A Relaxation of the Directed Disjoint Paths Problem: A Global Congestion Metric Helps

Raul Lopes
Departamento de Computação, Universidade Federal do Ceará, Fortaleza, Brazil
LIRMM, Université de Montpellier, France
raul.lopes@lia.ufc.br

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

Abstract

In the Directed Disjoint Paths problem, we are given a digraph D and a set of requests $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, and the task is to find a collection of pairwise vertex-disjoint paths $\{P_1, \ldots, P_k\}$ such that each P_i is a path from s_i to t_i in D. This problem is NP-complete for fixed $k = 2$ and W[1]-hard with parameter k in DAGs. A few positive results are known under restrictions on the input digraph, such as being planar or having bounded directed tree-width, or under relaxations of the problem, such as allowing for vertex congestion. Good news are scarce, however, for general digraphs. In this article we propose a novel global congestion metric for the problem: we only require the paths to be "disjoint enough", in the sense that they must behave properly not in the whole graph, but in an unspecified large part of it. Namely, in the Disjoint Enough Directed Paths problem, given an n-vertex digraph D, a set of k requests, and non-negative integers d and s, the task is to find a collection of paths connecting the requests such that at least d vertices of D occur in at most s paths of the collection. We study the parameterized complexity of this problem for a number of choices of the parameter, including the directed tree-width of D. Among other results, we show that the problem is W[1]-hard in DAGs with parameter d and, on the positive side, we give an algorithm in time $\mathcal{O}(n^{d+2} \cdot k^{d \cdot s})$ and a kernel of size $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ in general digraphs. This latter result has consequences for the Steiner Network problem: we show that it is FPT parameterized by the number k of terminals and d, where $d = n - c$ and c is the size of the solution.

2012 ACM Subject Classification Theory of computation → Fixed parameter tractability; Theory of computation → Design and analysis of algorithms

Keywords and phrases Parameterized complexity, directed disjoint paths, congestion, dual parameterization, kernelization, directed tree-width

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.68

Related Version A full version of the paper is available at `https://arxiv.org/abs/1909.13848`.

Funding *Raul Lopes*: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).
Ignasi Sau: Projects DEMOGRAPH (ANR-16-CE40-0028) and ESIGMA (ANR-17-CE23-0010).

1 Introduction

In the Disjoint Paths problem, we are given a graph G and a set of pairs of vertices $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, the *requests*, and the task is to find a collection of pairwise vertex-disjoint paths $\{P_1, \ldots, P_k\}$ such that each P_i is a path from s_i to t_i in G. Since this problem is NP-complete in the directed and undirected cases, even if the input graph is planar [13, 19], algorithmic approaches usually involve approximations, parameterizations, and relaxations. In this article, we focus on the latter two approaches and the directed case.

Previous work. For the undirected case, Robertson and Seymour [23] showed, in their seminal work on graph minors, that Disjoint Paths can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$ for some computable function f, where n is the number of vertices of G; that is, the problem is *fixed-parameter tractable* (FPT) when parameterized by the number of requests.

The directed case, henceforth referred to as the Directed Disjoint Paths (DDP) problem, turns out to be significantly harder: Fortune et al. [13] showed that the problem is NP-complete even for fixed $k = 2$. In order to obtain positive results, a common approach has been to consider restricted input digraphs. For instance, it is also shown in [13] that DDP is solvable in time $n^{\mathcal{O}(k)}$ if the input digraph is acyclic. In other words, DDP is XP in DAGs with parameter k. For some time the question of whether this could be improved to an FPT algorithm remained open, but a negative answer was given by Slivkins [25]: DDP is W[1]-hard in DAGs with parameter k. Johnson et al. introduced in [14] the notion of directed tree-width, as a measure of the distance of a digraph to being a DAG, and provided generic conditions that, if satisfied by a given problem, yield an XP algorithm on graphs of bounded directed tree-width. In particular, they gave an $n^{\mathcal{O}(k+w)}$ algorithm for DDP on digraphs with directed tree-width at most w. Another restriction considered in the literature is to ask for the underlying graph of the input digraph to be planar. Under this restriction, Schrijver [24] provided an XP algorithm for DDP with parameter k, which was improved a long time afterwards to an FPT algorithm by Cygan et al. [8].

A natural relaxation for the Directed Disjoint Paths problem is to allow for vertex and/or edge congestion. Namely, in the Directed Disjoint Paths with Congestion problem (DDPC for short, or DDPC-c if we want to specify the value of the congestion), the task is to find a collection of paths satisfying the k requests such that no vertex in the graph occurs in more than c paths of the collection. Amiri et al. [1] considered the tractability of this problem when restricted to DAGs. Namely, they showed that DDPC-c in DAGs is W[1]-hard for every fixed $c \geq 1$ but admits an XP algorithm with parameter d, where $d = k - c$. By a simple local reduction to the general version given in [1] and the framework given by Johnson et al. [14], it follows that DDPC-c also admits an XP algorithm with parameters k and w for every fixed $1 \leq c \leq n - 1$ in digraphs with directed tree-width at most w, and the same result also holds when we allow for congestion on the edges.

Motivated by Thomassen's proof [26] that DDP remains NP-complete for $k = 2$ when restricted to β-strongly connected digraphs, for any integer $\beta \geq 1$, Edwards et al. [11] recently considered the DDPC-2 problem (this version of the problem is usually called *half-integral* in the literature) and proved, among other results, that it can be solved in time $n^{f(k)}$ when restricted to $(36k^3 + 2k)$-strongly connected digraphs.

Kawarabayashi et al. [16] considered the following asymmetric version of the DDPC-4 problem: the task is to either find a set of paths satisfying the requests with congestion at most four, or to conclude that no set of pairwise vertex-disjoint paths satisfying the requests exists. In other words, we ask for a solution for DDPC-4 or a certificate that there is no solution for DDP. They proved that this problem admits an XP algorithm with parameter k in general digraphs, and claimed –without a proof– that Slivkins' reduction [25] can be modified to show that it is W[1]-hard in DAGs. In their celebrated proof of the Directed Grid Theorem, Kawarabayashi and Kreutzer [17] claimed that an XP algorithm can be also obtained for the asymmetric version with congestion at most three. To the best of our knowledge, the existence of an XP algorithm in general digraphs for the DDPC-2 problem, or even for its asymmetric version, remains open.

Summarizing, the existing positive results in the literature for parameterizations and/or relaxations of the Directed Disjoint Paths problem in *general digraphs* are quite scarce.

