paths, can be done for Directed Disjoint Paths to obtain a result analogous to [23, Theorem 6.3].

Another approach to obtain such an FPT algorithm relies on *directed cliquewidth*, which is bounded from above by $\mathsf{dpw}(T) + 2$ on every semicomplete digraph T [23, Lemma 2.14]. Then one can apply [23, Theorem 2.16] which states that, given an MSO_1 formula ϕ and a semicomplete digraph T, there is an algorithm which checks whether ϕ is satisfied by T in FPT time parameterized by the order of ϕ and $\mathsf{dpw}(T)$. This implies the existence of an FPT algorithm on semicomplete digraphs for Directed Disjoint Paths parameterized by the number of paths, since this problem can be modeled by MSO_1 (see, for instance, [28, Proposition 4.7]). In any case, the following is obtained.

▶ **Proposition 21** (Fomin and Pilipczuk [23]). Directed Disjoint Paths *on semicomplete digraphs is* FPT *parameterized by the number of paths and the directed pathwidth of the input semicomplete digraph.*

Applying classical techniques, one can show easily show that Proposition 21 also implies an FPT algorithm for (k,c)-DDP. Indeed, given an instance of this problem on a semicomplete digraph T, it suffices to add copies $v_2, \ldots, v_c$ of each $v \in V(T)$ with the same in- and out-neighborhood as v. Then, we add arcs among vertices in $\{v, v_2, \ldots, v_c\}$ in order to ensure that this set induces an acyclic tournament. This procedure easily implies that the directed pathwidth of T increases by at most a factor of c. Hence we can apply Proposition 21 to solve k-DDP in the newly constructed digraph and transport any solution, whether positive or negative, to the original instance in T. This implies that (k,c)-DDP is also FPT parameterized by k and the directed pathwidth of the input digraph.

When applying their irrelevant vertex rule, Fomin and Pilipczuk [23] use a series of results to construct in FPT time a triple of large order on semicomplete digraphs of sufficiently large directed pathwidth. Namely, given a semicomplete digraph T and an integer k, they first apply [23, Theorem 4.12] to either produce a directed path decomposition of width bounded by some computable function $g(k)$ or find one of the two certificates that $\mathsf{dpw}(T) > k$: a *degree tangle* or a *matching tangle*. Then, they show how to produce a *short jungle* from any of those two objects in [23, Lemmas 3.9 and 3.12]. We remark that all these constructions run in polynomial time. They argue that although the proof of how to extract triples from short jungles by Fradkin and Seymour [26] is not explicitly algorithmic, it is easy to extract an algorithm from it. In any case, the proof of [23, Theorem 9.9] shows how to circumvent the lack of an explicit algorithmic result to obtain triples from jungles, essentially obtaining the following result one way or another when applying the other results mentioned in this paragraph.

▶ **Proposition 22** (Fomin and Pilipzcuk [23]). *Let T be a semicomplete digraph. There is a computable function $g(t)$ and an* FPT *algorithm parameterized by t that either outputs a directed path decomposition of T with width at most $g(t)$ or finds a t-triple in T.*

In light of Propositions 21 and 22, it is now easy to apply Theorem 20 to obtain an FPT algorithm for (k, c)-DDP when $2c > k$. We restate Theorem 2 for convenience.

▶ **Theorem 2.** *(k, c)-DDP on semicomplete digraphs is* FPT *parameterized by k restricted to instances satisfying $c > k/2$.*

Proof. TOPROVE 15 ◀

5.1 Irrelevant vertices in k-triples

This section is dedicated to the proof of Theorem 20.

We first prove the version of Theorem 20 restricted to semicomplete digraphs, and at the very end of this section we explain which are the (easy) changes to be made to the proof so that it holds for any $h \geq 1$. We abbreviate $f(k, 0)$ by $f(k)$ and assume that an $f(k)$-triple is given. The choice of $f(k)$ is discussed later. Let $\mathcal{I} = (T, K, c)$ be an instance of (k, c)-DDP on a semicomplete digraph T with $2c > |K| = k$. For the remainder of this section, as well as in Subsection 5.2, Subsection 5.3, and Subsection 5.4, we assume that $\mathcal{K}$ is an $f(k)$-triple (A, B, C) of T and that $\mathcal{P}$ is a set of paths satisfying the following property.

