Parameterized Tractability of Edge-Disjoint Paths on Directed Acyclic Graphs*

Aleksandrs Slivkins†

June 2003
Revised: July 2007, September 2008

Abstract

Given a graph and terminal pairs (s_i, t_i), $i \in [k]$, the edge-disjoint paths problem is to determine whether there exist $s_i t_i$ paths, $i \in [k]$, that do not share any edges. We consider this problem on acyclic digraphs. It is known to be NP-complete and solvable in time $n^{O(k)}$ where n is the number of nodes. It has been a long-standing open question whether it is fixed-parameter tractable in k, i.e. whether it admits an algorithm with running time of the form $f(k)\, n^{O(1)}$. We resolve this question in the negative: we show that the problem is $W[1]$-hard, hence unlikely to be fixed-parameter tractable. In fact it remains $W[1]$-hard even if the demand graph consists of two sets of parallel edges.

On a positive side, we give an $O(m+k^{O(1)}\, k!\, n)$ algorithm for the special case when G is acyclic and $G + H$ is Eulerian, where H is the demand graph. We generalize this result (1) to the case when $G + H$ is "nearly" Eulerian, (2) to an analogous special case of the unsplittable flow problem, a generalized version of disjoint paths that has capacities and demands.

Keywords. Disjoint paths, fixed-parameter tractability, W[1]-hardness, Eulerian graphs, unsplittable flow.

AMS subject classification. Primary classification: 68Q25 (CS/ Theory of Computing: analysis of algorithms and problem complexity). Secondary classifications: 68R10 (CS/ Discrete Mathematics in relation to CS: Graph Theory), 68Q17 (CS/ Theory of Computing: computational difficulty of problems), 05C38 (Combinatorics/ Graph Theory: paths and cycles), 05C45 (Combinatorics/ Graph Theory: Eulerian graphs).

*Preliminary version of this paper appeared in the *11th Annual European Symposium on Algorithms*, 2003. This work has been completed while the author was a graduate student at Cornell University. This work has been supported in part by a David and Lucile Packard Foundation Fellowship and NSF ITR/IM Grant IIS-0081334 of Jon Kleinberg.

†Microsoft Research, Mountain View, CA 94043. Email: *slivkins at microsoft.com*.

1 Introduction

Given a graph G and k pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of terminals, the edge-disjoint paths problem is to determine whether there exist $s_i t_i$ paths that do not share any edges. It is one of the earliest studied NP-complete problems [13, 9]. Disjoint paths problems have a great theoretical and practical importance; see [18, 14, 12, 23, 8] for a comprehensive survey.

The problem for a bounded number of terminals have been of particular interest. For undirected graphs, Shiloach [20] gave an efficient polynomial-time algorithm for $k = 2$, and Robertson and Seymour [17] proved that the general problem is solvable in time $f(k)\, n^3$. The directed edge-disjoint paths problem was shown NP-hard even for $k = 2$ by Fortune et al. [7]. On acyclic digraphs the problem is known to be NP-complete (Even et al. [5]) and solvable in time $O(knm^k)$ [7].[1]

Since 1980 it has been an interesting open question whether a better algorithm is possible for acyclic graphs. We should not hope for a polynomial-time algorithm, but can we get rid of k in the exponent and get a running time of $f(k)\, n^c$ for some constant c, as Robertson and Seymour do for the undirected case? Such algorithms are called *fixed-parameter tractable*.

We resolve this open question in the negative using the theory of fixed-parameter tractability due to Downey and Fellows [4]. Specifically, we show that the directed edge-disjoint paths problem on acyclic graphs is $W[1]$-hard in k.