Our approach, results, and techniques. In this article, we propose another congestion metric for DDP. In contrast to the usual relaxations discussed above, which focus on a *local* congestion metric that applies to every vertex, our approach considers, on top of local congestion, a *global* congestion metric: we want to keep control of how many vertices (a global metric) appear in "too many" paths (a local metric) of the solution. That is, we want the paths to be such that "most" vertices of the graph do not occur in too many paths, while allowing for any congestion in the remaining vertices. In the particular case where we do not allow for local congestion, we want the paths to be pairwise vertex-disjoint not in the whole graph, but in a large enough set of vertices; this is why we call such paths "disjoint enough".

Formally, in the Disjoint Enough Directed Paths (DEDP) problem, we are given a set of requests $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ in a digraph D and two non-negative integers c and s, and the task is to find a collection of paths $\{P_1, \ldots, P_k\}$ such that each P_i is a path from s_i to t_i in D and at most c vertices of D occur in more than s paths of the collection. If $s = 1$, for instance, we ask for the paths to be pairwise vertex-disjoint in at least $n - c$ vertices of the graph, and allow for at most c vertices occurring in two or more paths. Choosing $c = 0$ and $s = 1$, DEDP is exactly the DDP problem and, choosing $s = 0$, DEDP is exactly the Steiner Network problem (see [12] for its definition).

By a simple reduction from the Directed Disjoint Paths with Congestion problem, it is easy to prove that DEDP is NP-complete for fixed $k \geq 3$ and $s \geq 1$, even if c is large with respect to n, namely at most $n - n^\alpha$ for some real value $0 < \alpha \leq 1$, and W[1]-hard in DAGs with parameter k. By applying the framework of Johnson et al. [14], we give an $n^{\mathcal{O}(k+w)}$ algorithm to solve DEDP in digraphs with directed tree-width at most w.

The fact that DEDP is NP-complete for fixed values of $k = 2$, $c = 0$, and $s = 1$ [13] motivates us to consider the "dual" parameter $d = n - c$. That is, instead of bounding from above the number of vertices of D that lie in the intersection of many paths of a collection satisfying the given requests, we want to bound from below the number of vertices that occur only in few paths of the collection. Formally, we want to find $X \subseteq V(D)$ with $|X| \geq d$ such that there is a collection of paths $\mathcal{P}$ satisfying the given requests such that every vertex in X is in at most s paths of the collection. We first prove, from a reduction from the Independent Set problem, that DEDP is W[1]-hard with parameter d for every fixed $s \geq 0$, even if the input graph is a DAG and all source vertices of the request set are the same.

Our main contribution consists of positive algorithmic results for this dual parameterization. On the one hand, we give an algorithm for DEDP running in time $\mathcal{O}(n^d \cdot k^{d \cdot s})$. This algorithm is not complicated, and basically performs a brute-force search over all vertex sets of size d, followed by k connectivity tests in a digraph D' obtained from D by an appropriate local modification. On the other hand, our most technically involved result is a kernel for DEDP with at most $d \cdot 2^{k-s} \cdot \binom{k}{s}$ non-terminal vertices. This algorithm first starts by a reduction rule that eliminates what we call *congested* vertices; we say that the resulting instance is *clean*. We then show that if D is clean and sufficiently large, and $k = s + 1$, then the instance is positive and a solution can be found in polynomial time. This fact is used as the base case of an iterative algorithm. Namely, we start with the original instance and proceed through $k - s + 1$ iterations. At each iteration, we choose one path from some s_i to its destination t_i such that a large part of the graph remains unused by any of the pairs chosen so far (we prove that such a request always exists) and consider only the remaining requests for the next iteration. We repeat this procedure until we arrive at an instance where the number of requests is exactly $s + 1$, and use the base case to output a solution for it. From this solution, we extract in polynomial time a solution for the original instance, yielding a kernel of the claimed size.

Table 1 Summary of hardness and algorithmic results for distinct choices of the parameters. A horizontal line in a cell means no restrictions for that case. In all cases, we have that $c = n - d$.

k	d	s	w	Complexity
fixed ≥ 3	n^{α}	fixed ≥ 1	—	NP-complete (Theorem 3)
parameter	n^{α}	fixed ≥ 1	0	W[1]-hard (Theorem 3)
input	parameter	fixed ≥ 0	—	W[1]-hard (Theorem 4)
parameter	—	—	parameter	XP (see full version)
input	parameter	parameter	—	XP (Theorem 6)
parameter	parameter	parameter	—	FPT (Theorem 15)

Since positive results for the Directed Disjoint Paths problem are not common in the literature, especially in general digraphs, we consider our algorithmic results to be of particular interest. Furthermore, the kernelization algorithm also brings good news for the Steiner Network problem: when $s = 0$ Feldmann and Marx in [12] showed that the tractability of the Steiner Network problem when parameterized by the number of requests depends on how the requests are structured. Our result adds to the latter by showing that the problem remains FPT if we drop this structural condition on the request set but add d, the number of vertices occurring in at most s paths of the solution, as a parameter. More details can be found in Section 2.

Table 1 shows a summary of our algorithmic and complexity results, which altogether provide an accurate picture of the parameterized complexity of the DEDP problem for distinct choices of the parameters.

Organization. In Section 2 we present some preliminaries and formally define the Disjoint Enough Directed Paths problem. We refer the reader to [7, 9] for basic background on parameterized complexity, and for completeness we recall some basic definitions in the full version of this article, available at `https://arxiv.org/abs/1909.13848`. Due to space limitations, the basic definitions of arboreal decompositions and directed tree-width can also be found in the full version. We provide the hardness results in Section 3. The XP algorithm with parameters k and w for DEDP in graphs with directed tree-width at most w can be found in the full version. The algorithms with d as a parameter are given in Section 4. We conclude the article in Section 5 with some open questions for further research. The proofs of the three results marked with '($\star$)' can be found in the full version.

2 Preliminaries and definitions

All paths mentioned henceforth, unless stated otherwise, are considered to be directed. For a graph $G = (V, E)$, directed or not, and a set $X \subseteq V(G)$, we write $G - X$ for the graph resulting from the deletion of X from G and $G[X]$ for the graph induced by X. If e is an edge of a directed or undirected graph with extremities u and v, we may refer to e as (u, v). For $v \in V(G)$, we write $\deg_G(v)$ to be the degree of v in G, and $N^+(v)$, $N^-(v)$ for the set of out-neighbors and in-neighbors of v, respectively, when G is a digraph. We also write $G' \subseteq G$ to say that G' is a subgraph of G. A *weak component* of a directed graph D is set of vertices inducing a connected component in the underlying graph of D. Unless stated otherwise, n will always denote the number of vertices of the input graph. For an integer $\ell \geq 1$, we denote by $[\ell]$ the set $\{1, 2, \ldots, \ell\}$. We also make use of Menger's Theorem [20] for digraphs. Here a (u, v)-*separator* is a set of vertices X such that there is no path from u to v in $D - X$.