▶ **Property 23.** *$\mathcal{P}$ is a solution $\{P_1, \ldots, P_k\}$ for $\mathcal{I}$ minimizing $\sum_{i \in [k]} |V(P_i)|$.*

Since congestion is allowed, any vertex is counted in the summation as many times as it is used by a path. This per-path measure is necessary as we often shortcut (we formally define shortcuts later in this section) a path $P \in \mathcal{P}$ through a vertex that is used by other paths in the collection $\mathcal{P}$.

To avoid some technicalities, here we make two assumptions about $\mathcal{K}$. First, by discarding at most $2k$ vertices of each set A, B, and C, we assume that the terminals of $\mathcal{I}$ do not appear in $V(\mathcal{K})$. Second, by deleting parallel arcs of T if necessary, we assume that there are no arcs in $E(\mathcal{K})$ from C to B and no arcs from B to A. The removal of those arcs poses no issue for the irrelevant vertex argument since any irrelevant vertex of $T' \subseteq T$ is also irrelevant in T.

The set B plays as distinguished role in this proof since it concentrates the heads of all arcs of $\mathcal{K}$ between A and B and the tails of all arcs of $\mathcal{K}$ between B and C. The goal is to show that there is room to reroute paths in $\mathcal{K}$ in order to show that if $\mathcal{I}$ is a yes-instance, then there is a vertex $b \in B$ and a solution $\mathcal{Q}$ for $\mathcal{I}$ such that every $Q \in \mathcal{Q}$ accessing b does so from a vertex in A and to a vertex in C.

We follow the blueprint of the proof for Directed Edge Disjoint Paths on semicomplete digraphs by Fomin and Pilipzcuk [23]. A fundamental property in their case is that any vertex with out-degree (resp. in-degree) at least $k + 1$ in the given triple has at least one arc leaving (resp. entering) it that is not used by any solution $\mathcal{Q}$ minimizing the sum of the lengths of its paths. This property allows them to prove that every such solution cannot use more than two arcs of the matching from C to A. This, in turn, is used to show that every path in $\mathcal{Q}$ uses at most $2k+4$ vertices from A and from C, and at most $4k$ vertices of B. The hard part of their proof is about how to apply these results to prove that an irrelevant vertex is guaranteed to exist in a sufficiently large triple, and how to find it in polynomial time.

In our case, the existence of a vertex $v \in V(\mathcal{K})$ with large out-degree or in-degree in the triple does *not* guarantee that we can find a vertex in $N^+(v) \cup N^-(v)$ that is not used by a path of $\mathcal{P}$. In fact, in many places of our proof we do not guarantee that at all, since we produce shortcuts for paths in $\mathcal{P}$ through vertices that have been used by other paths of $\mathcal{P}$ while but *at most* $c-1$ of them, hence being careful not to exceed the allowed congestion on each vertex. The fundamental property that we use comes from the assumption that $2c > k$: if u, v are both used by c paths of $\mathcal{P}$, then by the pigeonhole principle there is a path $P \in \mathcal{P}$ that uses *both* u and v. The need to rely on this property is justified by the counterexample provided in Subsection 3.3, which shows that no irrelevant vertex is guaranteed to exist in arbitrarily large triples whenever $2c \leq k$, and makes our analysis significantly harder than the one in [23]. For example, in order to show that a bounded number of vertices of A and C are used by paths of $\mathcal{P}$ (Lemma 26), we first need to show that some vertices of B can be used for shortcuts (Lemma 24) and these two proofs are already much harder than their counterparts in [23].

In addition to the definitions introduced in the beginning of this section, we adopt the following notation. We denote by M the arcs of the matching from C to A and, as discussed right below Definition 5, we may also refer to M as a bijective mapping from C to A according to the arcs of the matching defining $\mathcal{K}$. In addition, for $C' \subseteq C$ (resp. $A' \subseteq A$) we call the set $\mathsf{M}(C') = \{\mathsf{M}(v) \mid v \in C'\}$ (resp. $\mathsf{M}^{-1}(A') = \{\mathsf{M}^{-1}(u) \mid u \in A'\}$) the *mirror* of C' (resp. of A') in $\mathcal{K}$.