Fixed-parameter tractability. A problem is *parameterized* by $k \in \mathbb{N}$ if its input is a pair (x, k). Many NP-hard problems can be parameterized in a natural way; e.g. the edge-disjoint paths problem can be parameterized by the number of paths. Efficient solutions for small values of the parameter might be useful. For example, if the best-known solution is $O(2^n)$ and k is the parameter, then an $O(n^k)$ running time might be a big improvement, and $O(n\, 2^k)$ is even better. Call a decision problem P *fixed-parameter tractable* in k if there is an algorithm that for every input (x, k) decides whether $(x, k) \in P$ and runs in time $f(k)\, |x|^c$ for some constant c and some computable function f.

Proving that some NP-complete parameterized problem is *not* fixed-parameter tractable would imply that $\mathrm{P} \neq \mathrm{NP}$. However, Downey and Fellows [4] developed a technique for showing *relativized* fixed-parameter intractability. They use reductions similar to those for NP-completeness. Suppose there is a constant c and computable functions f, g such that there is a reduction that maps every instance (x, k) of problem P to an instance $(y, f(k))$ of problem Q, running in time $g(k)\, |x|^c$ and mapping "yes" instances to "yes" instances and "no" instances to "no" instances (we call it a *fixed-parameter reduction*). Then if P is fixed-parameter intractable then so is Q.

There are strong reasons to believe that the problem k-CLIQUE of deciding for a given undirected graph G and an integer k whether G contains a clique of size k is not fixed-parameter tractable [4]. Recently Downey et al. [3] gave a simpler (but weaker) alternative justification based on the assumption that there is no algorithm with running time $2^{o(n)}$ that determines, for a Boolean circuit of total description size n, whether there is a satisfying input vector.

Existence of a fixed-parameter reduction from k-CLIQUE to some problem P is considered to be an evidence of fixed-parameter intractability of P. Problems for which such reduction exists are called $W[1]$-hard, for reasons beyond the scope of this paper. For a thorough treatment of fixed-parameter tractability see Downey and Fellows [4] and a more recent account by Flum and Grohe [6].

Our contributions: disjoint paths. All routing problems in this paper are parameterized by the number k of terminal pairs. Given a digraph $G = (V, E)$ and terminal pairs $\{s_i, t_i\}$ the *demand graph* H is a digraph on a vertex set V with k edges $\{t_i s_i\}$. Note that H can contain parallel edges.

[1] Here and throughout the paper, n is the number of nodes and m is the number of edges.

Our main result is that the directed edge-disjoint paths problem on acyclic graphs is $W[1]$-hard. In fact, we show that it is so even if the demand graph H consists of two sets of parallel edges; this case was known to be NP-complete [5]. Our proof carries over to the node-disjoint version of the problem.

On the positive side, recall that for a general demand graph H the problem is solvable in time $n^{O(k)}$ by an algorithm in [7]. We show a special case which is still NP-complete but fixed-parameter tractable. We focus on Eulerian digraphs, a natural and well-known class of digraphs. (A digraph is called *Eulerian* if for each vertex the in-degree is equal to the out-degree.) Specifically, consider the directed edge-disjoint paths problem if G is acyclic and $G + H$ is Eulerian. This problem is NP-complete, even if H consists of three sets of parallel edges [22]. We give an algorithm with a running time $O(m + k^{O(1)}\, k!\, n)$.[2] We extend this result to the case when G is acyclic and $G + H$ is *nearly* Eulerian.

Our contributions: unsplittable flows. We consider the unsplittable flow problem [10], a generalized version of disjoint paths that has capacities and demands. The instance is a triple (G, H, w) where w is a function from $E(G \cup H)$ to positive reals and $w(t_i s_i)$ is the demand on the i-th terminal pair. The question is whether there are $s_i t_i$ paths such that for each edge e of G the capacity w_e is greater or equal to the sum of demands of all paths that come through e. The edge-disjoint paths problem is a special case of the unsplittable flow problem with $w \equiv 1$.

The unsplittable flow problem can model a variety of problems in virtual-circuit routing, scheduling and load balancing [10, 15]. There has been a number of results on approximation [10, 15, 2, 21, 1]. Most relevant to this paper is the result of Kleinberg [11] that the problem is fixed-parameter tractable on undirected graphs if all capacities are 1 and all demands are at most $\frac{1}{2}$.