▶ **Theorem 1** (Menger's Theorem). *Let D be a digraph and $u, v \in V(D)$ such that $(u, v) \notin E(D)$. Then the minimum size of a (u, v)-separator equals the maximum number of pairwise internally vertex-disjoint paths from u to v.*

For two positive integers a and b with $a \geq b$, the *Stirling number of the second kind* [4], denoted by $\mathsf{Stirling}(a, b)$, counts the number of ways to partition a set of a objects into b non-empty subsets, and is bounded from above by $\frac{1}{2}\binom{a}{b} \cdot b^{a-b}$.

Before defining the problem, we define requests and satisfying collections.

▶ **Definition 2** (Requests and satisfying collections). *Let D be a digraph and $\mathcal{P}$ be a collection of paths of D. A* request *in D is a pair of vertices of D. For a request set $I = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$, we say that the vertices $\{s_1, s_2, \ldots, s_k\}$ are* source *vertices and that $\{t_1, t_2, \ldots, t_k\}$ are* target *vertices, and we refer to them as $S(I)$ and $T(I)$, respectively. We say that $\mathcal{P}$* satisfies *I if $\mathcal{P} = \{P_1, \ldots, P_k\}$ and P_i is a path from s_i to t_i, for $i \in [k]$.*

We remark that a request set may contain many copies of the same pair, and that when considering the union of two or more requests, we keep all such copies in the resulting request set. For instance, if $I_1 = \{(u_1, v_1)\}$ and $I_2 = \{(u_1, v_1), (u_2, v_2)\}$ then $I_1 \cup I_2 = \{(u_1, v_1), (u_1, v_1), (u_2, v_2)\}$, and this indicates that a collection of paths satisfying this request set must contain two paths from u_1 to v_1. The DEDP problem is defined as follows.

Disjoint Enough Directed Paths (DEDP)	
Input:	A digraph D, a request set I of size k, and two non-negative integers c and s.
Output:	A collection of paths $\mathcal{P}$ satisfying I such that at most c vertices of D occur in at least $s+1$ paths of $\mathcal{P}$ and all other vertices of D occur in at most s paths of $\mathcal{P}$.

Unless stated otherwise, we consider $d = n - c$ for the remaining of this article. Intuitively, c imposes an upper bound on the size of the "**c**ongested" part of the solution, while d imposes a lower bound on the size of the "**d**isjoint" part. For a parameterized version of DEDP, we sometimes include the parameters before the name. For instance, we denote by (k, d)-DEDP the Disjoint Enough Directed Paths problem with parameters k and d.

Notice that if $c \geq n$ or $s \geq k$, the problem is trivial since every vertex of the graph is allowed to be in all paths of a collection satisfying the requests, and thus we need only to check for connectivity between the given pairs of vertices. Furthermore, if there is a pair (s_i, t_i) in the request set such that there is no path from s_i to t_i in the input digraph D, the instance is negative. Thus we henceforth assume that $c < n$, that $s < k$, and that there is a path from s_i to t_i in D for every pair (s_i, t_i) in the set of requests.

Let $0 < \alpha \leq 1$. Choosing the values of k, d, and s appropriately, we show in Table 2 that the DEDP problem generalizes several problems in the literature.

Table 2 Summary of related problems and complexity results.

Parameters	Generalizes	Complexity
$d = n$, $s = 1$	Disjoint Paths	NP-complete for $k = 2$ [13]
$d = n$, $s \geq 2$	Disjoint Paths with Congestion	NP-complete for $k \geq 3$
$d \geq 1$, $s = 0$	Steiner Network	FPT with parameters k and d

The last line of Table 2 is of particular interest, and we focus on it in the next two paragraphs. In the Steiner Network problem, we are given a digraph D and a request set I and we are asked to find an induced subgraph D' of D with minimum number of vertices such that D' admits a collection of paths satisfying I. For a request set I in a digraph D, let $D(I)$ be the digraph with vertex set $S(I) \cup T(I)$ and edge set $\{(s,t) \mid (s,t) \in I\}$. The complexity landscape of the Steiner Network problem when parameterized by the size of the request set was given by Feldmann and Marx [12]. They showed that the tractability of the problem depends on $D(I)$. Namely, they proved that if $D(I)$ is close to being a *caterpillar*, then the Steiner network problem is FPT when parameterized by $|I|$, and W[1]-hard otherwise. When parameterized by the size of the solution, Jones et al. [15] showed that the Steiner Network problem is FPT when $D[I]$ is a star whose edges are all oriented from the unique source and the underlying graph of the input digraph excludes a topological minor, and W[2]-hard on graphs of degeneracy two [15].

Our algorithmic results for DEDP for the particular case $s = 0$ yield an FPT algorithm for another parameterized variant of the Steiner Network problem. In this case, we want to decide whether D admits a large set of vertices whose removal does not disconnect any pair of requests. That is, we want to find a set $X \subseteq V(D)$ with $|X| \geq d$ such that $D - X$ contains a collection of paths satisfying I. In Theorem 15 we give an FPT algorithm (in fact, a kernel) for this problem with parameters $|I|$ and d. We remark that this tractability does not depend on $D(I)$.

3 Hardness results for DEDP

In this section we provide hardness results for the DEDP problem. Namely, we first provide in Theorem 3 a simple reduction from Disjoint Paths with Congestion, implying NP-completeness for fixed values of k, c, d and W[1]-hardness in DAGs with parameter k. We then prove in Theorem 4 that DEDP is W[1]-hard in DAGs with parameter d.