For $P_i \in \mathcal{P}$ we denote by $\prec_i$ the order in which $V(P_i)$ appears in P_i. A *shortcut* for P_i is a walk R such that there is some subpath $R' \subseteq P_i$ where $|V(R)| < |V(R')|$ and R starts and ends in the same vertices as R'. See Figure 9 for examples of shortcuts. We may refer to a shortcut R as the sequence $v_1 \to v_2 \to \ldots \to v_m$ of the vertices of R as they appear in the walk. The existence of a shortcut for any path in $\mathcal{P}$ respecting the congestion measure contradicts Property 23, and the goal of the first part of the proof of Theorem 20 is to exploit this fact to prove useful properties on how the paths of $\mathcal{P}$ can intersect an $f(k,h)$-triple. To avoid repetition, we refrain from stating that these shortcuts lead to contradictions in the proofs of this section. Note that simply replacing R' with R may not lead to a minimal solution: there could be forward arcs that further shorten the resulting object, or cycles could be introduced, depending on the visiting order of the involved vertices. For example, in Figure 9, it could be the case that the vertex of C appeared in the depicted path *after* the vertex of A, but with the addition of the blue segment we now have a cycle.

For every $v \in V(\mathcal{K})$, we assign to v a list of indices $\mathsf{L}(v) \subseteq 2^{[k]}$ representing which paths of $\mathcal{P}$ are using v. Thus $|\mathsf{L}(v)| \leq c$ holds for any vertex v since $\mathcal{P}$ is a solution for $\mathcal{I}$. For $i \in [k]$, we say that v is *i-free* if $i \in \mathsf{L}(v)$ or if $|\mathsf{L}(v)| \leq c-1$, and that v is *i-saturated* otherwise. Intuitively, an i-free vertex can be used to construct shortcuts for path P_i since either P_i is already using v and thus the route may be replaced with a shorter one without increasing the congestion of v, or the congestion of v is not yet saturated and can be increased by one to generate a better solution. We denote by $\mathcal{L}$ the set of all possible lists of indices that can be assigned to a vertex by a solution. More precisely, $\mathcal{L} = \{L \subseteq 2^{[k]} \mid |L| \leq c\}$. For $L \in \mathcal{L}$ we denote by $V(L)$ the set $\{v \in V(\mathcal{K}) \mid \mathsf{L}(v) = L\}$ of vertices of $\mathcal{K}$ which are assigned indices

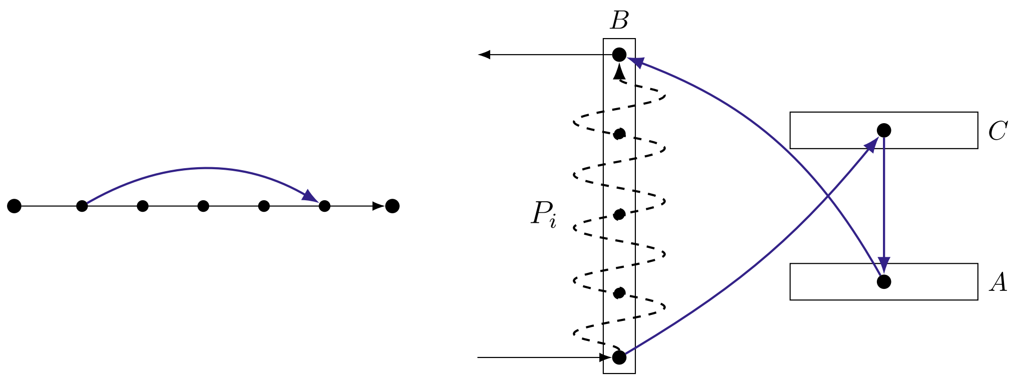


Figure 9 Examples of shortcuts. In both cases, the blue path denotes a shortcut. On the right, we give an illustration of a typical shortcut, internal to a k-triple (A, B, C), that is often built in the proof of Theorem 20. The dashed subpath of P_i is exchanged by the blue path in the figure.

in L. Finally, for $u \in C$ and $v = \mathsf{M}(u)$ we say that $\{u, v\}$ is an *i-free pair* if both u and v are i-free.

5.2 Freeing B

In this section, we show that only a bounded number of vertices $b \in B$ have $|\mathsf{L}(b)| = c$. We begin with the following warm up lemma.