We show that the unsplittable flow problem is $W[1]$-hard on acyclic digraphs even if H is a set of parallel edges. If furthermore all capacities are 1 the problem is still NP-hard [10] (since for a two-node input graph with multiple edges it is essentially a bin-packing problem). We show it is fixed-parameter tractable with the running time of $O(e^k)$ plus one max-flow computation. However, the problem becomes $W[1]$-hard again if there are (a) *three* sink nodes, even if all demands are at most $\frac{1}{2}$, (b) two source nodes and two sink nodes, even if all demands are *exactly* $\frac{1}{2}$. This should be contrasted with the above-mentioned result of [11].

Moreover we show that similarly to disjoint paths, the unsplittable flow problem (a) can be solved in time $O(knm^k)$ if G is directed acyclic, (b) becomes fixed-parameter tractable if furthermore $G + H$ is *Eulerian under* w, that is if for each node the total weight of incoming edges is equal to the total weight of outgoing edges. The running time for the latter case is $O(m + k^{4k}n)$.

Notation. Terminals s_i, t_i are called sources and sinks, respectively. Each terminal is located at some node; we allow multiple terminals to be located at the same node. We call a node at which one or more sources is located a *source node*. Similarly a node with one or more sinks is a *sink node*. Note that a node can be both a source and a sink node. Both source and sink nodes are called *terminal nodes*. If no confusion arises, we may use a terminal name to refer to the node at which the terminal is located.

We parameterize all routing problems by the number k of terminal pairs. We denote the number of nodes and edges in the input graph G by n and m respectively.

Map of the paper. In Section 2 we present our hardness results, and Section 3 is on the algorithmic results. We conclude in Section 4.

[2] An alternative way to see that this problem is fixed-parameter tractable is to show that it is equivalent to the edge-disjoint paths problem on the undirected version of G [22], so the undirected disjoint paths algorithm of Robertson and Seymour [17] applies. However, as observed in [8], their algorithm is extremely complicated and completely impractical even for $k = 3$.

2 Hardness results

We start with our main result on the hardness of the edge-disjoint paths problem.

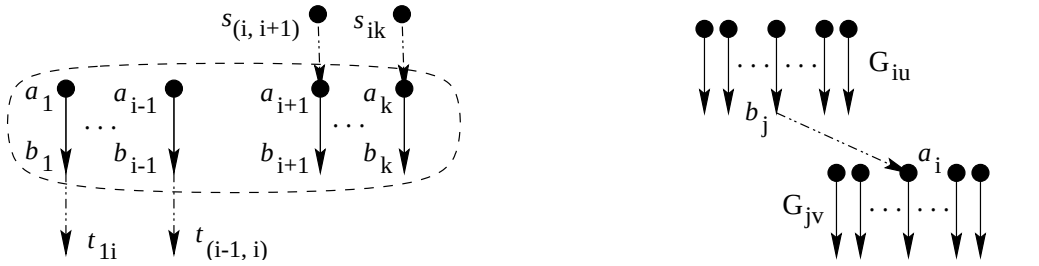
Theorem 2.1 *The edge-disjoint paths problem is $W[1]$-hard on acyclic digraphs.*
Proof: We define a fixed-parameter reduction from k-CLIQUE to the directed edge-disjoint paths problem on acyclic digraphs. Let (G, k) be the instance of k-CLIQUE, where $k \in \mathbb{N}$ and $G = (V, E)$ is an undirected graph without loops. We construct an equivalent instance (G', k') of the directed edge-disjoint paths problem where G' is a directed acyclic graph and $k' = k(k+1)/2$. Denote $[n] = \{1 \dots n\}$ and assume $V = [n]$.