As mentioned in [15], the Steiner Network problem is W[2]-hard when parameterized by the size of the solution (as a consequence of the results of [21]). Hence (c)-DEDP is W[2]-hard for fixed $s = 0$. As discussed in the introduction, the Directed Disjoint Paths problem is NP-complete for fixed $k = 2$ [13] and W[1]-hard with parameter k in DAGs [25]. Allowing for vertex congestion does not improve the tractability of the problem: Disjoint Paths with Congestion parameterized by number of requests is also W[1]-hard in DAGs for every fixed congestion $c \geq 1$, as observed in [1]. When $c = 0$ and $s \geq 1$, DEDP is equivalent to the Directed Disjoint Paths with Congestion problem and thus the aforementioned bounds also apply to it. In the following theorem we complete this picture by showing that DEDP is NP-complete for fixed $k \geq 3$ and $s \geq 1$, even if c is quite large with respect to n (note that if $c = n$ all instances are trivially positive), namely for c as large as $n - n^\alpha$ with α being any fixed real number such that $0 < \alpha \leq 1$. The same reduction also allows to prove W[1]-hardness in DAGs with parameter k. The idea is, given the instance of DDPC, build an instance of DEDP where the "disjoint" part corresponds to the original instance, and the "congested" part consists of c new vertices that are necessarily used by $s + 1$ paths. This is why we restrict the value of d to be of the form n^α, but not smaller; otherwise, the "disjoint" part, which corresponds to the instance of DDPC, would be too small compared to the total size of the graph, and a brute-force algorithm could solve the problem in polynomial time.

▶ **Theorem 3** ($\star$). *Let $0 < \alpha \leq 1$, $d = n^\alpha$, and $c = \lceil n - d \rceil$. Then:*

(i) *DEDP is* NP*-complete for every fixed $k \geq 3$ and $s \geq 1$; and*

(ii) *(k)-DEDP is* W[1]*-hard in DAGs for every fixed $s \geq 1$.*

Next, we show that (d)-DEDP is W[1]-hard, even when the input graph is acyclic and all source vertices of the request set are the same. The reduction is from Independent Set parameterized by the size of the solution, which is W[1]-hard [7, 9].

▶ **Theorem 4.** *The DEDP problem is* W[1]*-hard with parameter d for every fixed $s \geq 0$, even when the input graph is acyclic and all source vertices in the request set are the same.*

Proof. Let (G, d) be an instance of the Independent Set problem, in which we want to decide whether the (undirected) graph G contains an independent set of size at least d, and s be a non-negative integer. Let V^E be the set $\{v_e \mid e \in E(G)\}$ and D a directed graph with vertex set $V(G) \cup \{r\} \cup V^E$. Add to D the following edges:

- for every $v \in V(G)$, add the edge (r, v); and
- for every edge $e \in E(G)$ with endpoints u and w, add the edges (u, v_e) and (w, v_e).

Finally, for every $v_e \in V^E$, add to I $2s + 1$ copies of the pair (r, v_e). Figure 1 illustrates this construction.

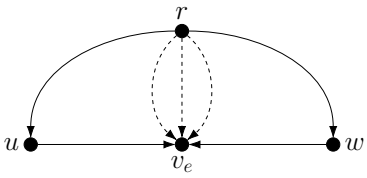

Figure 1 Example of the construction from Theorem 4 with $s = 1$ and $e = (u, w)$. A dashed line indicates a request in I.

Notice that each vertex v_e of D associated with an edge E of D has out-degree zero in D and r has in-degree zero. Moreover, every edge of D has as extremity either r or a vertex of the form v_e. Thus D is a acyclic, as desired. Furthermore $|S(I)| = 1$ since all of its elements are of the form (r, v_e), for $e \in E(G)$. We now show that (G, d) is positive if and only if (D, I, k, c, s) is positive, where $k = |I| = m \cdot (2s + 1)$.

For the necessity, let X be an independent set of size d in G and let $u \in X$. Start with a collection $\mathcal{P} = \emptyset$. We classify the edges of G into two sets: the set E_1 containing all edges with both endpoints in $V(G) - X$, and the set E_2 containing all edges with exactly one endpoint in X. Now, for each $e \in E_1$, chose arbitrarily one endpoint u of e and add to $\mathcal{P}$ $2s + 1$ copies of the path in D from r to v_e using u. For each $e \in E_2$ with $e = (u, w)$ and $w \notin X$, add to $\mathcal{P}$ $2s + 1$ copies of the path in D from r to v_e using w. Since X is an independent set, no vertex in X occurs in any path of $\mathcal{P}$, and since $E(G) = E_1 \cup E_2$, $\mathcal{P}$ satisfies I and the necessity follows as $c = n - d$.

Let $\mathcal{P}$ be a solution for (D, I, k, c, s) and $X \subseteq V(D)$ be a set of vertices with $|X| = d$ and such that each vertex of X occurs in at most s paths of $\mathcal{P}$. Such choice is possible since $d = n - c$. For contradiction, assume that X is not an independent set in G. Then there is an edge $e \in E(G)$ with $e = (u, w)$ and $u, w \in X$, and $2s + 1$ copies of the request (r, v_e) in I. Thus each path satisfying one of those requests uses u or w, but not both, and therefore either u or w occurs in at least $s + 1$ paths of $\mathcal{P}$, a contradiction. We conclude that X is an independent set in G and the sufficiency follows. ◀

4 Algorithms for DEDP including d as a parameter

In this section we focus on algorithmic results for DEDP. In Theorem 3 we showed that DEDP is NP-complete for fixed $k \geq 3$ and a large range of values of c. The proof does not hold, for example, when $c = n - d$ for a constant d. We also showed that considering only d

as a parameter is still not enough to improve the tractability of the problem: Theorem 4 shows that (d)-DEDP is W[1]-hard in DAGs even if all requests share the same source. The situation changes when we consider stronger parameterizations including d: we show that the problem is XP with parameters d and s (cf. Theorem 6), and FPT with parameters k and d (hence s as well, since we may assume that $k > s$ as discussed below; cf. Theorem 15). It is worth mentioning that this kind of *dual parameterization* has proved useful in order to improve the tractability of several notoriously hard problems (cf. for instance [1–3, 6, 10]).

The following definition will be useful in the description of the algorithms of this section.

▶ **Definition 5.** *Let D be a graph, I be a request set with $I = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, and s be an integer. We say that a set $X \subseteq V(D)$ is* s-viable for I *if there is a collection of paths $\mathcal{P}$ satisfying I such that each vertex of X occurs in at most s paths of $\mathcal{P}$. We also say that $\mathcal{P}$ is* certifying X*.*

Thus an instance (D, I, k, c, s) of DEDP is positive if and only if D contains an s-viable set X with $|X| \geq d$. Since we now consider d as a parameter instead of c, from this point onwards we may refer to instances of DEDP as (D, I, k, d, s).

▶ **Theorem 6** ($\star$)**.** *There is an algorithm running in time $\mathcal{O}(n^{d+2} \cdot k^{d \cdot s})$ for the* Disjoint Enough Directed Paths *problem.*

We now proceed to show that (k, d, s)-DEDP is FPT, by providing a kernel with at most $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ vertices. We start with some definitions and technical lemmas.