▶ **Lemma 24.** *For all $P_i \in \mathcal{P}$, if $|V(P_i) \cap B| \geq 5$ then there are no i-free pairs.*

Proof. TOPROVE 16 ◀

▶ **Lemma 25.** *For all $L \in \mathcal{L}$ with $|L| = c$, it holds that $|V(L) \cap B| \leq 4$.*

Proof. TOPROVE 17 ◀

Lemma 25 implies that at least $f(k) - 4\binom{k}{c}$ vertices of B are used by at most $c - 1$ paths of $\mathcal{P}$. Thus, all such vertices are i-free for any $i \in [k]$ and can be used to build shortcuts for any path in $\mathcal{P}$.

5.3 Freeing A and C

We now show that only a bounded number of vertices of A and C are used by paths of $\mathcal{P}$.

▶ **Lemma 26.** *If there is $b \in B$ such that $|\mathsf{L}(b)| \leq c - 1$, then for all $P_i \in \mathcal{P}$ it holds that*
1. *$|V(P_i) \cap A| \leq 8c + 4$; and*
2. *$|V(P_i) \cap C| \leq 8c + 4$.*

Proof. TOPROVE 18 ◀

5.4 Finding the irrelevant vertex in polynomial time

Applying Lemma 26 we show an improved version of Lemma 25 which is needed in our proof.

▶ **Corollary 27.** *For all $i \in [k]$, $|V(P_i) \cap B| \leq 4$.*

Proof. TOPROVE 19 ◀

We are now ready to prove Theorem 20 restricted to semicomplete digraphs. The goal is to give a polynomial-time algorithm that finds a vertex $b \in B$ such that if there is a solution for $\mathcal{I}$, then there is another solution whose paths all avoid b.

We follow the blueprint of the proof by Fomin and Pilipczuk [24, Theorem 9.1]. The goal is to first find a large set $X \subseteq B$ such that every path of a solution entering some $b' \in X$ from a vertex not in A can be rerouted to access b from a vertex in A. Then, we must show that there is a $b \in X$ such that every path of the solution leaving b to a vertex not in C can be rerouted to leave b to a vertex in C. As these rerouting steps do not use too many extra vertices in the triple, by Corollary 27, and our hypothesis that the triple is large, we can then argue that b be can be replaced with another $b^* \in B$, that is unused by the solution, and thus b can be safely removed from the graph.

Thus, at some point in the proof, we need to analyze how a path of a solution for $\mathcal{I}$ can enter and leave B, and show how to reroute paths not respecting the desired behavior using vertices *outside* of the triple. In Fomin and Pilipczuk's proof, the fact that they consider collisions in arcs plays a major role here in this step. Informally, in a particular case of their analysis and using our notation, if a path P_i accesses $b' \in B$ through a vertex $v \in V(T) \setminus A$ and v has "sufficiently many" out-neighbors in $V(T) \setminus A$ which are in-neighbors of distinct vertices of B, then at least one of those out-neighbors, say v', can be used to reroute P_i through $v \to v' \to u \to \mathsf{M}(u) \to b$. This holds due to three main properties: (i) the fact that at most k arcs leaving a vertex are used by any solution, and thus any vertex of out-degree at least $k+1$ has one arc leaving it that is free, (ii) their versions of Lemmas 25 and 26, and (iii) a clever analysis on the behavior of the arcs between in-neighbors of vertices in B outside of A.

In our case, an analogous of the first of those three points requires more work since, a priori, it may seem that we have little control on how a solution intersects the in-neighborhood of B outside of A. We prove that this is not the case by applying another shortcutting argument which allows us to bound how many vertices a path $P_i \in \mathcal{P}$ can use in a particular subset of in-neighbors of B. In addition, after the first rerouting round, we can no longer rely on Property 23, as we do increase the length of each path by an $\mathcal{O}(1)$ factor. Thus a small trick is needed to apply a similar shortcutting argument as the one used in the in-neighbors of B, this time to the out-neighbors of B; this happens when we want to find a vertex of B that, when used by a path, is always followed by a vertex of C.

Proof. TOPROVE 20 ◀

Note that Lemmas 24–26 and Corollary 27 only deal with arcs and reroutings *inside* of the $f(k)$-triple $\mathcal{K}$. Thus, in order to adapt the proof to h-semicomplete digraphs, it suffices to add a factor of h to the steps above where vertices of large in- or out-neighborhood in B or in the heads or tails of the matchings of the form Y_0 (after **??**) and Y' (after **??**), for example. This implies that at each of those steps, the number of non-neighbors of the observed vertex is taken into account, and thus the same ideas work for $h \geq 1$.