Idea. We create a $k \times n$ array of identical gadgets. Intuitively we think of each row as a copy of V. For each row there is a path ('selector') that goes through all gadgets, skipping at most one and killing the rest, in the sense that other paths cannot route through them. This corresponds to selecting one gadget (and hence one vertex of G) from each row. The selected vertices form a multi-set of size k. We make sure that we can select a given multi-set if and only if it is a k-clique in G. Specifically, for each pair of rows there is a path ('verifier') that checks that the vertices selected in these rows are connected in G. Note that this way we don't need to check separately that the selected vertices are distinct.

Construction. We'll use k paths P_i ('selectors') and $\binom{k}{2}$ paths P_{ij}, $i < j$ ('verifiers'). Denote the terminal pairs by $s_i t_i$ and $s_{ij} t_{ij}$ respectively. Selector P_i will select one gadget from row i; verifier P_{ij} will verify that there is an edge between the vertices selected in rows i and j.

Denote gadgets by G_{iu}, $i \in [k]$, $u \in V$. The terminals are distinct vertices not contained in any of the gadgets. There are no edges from sinks or to sources. We draw the array of gadgets so that row numbers increase downward, and column numbers increase to the right. Edges between rows go *down*; within the same row edges go *right*.

We start with the part of construction used by verifiers. Each gadget G_{iu} consists of $k-1$ parallel paths $(a_r, b_r) = (a_r^{(iu)}, b_r^{(iu)})$, $r \in [k] - \{i\}$. (We will omit the superscripts when they are clear from the context.) For each s_{ij} there are edges $s_{ij} a_j$ to every gadget in row i. For each t_{ij} there are edges $b_i t_{ij}$ from every gadget in row j (Fig. 1a); there will be no more edges from s_{ij}'s or to t_{ij}'s. To express the topology of G, for each edge uv in G and each $i < j$ we create an edge from b_j in G_{iu} to a_i in G_{jv} (Fig. 1b). There will be no more edges or paths between rows. The following claim explains what we have done so far.



(a) Gadget G_{iv} (b) G_{iu} connected to G_{jv}, $i < j$.

Figure 1: Gadgets and verifiers.

Claim 2.2

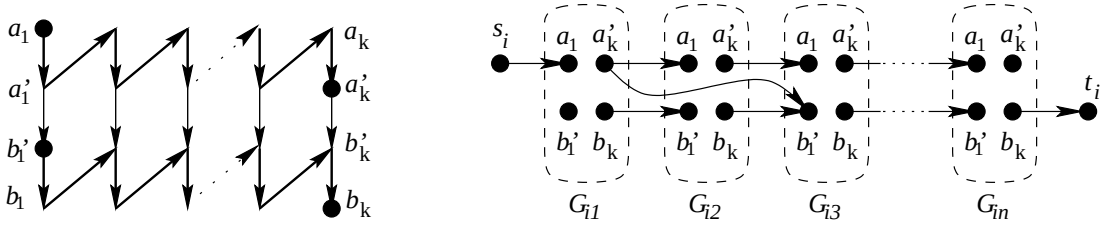
(a) Suppose we erase all gadgets in rows i and j except G_{iu} and G_{jv}. Then an $s_{ij}t_{ij}$ path exists if and only if $uv \in E$.

(b) Suppose we select one gadget in each row and erase all others. Then there exist edge-disjoint paths from s_{ij} to t_{ij} for each $i, j \in [k]$, $i < j$, if and only if the selected gadgets correspond to a k-clique in G.

We could prove Claim 2.2 right away, but we will finish the construction first. Recall that each gadget G_{iu} consists of $k-1$ parallel wires $(a_r, b_r) = (a_r^{(iu)}, b_r^{(iu)})$, $r \in [k] - \{i\}$. Each wire is a simple path of length 3: (a_r, a'_r, b'_r, b_r) (Fig. 2a). Let "level 1" be the set of all a_r and a'_r (in all wires and in all gadgets). Let "level 2" be the set of all b'_r and b_r. Each selector enters its row at level 1. The idea is that the only way it can skip a gadget is by going from level 1 to level 2, so, since within a given row there is no path back to level 1, at most one gadget can be skipped. The remainder of the construction makes this concrete.