We remark that any vertex in D whose deletion disconnects more than s pairs in the request set cannot be contained in any solution for an instance of DEDP. Hence we make use of an operation to eliminate all such vertices from the input graph while maintaining connectivity. We use the following definitions. We remind the reader that, for a request set I, we denote by $S(I)$ the set of source vertices in I and by $T(I)$ the set of target vertices in I (cf. Definition 2).

▶ **Definition 7** (Non-terminal vertices)**.** *Let (D, I, k, d, s) be an instance of DEDP. For a digraph D' such that $V(D') \subseteq V(D)$, we define $V^*(D') = V(D') \setminus (S(I) \cup T(I))$.*

That is, $V^*(D)$ is the set of non-terminal (i.e., neither source nor target) vertices of D.

▶ **Definition 8** (Congested vertex and blocking collection)**.** *Let (D, I, k, d, s) be an instance of DEDP. For $X \subseteq V^*(D)$, we define I_X as the subset of I that is* blocked *by X, that is, there are no paths from s to t in $D - X$ for every $(s_i, t_i) \in I_X$. We say that a vertex $v \in V^*(D)$ is an* (I, s)-congested vertex *of D if $|I_{\{v\}}| \geq s + 1$. The* blocking collection of I *is the collection $\{B_1, \ldots, B_k\}$ where $B_i = \{v \in V^*(D) \mid (s_i, t_i) \in I_{\{v\}}\}$, for $i \in [k]$. We say that D is* clean for I *and that (D, I, k, d, s) is a* clean instance *if there are no congested vertices in $V^*(D)$. When I and s are clear from the context, we drop them from the notation.*

We use the following operation (cf. Figure 2) to eliminate congested vertices of D while maintaining connectivity. It is used, for instance, in [5] (as the *torso* operation) and in [18].

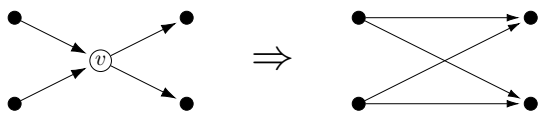

Figure 2 Bypassing a vertex v.

▶ **Definition 9** (Bypassing vertices and sets). *Let D be a graph and $v \in V(D)$. We refer to the following operation as* bypassing *v: delete v from D and, for each $u \in N^-(v)$ add one edge from u to each vertex $w \in N^+(v)$. We denote by D/v the graph generated by bypassing v in D. For a set of vertices $B \subseteq V(D)$, we denote by D/B the graph generated by bypassing, in D, all vertices of B in an arbitrary order.*

We restrict our attention to vertices in $V^*(D)$ in Definition 8 because we want to avoid bypassing source or target vertices, and work only with vertices inside $V^*(D)$. Since $|S(I) \cup T(I)| \leq 2k$, we show later that this incurs an additive term of $2k$ in the size of the constructed kernel.

In [18] it is shown that the ending result of bypassing a set of vertices in a digraph does not depend on the order in which those vertices are bypassed. Furthermore, bypassing a vertex of D cannot generate a new congested vertex: if u is a congested vertex of D/v, then u is also a congested vertex of D, for any $v \in V(D) \setminus \{u\}$. Thus any instance (D, I, k, d, s) of DEDP is equivalent to the instance $(D/v, I, k, d, s)$, if v is a congested vertex of D, and arbitrarily bypassing a vertex of D can only make the problem harder. We formally state those observations below.

▶ **Lemma 10.** *Let (D, I, k, d, s) be an instance of DEDP and B be the set of (I, s)-congested vertices of D. Let $D' = D/B$ and consider the instance (D', I, k, d, s). Then, X is a solution for the former if and only if X is a solution for the latter.*

Proof. Let X be a solution for (D, I, k, d, s). We claim that $X \cap B = \emptyset$. Otherwise, at least $s + 1$ paths of any collection satisfying I must intersect in X, contradicting our choice for X. Hence $X \subseteq V(D')$ and is a solution for (D', I, k, d, s). Similarly, if $X \subseteq V(D')$ then $X \cap B = \emptyset$ and the sufficiency follows. ◀

Thus, any solution for an instance resulting from bypassing a set of vertices in $V^*(D)$ is also a solution for the original instance.

▶ Remark 11. Let (D, I, k, d, s) be an instance of DEDP and $Y \subseteq V^*(D)$. If X is a solution for $(D/Y, I, k, d, s)$, then X is also a solution for (D, I, k, d, s).

The main ideas of the kernelization algorithm are the following. Let (D, I, k, d, s) be an instance of DEDP and $\{B_1, \ldots, B_k\}$ be the blocking collection of I. First, we show that, if D is clean for I, there is an $i \in [k]$ such that $|V^*(D) - B_i| \geq n/(k - s)k$ (Lemma 12). Then, we show that if D is clean and sufficiently large, and $|I| = s + 1$, then the instance is positive and a solution can be found in polynomial time (Lemma 13).

Lemma 13 is used as the base case for our iterative algorithm. We start with the first instance, say (D, I, k, d, s), and proceed through $k - s + 1$ iterations. At each iteration, we will choose one path from some s_i to its destination t_i such that a large part of the graph remains unused by any of the pairs chosen so far (by Lemma 12) and consider the request set containing only the remaining pairs for the next iteration. We repeat this procedure until we arrive at an instance where the number of requests is exactly $s + 1$, and show that if n is large enough, then we can use Lemma 13 to output a solution for the last instance. From this solution, we extract a solution for (D, I, k, d, s) in polynomial time.

▶ **Lemma 12.** *Let (D, I, k, d, s) be an instance of DEDP, $\{B_1, \ldots, B_k\}$ be the blocking collection of I, and $n^* = |V^*(D)|$. If D is clean, then there is an $i \in [k]$ such that $|V^*(D/B_i)| \geq n^*(k - s)/k$ and there is a path P in D/B_i from s_i to t_i such that $|V^*(P)| \leq |V^*(D/B_i)|/2$.*

Proof. Consider the undirected bipartite graph H with vertex set $\{b_i \mid i \in [k]\} \cup V^*(D)$, where each b_i is a new vertex. Add to H one edge linking a vertex b_i to a vertex v if and only if $v \in B_i$. Now, the size of each B_i is exactly the degree $\deg_H(b_i)$ of b_i in H, and the size of $|I_v|$, for $v \in V^*(D)$, is exactly the degree of v in H. Also notice that $|I_v| \leq s$ for every $v \in V^*(D)$, since D is clean for I. As H is bipartite, we have that

$$\sum_{i\in[k]} \deg_H(b_i) = \sum_{v\in V^*(D)} \deg_H(v), \quad \text{which in turn implies that}$$
$$\sum_{i\in[k]} |B_i| = \sum_{v\in V^*(D)} |I_v| \leq n^* \cdot s. \tag{1}$$

Now, if $|B_i| > n^* \cdot s/k$ for every $i \in [k]$, we have a contradiction with Equation (1). We conclude that there is an $i \in [k]$ such that $|B_i| \leq n^* \cdot s/k$. Thus, $V^*(D/B_i) = n^* - |B_i| \geq n^*(k-s)/k$, as desired.