6 Future research

This work touched on several common strategies used to cope with the hardness of the Directed Disjoint Paths problem. We investigated its (parameterized) complexity on tournaments and some of its superclasses, both with and without the typical relaxation of allowing congestion in the vertices. One of our key contributions is a fix (Theorem 1) for a gap in the literature originating in a flaw in the NP-hardness proof for k-DDP on tournaments given in [2]; we also as adapt a win/win approach based on directed pathwidth first used in an FPT algorithm for Directed Edge-Disjoint Paths parameterized by the number of paths on tournaments [23] to (k, c)-DDP when $2c > k$. We note, however, that this latter problem is *not* known to be NP-complete; while we are able to show that (k, k^{ε})-DDP is hard for every $\varepsilon \in [0, 1)$, we are unable to extend this to $c = \varepsilon k$, or even $c = k/2 + 1$. Alongside the already challenging problems listed in Section 1, we consider this the main open question related to our work. We do not provide all the details for the proof of Theorem 20 in its full generality, only for the case of semicomplete graphs, i.e., $h = 0$. Extending our arguments to $h > 0$ is essentially going deeper into technicalities: we must increase the size of the triple taking h into account and, when discussing the exterior neighborhood of the triple, increase the thresholds to classify the in- and out-neighbors of B as having “too many” neighbors in B or not. This extension, however, is not enough to give an FPT algorithm for (k, c)-DDP on h-semicomplete graphs when $2c > k$. In particular, two challenges remain: (i) computing a triple for elements of this class in FPT-time, as the only known algorithm being an XP one introduced by Kitsunai, Kobayashi, and Tamaki [36]; and (ii) devising an FPT algorithm for the joint parameterization by k, h, and the directed pathwidth, which would also improve upon the XP algorithm shown in [36]. We recall that, among the classes studied in our work, an FPT algorithm for h-semicomplete digraphs is the best we can hope for, as we have shown that k-DDP and (k, c)-DDP are W[1]-hard when parameterized by k and the number of covering tournaments on digraphs of directed pathwidth two (Theorem 4 and Theorem 19).

References

1 Saeed A. Amiri, Stephan Kreutzer, Dániel Marx, and Roman Rabinovich. Routing with congestion in acyclic digraphs. *Information Processing Letters*, 151:105836, 2019. doi:doi.org/10.1016/j.ipl.2019.105836.

2 Jørgen Bang-Jensen and Carsten Thomassen. A polynomial algorithm for the 2-path problem for semicomplete digraphs. *SIAM Journal on Discrete Mathematics*, 5(3):366–376, 1992. arXiv:https://doi.org/10.1137/0405027, doi:10.1137/0405027.

3 Jørgen Bang-Jensen and Carsten Thomassen. Personal communication, 2024.

4 Jørgen Bang-Jensen and Gregory Gutin. *Classes of Directed Graphs.* Springer Monographs in Mathematics, 2018. doi:10.1007/978-3-319-71840-8.

5 Florian Barbero, Christophe Paul, and Michal Pilipczuk. Strong immersion is a well-quasi-ordering for semicomplete digraphs. *Journal of Graph Theory*, 90(4):484–496, 2019. doi:10.1002/JGT.22408.

6 Adrian Bondy and U.S.R. Murty. *Graph Theory.* Springer-Verlag London, 2008.

7 Ulrik Brandes, Wolfram Schlickenrieder, Gabriele Neyer, Dorothea Wagner, and Karsten Weihe. A software package of algorithms and heuristics for disjoint paths in planar networks. *Discrete Applied Mathematics*, 92(2-3):91–110, 1999. doi:10.1016/S0166-218X(99)00048-7.

8 Rowland Leonard Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, 1941. doi:10.1017/S030500410002168X.

9 Victor Campos, Raul Lopes, Ana Karolinna Maia, and Ignasi Sau. Adapting the Directed Grid Theorem into an FPT Algorithm. *SIAM Journal on Discrete Mathematics*, 36(3):1887–1917, 2022. doi:10.1137/21M1452664.

10 Victor A. Campos, Jonas Costa, Raul Lopes, and Ignasi Sau. New Menger-Like Dualities in Digraphs and Applications to Half-Integral Linkages. In *Proc. of the 31st Annual European Symposium on Algorithms (ESA)*, volume 274 of *LIPIcs*, pages 30:1–30:18, 2023. doi:10.4230/LIPICS.ESA.2023.30.