First we complete the construction of a single gadget (Fig. 2a). In each gadget G_{iu} there are two edges from each wire r to the next one, one for each level. For $r \neq i-1, i$ these are $(a'_r,\ a_{r+1})$ and $(b_r,\ b'_{r+1})$ (note that there is no wire i). The edges between wires $i-1$ and $i+1$ are $(a'_{i-1},\ a_{i+1})$ and $(b_{i-1},\ b'_{i+1})$.

It remains to connect gadgets within a given row i (Fig. 2b). There are edges from s_i to a_1 in G_{i1}, and from b_k in G_{in} to t_i. There are two edges from each gadget to the next one, one for each level: from a'_k to a_1 and from b_k to b'_1. Finally, there are *jumps* over any given gadget in the row: an edge from s_i to b'_1 of G_{i2} jumps over G_{i1}, edges from a'_k of $G_{(i,u-1)}$ to b'_1 of $G_{(i,u+1)}$ jump over G_{iu}, and an edge from a'_k in $G_{(i,n-1)}$ to t_i jumps over G_{in}.



(a) A single gadget (entry and exit points are circled)

(b) The i-th row (only one jump edge is shown)

Figure 2: Additional wiring for selectors.

Proof of correctness. First we check that our construction is acyclic. It suffices to provide a topological ordering. For $i \in [k]$ and $j \in [2]$, let Q_{ij} be the ordering of vertices in the level j path in row i, i.e. Q_{i1} is the unique path from a_1 in G_{i1} to a'_k in G_{in} and Q_{i2} is the unique path from b'_1 in G_{i1} to b_k in G_{in}. Then the required ordering is given by

$$(\text{all sources};\ Q_{11}, Q_{12};\ Q_{21}, Q_{22};\ \ldots\ ; Q_{k1}, Q_{k2};\ \text{all sinks}).$$

Now we prove Claim 2.2. We stated part (a) for intuition only. The proof is obvious. For part (b), the 'if' direction is now straightforward since each gadget assigns a separate wire to each verifier than can potentially route through it, and the wires corresponding to a given verifier are connected in the right way. For the 'only if' direction, note that there is at most one edge between any given pair of gadgets in different rows, so the total number of edges between the selected gadgets is at most $\binom{k}{2}$. In fact it is exactly $\binom{k}{2}$ since each verifier has to use at least one of these edges. Therefore any pair of selected gadgets is connected, which happens if and only if the corresponding vertices are connected in G. Claim proved.

Claim 2.3 *For each possible $s_i t_i$ path there is a gadget such that verifiers cannot enter any other gadget in row i.*

Proof: All edges between rows go "down", so if P_i ever leaves row i, it can never come back up. Thus P_i must stay in row i and visit each gadget in it successively, possibly jumping over one of them. If P_i enters a given gadget at a_1, it can either route through level 1 and exit at a'_k, or switch to level 2 somewhere in the middle and exit at b_k. If P_i enters at b'_1, it must route through level 2 and exit at b_k.

P_i starts out at level 1. If it never leaves level 1 then it uses up every edge $a_r a'_r$ (so verifiers cannot enter any gadget in the row). Else it switches to level 2, either within a gadget or by jumping over a gadget, call it G_{iu}. To the left of G_{iu} all edges $a_r a'_r$ are used by P_i, so verifiers cannot enter. To the right of G_{iu} the selector uses all edges $b'_r b_r$, so verifiers cannot exit the row from any G_{iv}, $v > u$. If a verifier enters such gadget it never leaves the row since within a row inter-gadget edges only go right. Therefore verifiers cannot enter gadgets to the right of G_{iu}, either. □

We need to prove that our construction is a positive instance of the directed edge-disjoint paths problem if and only if (G, k) is a positive instance of k-CLIQUE. For the "if" direction, let $u_i \dots u_k$ be a k-clique in G, let each selector P_i jump over G_{iu_i} and apply Claim 2.2b. For the "only if" direction, suppose our construction has a solution. By Claim 2.3 verifiers use only one gadget in each row (that is, all verifiers use the same gadget). Therefore by Claim 2.2b these gadgets correspond to a k-clique in G. This completes the proof of Theorem 2.1. □

Now we extend our result by restricting the demand graph.