If there is only one path P from s_i to t_i in D/B_i, then $V^*(P) = \emptyset$ since every path from s_i to t_i in D is contained in $B_i \cup S(I) \cup T(I)$ and the result follows. Thus we can assume that there are at least two paths from s_i to t_i in D/B_i and at least one of them intersects $V^*(D/B_i)$ (see Figure 3). Let $X = (S(I) \cup T(I)) \setminus \{s_i, t_i\}$. By Menger's Theorem, there

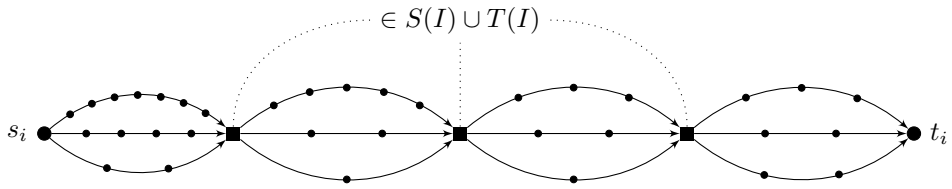

Figure 3 Three paths from s_i to t_i in D/B_i. Square vertices are used to identify vertices in $S(I) \cup T(I)$, which may not be bypassed.

are two internally disjoint paths P_1 and P_2 from s_i to t_i in $(D/B_i)/X$. Without loss of generality, assume that P_1 is the shortest of those two paths, breaking ties arbitrarily. Then $|V^*(P_1)| \leq |V^*(D/B_i)|/2$ since P_1 and P_2 are disjoint, and the result follows. ◀

Figure 4 illustrates the procedure described in Lemma 12. We find a set B_i containing at most $n^* \cdot s/k$ vertices, and bypass all of its vertices in any order. Then we argue that a shortest path from s_i to t_i in D/B_i avoids a large set of vertices in D.

▶ **Lemma 13.** *Let (D, I, k, d, s) be an instance of DEDP, $m = |E(D)|$, and $n^* = V^*(D)$. If D is clean, $n^* \geq 2d(s+1)$, and $k = s+1$, then (D, I, k, d, s) is positive and a solution can be found in time $\mathcal{O}(k \cdot n(n+m))$.*

Proof. Let $\{B_1, \ldots, B_k\}$ be the blocking collection of I and $D'_i = D/B_i$, for $i \in [k]$. By Lemma 12, there is an $i \in [k]$ such that $|V^*(D'_i)| \geq n^*/(s+1)$ and a path P from s_i to t_i such that $V^*(P) \leq |V^*(D'_i)|/2$. Let $D_i = D'_i/V^*(P)$. Now,

$$|V^*(D_i)| \;\geq\; \frac{|V^*(D'_i)|}{2} \;\geq\; \frac{n^*}{2(s+1)}$$

and since $|I \setminus \{(s_i, t_i)\}| = s$, we are free to choose arbitrarily any collection of paths satisfying $I \setminus \{(s_i, t_i)\}$ in D_i. Reversing the bypasses done in D, this collection together with P_i yields a collection of paths satisfying I in D such that all vertices in $V^*(D_i)$ are contained in at most s of those paths. Since $n^* \geq 2d(s+1)$ by hypothesis, we have that $|V^*(D_i)| \geq d$ as required.

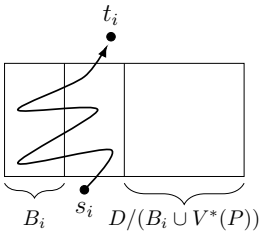

Figure 4 A path P from s_i to t_i avoiding a large part of D.

We can generate the sets B_i in time $\mathcal{O}(n(n+m))$ by deleting a vertex of D and testing for connectivity between s_i and t_i. Thus a solution can be found in time $\mathcal{O}(k \cdot n(n+m))$, as desired. ◀

We are now ready to show how to solve large clean instances of the Disjoint Enough Directed Paths problem in polynomial-time.

▶ **Theorem 14.** *Let (D, I, k, d, s) be a clean instance of DEDP with $|V^*(D)| = n^* \geq d \cdot 2^{k-s} \cdot \binom{k}{s}$. Then (D, I, k, d, s) is positive and a solution can be found in time $\mathcal{O}(k \cdot n^2(n+m))$.*

Proof. Let $\mathcal{B}_0 = \{B_1, \ldots, B_k\}$ be the blocking collection of I. We consider $\mathcal{B}_0$ to be sorted in non-decreasing order by the size of its elements and, by rearranging I if needed, we assume that this order agrees with I. For $i \in [k-(s+1)]$, we construct a sequence of sets $\{D_i, \mathcal{B}_i, \mathcal{P}_i\}$ where $n_i^* = |V^*(D_i)|$ and

- **(i)** $\mathcal{B}_i = \{B_{i+1}, \ldots, B_k\}$;
- **(ii)** $\mathcal{P}_i$ is a collection of paths $\{P_1, P_2, \ldots, P_i\}$ such that P_j is a path from s_j to t_j in D_j, for $j \in [i]$; and
- **(iii)** n_{i-1}^* is large enough to guarantee that we can find a path from s_i to t_i avoiding a large part of D_{i-1}. Formally, we want that

$$n_i^* \geq n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+1)}{2^i \cdot k(k-1)\cdots(k-i+1)}.$$

We begin with $D_0 = D$, $n_0^* = n^*$, and $\mathcal{P}_0 = \emptyset$. Let $D_1' = D_0/B_1$. By applying Lemma 12 with input (D_0, I, k, d, s), we conclude that $|V^*(D_1')| \geq n^*(k-s)/k$ and there is a path P_1 from s_1 to t_1 in D_1' with $|V^*(P_1)| \leq |V^*(D_1')|/2$. Let $D_1 = D_1'/V^*(P_1)$ and $\mathcal{P}_1 = \{P_1\}$. Now,

$$n_1^* \geq \frac{|V^*(D_1')|}{2} \geq \frac{n_0^*(k-s)}{2k}$$

and conditions (i), (ii), and (iii) above hold for $(D_1, \mathcal{B}_1, \mathcal{P}_1)$. Assume that $i-1$ triples have been chosen in this way.