11 Dario Giuliano Cavallaro, Ken-ichi Kawarabayashi, and Stephan Kreutzer. Edge-Disjoint Paths in Eulerian Digraphs. In *Proc. of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 704–715, 2024. doi:10.1145/3618260.3649758.

12 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(7):137–146, 2006. doi:10.4086/TOC.2006.V002A007.

13 Maria Chudnovsky, Alexandra O. Fradkin, and Paul D. Seymour. Tournament immersion and cutwidth. *Journal of Combinatorial Theory, Series B*, 102(1):93–101, 2012. doi:10.1016/J.JCTB.2011.05.001.

14 Maria Chudnovsky, Alex Scott, and Paul Seymour. Disjoint paths in unions of tournaments. *Journal of Combinatorial Theory, Series B*, 135:238–255, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0095895618300789, doi:10.1016/j.jctb.2018.08.007.

15 Maria Chudnovsky and Paul D. Seymour. A well-quasi-order for tournaments. *Journal of Combinatorial Theory, Series B*, 101(1):47–53, 2011. doi:10.1016/J.JCTB.2010.10.003.

16 Maria Chudnovsky, Paul D. Seymour, and Alex Scott. Disjoint paths in tournaments. *Advances in Mathematics*, 270:582–597, 2015. doi:10.1016/j.aim.2014.11.011.

17 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.

18 Andreas Darmann and Janosch Döcker. On simplified np-complete variants of monotone 3 -sat. *Discrete Applied Mathematics*, 292:45–58, 2021. URL: https://www.sciencedirect.com/science/article/pii/S0166218X2030528X, doi:10.1016/j.dam.2020.12.010.

19 Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

20 Katherine Edwards, Irene Muzi, and Paul Wollan. Half-Integral Linkages in Highly Connected Directed Graphs. In *Proc. of the 25th Annual European Symposium on Algorithms (ESA)*, volume 87 of *LIPIcs*, pages 36:1–36:12, 2017. Full version available in https://arxiv.org/abs/1611.01004. doi:10.4230/LIPIcs.ESA.2017.36.

21 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer-Verlag, 2006. doi:10.1007/3-540-29953-X.

22 Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing*. Cambridge University Press, 2019. doi:10.1017/9781107415157.

23 Fedor V. Fomin and Michal Pilipczuk. On width measures and topological problems on semi-complete digraphs. *Journal of Combinatorial Theory, Series B*, 138:78–165, 2019. doi:10.1016/J.JCTB.2019.01.006.

24 Fedor V. Fomin and Michał Pilipczuk. On width measures and topological problems on semi-complete digraphs. *Journal of Combinatorial Theory, Series B*, 138:78–165, 2019. doi:10.1016/j.jctb.2019.01.006.

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

26 Alexandra O. Fradkin and Paul D. Seymour. Tournament pathwidth and topological containment. *Journal of Combinatorial Theory, Series B*, 103(3):374–384, 2013. doi:10.1016/J.JCTB.2013.03.001.

27 Alexandra O. Fradkin and Paul D. Seymour. Edge-disjoint paths in digraphs with bounded independence number. *Journal of Combinatorial Theory, Series B*, 110:19–46, 2015. doi:10.1016/J.JCTB.2014.07.002.

28 Robert Ganian, Petr Hliněný, Joachim Kneis, Alexander Langer, Jan Obdržálek, and Peter Rossmanith. Digraph width measures in parameterized algorithmics. *Discrete Applied Mathematics*, 168:88–107, 2014. doi:10.1016/j.dam.2013.10.038.

29 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. URL: https://dl.acm.org/doi/10.5555/574848.

30 Archontia C. Giannopoulou, Ken-ichi Kawarabayashi, Stephan Kreutzer, and O-joung Kwon. The canonical directed tree decomposition and its applications to the directed disjoint paths problem, 2020. This article is the full version of reference [31]. arXiv:2009.13184.

31 Archontia C. Giannopoulou, Ken-ichi Kawarabayashi, Stephan Kreutzer, and O-joung Kwon. Directed tangle tree-decompositions and applications. In *Proc. of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–405, 2022. The full version of this article is reference [30]. doi:10.1137/1.9781611977073.19.

32 Meike Hatzel, Stephan Kreutzer, Marcelo Garlet Milani, and Irene Muzi. Cycles of Well-Linked Sets and an Elementary Bound for the Directed Grid Theorem. In *Proc. of the 65th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–20, 2024. doi:10.1109/FOCS61266.2024.00011.