As before, we assume that $\mathcal{B}_{i-1}$ is sorted in non-increasing order by the size of its elements, and that this order agrees with $I \setminus I_{i-1}$. Furthermore, as D_0 is clean, so is D_{i-1}.

Let $D_i' = D_{i-1}/B_i$. Applying Lemma 12 with input $(D_{i-1}, I \setminus I_{i-1}, k-i+1, d, s)$, we conclude that $|V^*(D_i')| \geq n_i^*(k-i+1-s)/(k-i+1)$ and there is a path P_i from s_i to t_i in D_i' with $|V^*(P_i)| \leq |V^*(D_i')|/2$. Let $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{P_i\}$ and $D_i = D_i'/B_i$. Then

$$n_i^* \geq n_{i-1}^* \cdot \frac{k-i+1-s}{2(k-i+1)}$$

and by our assumption that (iii) holds for n_{i-1} it follows that

$$\begin{aligned} n_i^* &\geq n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+2)}{2^{i-1}k(k-1)\cdots(k-i+2)} \cdot \left(\frac{k-s-i+1}{2(k-i+1)}\right) \\ &= n_0^* \cdot \frac{(k-s)(k-s-1)\cdots(k-s-i+1)}{2^i \cdot k(k-1)\cdots(k-i+1)}, \end{aligned}$$

as desired and thus (i), (ii), and (iii) hold for $(D_i, \mathcal{B}_i, \mathcal{P}_i)$. The algorithm ends after iteration $k-(s+1)$. Following this procedure, we construct the collection $\mathcal{P}_{k-(s+1)} = \{P_1, P_2, \ldots, P_{k-(s+1)}\}$ satisfying (ii) and the graph $D_{k-(s+1)}$ with $n_{k-(s+i)}$ satisfying (iii). Noticing that $|I - I_{k-(s+1)}| = s+1$ (that is, only $s+1$ pairs in I are not accounted for in $\mathcal{P}_{k-(s+1)}$), it remains to show that our choice for n^* is large enough so that we are able to apply Lemma 13 on the instance $(D_{k-(s+1)}, I - I_{k-(s+1)}, s+1, d, s)$ of DEDP. That is, we want that $n^*_{k-(s+1)} \geq 2d(s+1)$. By (iii) it is enough to show that

$$n^*_{k-(s+1)} \geq n_0^* \cdot \frac{(k-s)(k-s-1)\cdots 3 \cdot 2}{2^{k-(s+1)} \cdot k(k-1)\cdots(s+3)(s+2)} \geq 2d \cdot (s+1),$$

and rewriting both sides of the fraction as $k!$ and $k!/(s+1)!$, respectively, we get

$$n_0 \cdot \frac{(k-s)!}{2^{k-(s+1)}} \geq 2d \cdot (s+1) \cdot \frac{k!}{(s+1)!} = \frac{2d \cdot k!}{s!},$$

which holds for

$$n_0 \geq \left(\frac{2^{k-(s+1)} \cdot 2d \cdot (s+1)}{(s+1)!}\right) \cdot \left(\frac{k!}{(k-s)!}\right) = d \cdot 2^{k-s} \cdot \binom{k}{s},$$

as desired.

Applying Lemma 13 with input $(D_{k-(s+1)}, I \setminus I_{k-(s+1)}, s+1, d, s)$ yields a collection $\hat{\mathcal{P}}$ satisfying $I - I_{k-(s+1)}$ and a set $X \subseteq V(D)$ of size d such that X is disjoint from all paths in $\mathcal{P}_{k-(s+1)}$, since all vertices in $V^*(P)$ were bypassed in $D_{k-(s+1)}$ for every $P \in \mathcal{P}_{k-(s+1)}$, and all vertices in X occur in at most s paths of $\hat{\mathcal{P}}$. We can construct a collection of paths satisfying I from $\hat{\mathcal{P}} \cup \mathcal{P}_{k-(s+1)}$ by reversing all the bypasses done in D and connecting appropriately the paths in the collections. We output this newly generated collection together with X as a solution for (D, I, k, d, s).

For the running time, let $m = |(E(D)|$. We need time $\mathcal{O}(k \log k)$ to order the elements of $\mathcal{B}_0$, $\mathcal{O}(k \cdot n(n+m))$ to find the sets B_i, for $i \in [k]$, and $\mathcal{O}(n+m)$ to find each of the paths $\{P_1, \ldots, P_k\}$. Hence the algorithm runs in time $\mathcal{O}(k \cdot n^2(n+m))$. ◀

We acknowledge that it is possible to prove Theorem 14 without using Lemma 13 by stopping the iteration at the digraph D_{k-s} instead of D_{k-s-1}. However we believe it is easier to present the proof of Theorem 14 by having separate proofs for the iteration procedure (Lemma 12) that aims to generate an instance of DEDP for which we can apply our base case (Lemma 13).

Since any instance can be made clean in polynomial time, the kernelization algorithm for (k, d, s)-DEDP follows easily. Given an instance (D, I, k, d, s), we bypass all congested vertices of D to generate D'. If $|V^*(D')|$ is large enough to apply Theorem 14, the instance is positive and we can find a solution in polynomial time. Otherwise, we generated an equivalent instance (D', I, k, d, s) with $|V(D')|$ bounded from above by a function depending on k, d, and s only. As we restrict $|S(I) \cup T(I)| \leq 2k$, if D is clean and $V(D) \geq d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ we get the desired bound for $|V^*(D)|$. Thus, the following is a direct corollary of Theorem 14.

▶ **Theorem 15.** *There is a kernelization algorithm running in time $\mathcal{O}(k \cdot n^2(n+m))$ that, given an instance (D, I, k, d, s) of DEDP, outputs either a solution for the instance or an equivalent instance (D', I, k, d, s) with $|V(D')| \leq d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$.*

5 Concluding remarks

We introduced the Disjoint Enough Directed Paths problem and provided a number of hardness and algorithmic results, summarized in Table 1. Several questions remain open.

We showed that DEDP is NP-complete for every fixed $k \geq 3$ and $s \geq 1$. We do not know whether DEDP is also NP-complete for $k = 2$ and $s = 1$.

We provided an algorithm running in time $\mathcal{O}(n^{d+2} \cdot k^{d \cdot s})$ to solve the problem. This algorithm tests all partitions of a given $X \subseteq V(D)$ in search for one that respects some properties. Since there are at most $\binom{n}{d}$ subsets of $V(D)$ of size d, this yields an XP algorithm. The second term on the time complexity comes from the number of partitions of X we need to test. The problem may become easier if X is already given or, similarly, if d is a constant. In other words, is the (s)-DEDP problem FPT for fixed d?

Our main result is a kernel with at most $d \cdot 2^{k-s} \cdot \binom{k}{s} + 2k$ vertices. The natural question is whether the problem admits a polynomial kernel with parameters k, d, and s, or even for fixed s. Notice that if there is a constant ℓ such that $k - s = \ell$, then the size of the kernel is $d \cdot 2^{\ell} \cdot k^{\ell}$, which is polynomial on d and k. The case $s = 0$ is also particularly interesting, as DEDP with $s = 0$ is equivalent to the Steiner Network problem. In this case, we get a kernel of size at most $d \cdot 2^k + 2k$.

While we do not know whether (k, d, s)-DEDP admits a polynomial kernel, at least we are able to prove that a negative answer for $s = 0$ is enough to show that (k, d, s)-DEDP is unlikely to admit a polynomial kernel for any value of $s \geq 1$ when k is "far" from s, via the following polynomial time and parameter reduction.

▶ Remark 16. For any instance $(D, I, k, d, 0)$ of DEDP and integer $s > 0$, one can construct in polynomial time an equivalent instance (D, I', k', d, s) of DEDP with $k' = k \cdot d \cdot s + 1$.

Proof. For a request set I in D, let I' be the request set on D formed by $k \cdot d \cdot s + 1$ copies of each pair in I and $k' = k \cdot d \cdot s + 1$. We claim that an instance $(D, I, k, d, 0)$ of DEDP is positive if and only if the instance $(D, I', k'd, s)$ also of DEDP is positive.

Any solution for the first instance is also a solution for the second, and thus the necessity holds. For the sufficiency, let X be a s-viable set for (D, I', k', d, s) with certifying collection $\mathcal{P}'$. Since $k' = k \cdot d \cdot s + 1$ and at most $d \cdot s$ paths in $\mathcal{P}$ can intersect X, for each pair $(s, t) \in I$ there is path $P \in \mathcal{P}$ from s to t in $D - X$. Choosing all such paths we construct a collection $\mathcal{P}'$ satisfying I in $D - X$ and the result follows. ◀

In the undirected case, the Steiner Tree problem is unlikely to admit a polynomial kernel parameterized by k and c, with $c = n - d$ (in other words, the size of the solution); a simple proof for this result can be found in [7, Chapter 15]. Even if we consider a stronger parameter (that is, d instead of c), dealing with directed graphs may turn the problem much harder. We also remark that the problem admits a polynomial kernel in the undirected case if the input graph is planar [22]. It may also be the case for directed graphs.

References

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

2 Júlio Araújo, Victor A. Campos, Carlos Vinícius G. C. Lima, Vinícius Fernandes dos Santos, Ignasi Sau, and Ana Silva. Dual Parameterization of Weighted Coloring. In *Proc. of the 13th International Symposium on Parameterized and Exact Computation, (IPEC)*, volume 115 of *LIPIcs*, pages 12:1–12:14, 2018. doi:10.4230/LIPIcs.IPEC.2018.12.

3 Manu Basavaraju, Mathew C. Francis, M. S. Ramanujan, and Saket Saurabh. Partially polynomial kernels for set cover and test cover. *SIAM Journal on Discrete Mathematics*, 30(3):1401–1423, 2016. doi:10.1137/15M1039584.

4 Khristo N. Boyadzhiev. Close Encounters with the Stirling Numbers of the Second Kind. *Mathematics Magazine*, 85(4):252–266, 2012. doi:10.4169/math.mag.85.4.252.

5 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013. doi:10.1137/12086217X.

6 Benny Chor, Mike Fellows, and David W. Juedes. Linear Kernels in Linear Time, or How to Save k Colors in $O(n^2)$ Steps. In *Proc. of the 30th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 3353 of *LNCS*, pages 257–269, 2004. doi:10.1007/978-3-540-30559-0_22.

7 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.

8 Marek Cygan, Daniel Marx, Marcin Pilipczuk, and Michal Pilipczuk. The Planar Directed k-Vertex-Disjoint Paths Problem Is Fixed-Parameter Tractable. In *Proc. of the IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, volume 1, pages 197–206, 2013. doi:10.1109/FOCS.2013.29.

9 Rod Downey and Michael. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

10 Rong-chii Duh and Martin Fürer. Approximation of k-Set Cover by Semi-Local Optimization. In *Proc. of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 256–264, 1997. doi:10.1145/258533.258599.

11 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), 2017*, pages 36:1–36:12, 2017. doi:10.4230/LIPIcs.ESA.2017.36.

12 Andreas Emil Feldmann and Dániel Marx. The complexity landscape of fixed-parameter directed Steiner network problems. In *Proc. of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 27:1–27:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.27.

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

14 Thor Johnson, Neil Robertson, Paul Seymour, and Robin Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82(01):138–154, 2001. doi:10.1006/jctb.2000.2031.

15 Mark Jones, Daniel Lokshtanov, M. S. Ramanujan, Saket Saurabh, and Ondřej Suchý. Parameterized Complexity of Directed Steiner Tree on Sparse Graphs. *SIAM Journal on Discrete Mathematics*, 31(2):1294–1327, 2017. doi:10.1137/15M103618X.

16 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 ACM Symposium on Theory of Computing (STOC)*, pages 70–78, 2014. doi:10.1145/2591796.2591876.

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

18 Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Magnus Wahlström. Fixed-parameter tractability of multicut in directed acyclic graphs. *SIAM Journal on Discrete Mathematics*, 29(1):122–144, 2015. doi:10.1137/120904202.

19 James F. Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3):31–36, 1975. doi:10.1145/1061425.1061430.

20 Karl Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927. URL: http://eudml.org/doc/211191.

21 Daniel Mölle, Stefan Richter, and Peter Rossmanith. Enumerate and expand: improved algorithms for connected vertex cover and tree cover. *Theory of Computing Systems*, 43(2):234–253, 2008. doi:10.1007/s00224-007-9089-3.

22 Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan Van Leeuwen. Network Sparsification for Steiner Problems on Planar and Bounded-Genus Graphs. *ACM Transactions on Algorithms*, 14(4):53:1–53:73, 2018. doi:10.1145/3239560.

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

24 Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM Journal on Computing*, 23(4):780–788, 1994. doi:10.1137/S0097539792224061.

25 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.

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