33 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Stephan Kreutzer. An excluded half-integral grid theorem for digraphs and the directed disjoint paths problem. In *Proc. of the 46th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 70–78, 2014. doi:10.1145/2591796.2591876.

34 Ken-ichi Kawarabayashi and Stephan Kreutzer. The Directed Grid Theorem. In *Proc. of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 655–664. ACM, 2015. doi:10.1145/2746539.2746586.

35 Ilhee Kim and Paul D. Seymour. Tournament minors. *Journal of Combinatorial Theory, Series B*, 112:138–153, 2015. doi:10.1016/J.JCTB.2014.12.005.

36 Kenta Kitsunai, Yasuaki Kobayashi, and Hisao Tamaki. On the pathwidth of almost semi-complete digraphs. In Nikhil Bansal and Irene Finocchi, editors, *Proc. of the 23rd Annual European Symposium on Algorithms (ESA)*, volume 9294 of *LNCS*, pages 816–827, 2015. doi:10.1007/978-3-662-48350-3_68.

37 Raul Lopes and Ignasi Sau. A relaxation of the Directed Disjoint Paths problem: A global congestion metric helps. *Theoretical Computer Science*, 898:75–91, 2022. doi:10.1016/j.tcs.2021.10.023.

38 Lúcia Martins, Teresa Gomes, and David Tipper. Efficient heuristics for determining node-disjoint path pairs visiting specified nodes. *Networks*, 70(4):292–307, 2017. doi:10.1002/NET.21778.

39 Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. doi:10.1093/acprof:oso/9780198566076.001.0001.

40 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.

41 Aleksandrs Slivkins. Parameterized tractability of edge-disjoint paths on directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 24(1):146–157, 2010. doi:10.1137/070697781.

42 Carsten Thomassen. Highly connected non-2-linked digraphs. *Combinatorica*, 11(4):393–395, 1991. doi:10.1007/BF01275674.

43 Michal Wlodarczyk. Constant Approximating Disjoint Paths on Acyclic Digraphs Is W[1]-Hard. In *Proc. of the 35th International Symposium on Algorithms and Computation (ISAAC)*, volume 322 of *LIPIcs*, pages 57:1–57:16, 2024. doi:10.4230/LIPICS.ISAAC.2024.57.

A Flaw in the NP-completeness proof of Bang-Jensen and Thomassen

In this appendix we sketch the flaw in the proof of Bang-Jensen and Thomassen claiming that Directed Disjoint Paths is NP-complete on tournaments when k is part of the input.

Their approach is to first prove the NP-completeness of a related problem on tournaments, and then reduce from this problem to Directed Disjoint Paths. Namely, it is proved in [2, Theorem 6.1] that the following problem Q is NP-complete on tournaments: given a tournament T and a set of arcs $A \subseteq E(T)$, decide whether T contains a directed cycle visiting all the arcs in A (in any order). The proof of [2, Theorem 6.1] is correct, and consists in a simple reduction from Hamiltonian Cycle on general digraphs to Q on tournaments.

The problem comes later: right after the proof of [2, Theorem 6.1], it is claimed that "This proves that, if k is not fixed, then the k-DDP problem is NP-complete on tournaments". This statement is not clear at all, the main issue being that in problem Q the set of arcs A to be visited by the cycle is *unordered*. Indeed, in order to construct an instance of k-DDP starting from an instance (T, A) of problem Q, the natural strategy would be to consider an arbitrary ordering $\sigma = (e_0, \ldots, e_{k-1})$ of the arcs in A, and construct a set K of k requests in T as follows. For every $i \in \{0, \ldots, k-1\}$, add to K a request from the head of e_i to the tail of e_{i+1}, where indices are taken modulo k. One may hope that the desired cycle C in T visiting all the arcs in A would translate to the existence of k pairwise vertex-disjoint paths in T satisfying all the requests in K. But this is not true, as C may exist in T, but may yield a visiting ordering of the arcs in A that is different from the ordering σ that we have fixed arbitrarily. One could try to fix this issue by guessing all the possible orderings σ of the arcs in A, but this would result in $k!$ choices, which is not allowed in a polynomial-time reduction since k is considered as part of the input.

We have shared the above issue with the authors of [2] and they have confirmed to us [3] that it does not seem to be easily fixable.