Theorem 2.4

(a) The edge-disjoint paths problem is $W[1]$-hard on acyclic digraphs even if the demand graph consists of two sets of parallel edges.

(b) The unsplittable flow problem is $W[1]$-hard on acyclic digraphs even if the demand graph is a set of parallel edges.

Proof: (Sketch) In the construction from the proof of Theorem 2.1, contract all s_i, s_{ij}, t_i and t_{ij} to s, s', t and t', respectively. Clearly each selector has to start in a distinct row; let P_i be the selector that starts in row i. Since there is only one edge to t from the k-th row, P_{k-1} has to stay in row $k-1$. Iterating this argument we see that each P_i has to stay in row i, as in the original construction. So Claim 2.3 carries over. Each $s't'$ path has to route between some pair of rows, and there are at most $\binom{k}{2}$ edges between selected gadgets. This proves Claim 2.2b and completes part (a).

For part (b) all edges incident to s' or t' and all edges between rows are of capacity 1; all other edges are of capacity 2. Each verifier has demand 1, each selector has demand 2. Contract s' to s and t' to t. □

Kleinberg [11] showed that the *undirected* unsplittable flow problem apparently becomes more tractable when the maximal demand is at most a half of the minimal capacity. The next theorem shows that on acyclic digraphs this does not seem to be the case.

Theorem 2.5 *On acyclic digraphs, if all capacities are 1 and all demands are at most $\frac{1}{2}$, the unsplittable flow problem is $W[1]$-hard even if there are only (a) two source nodes and two sink nodes, (b) one source node and three sink nodes. Moreover, the first result holds even if all demands are* exactly $\frac{1}{2}$.

Proof: (Sketch) We will extend the basic construction from Theorem 2.4 and explain why the modifications do what they are supposed to do.

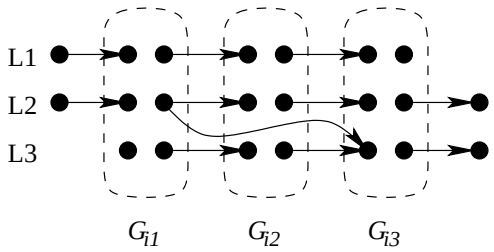
Let's start with part (a): all capacities are 1, all demands are $\frac{1}{2}$, one terminal pair (st) is for selectors, another ($s't'$) is for verifiers. Each row has three levels instead of two, call them L_1, L_2, L_3; two edges from

s enter the row (at L_1, L_2) and two edges to t exit it (at L_2, L_3). There are four selectors per row; jump edges are from L_2 to L_3 (Fig. 3a).

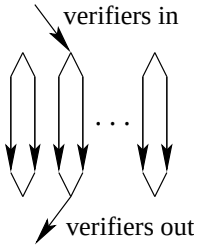
First of all, as in the proof of Theorem 2.4a, all edges to t must be saturated, so each selector must stay in its row. To see that most one gadget can be skipped in a given row, consider some gadget. There are three levels, each of which fits only two of the four selectors. So if neither of the selectors jumps over, each wire in this gadget will have some level occupied by two selectors so that the gadget is blocked (i.e. no verifiers can come through). Therefore if *any* verifiers is to come through, either two selectors jump over, or one jumps over and another switches to L_3 (else there are three selectors on L_1 and L_2, so the gadget is blocked). In either case in the next gadget there are two selectors on L_3. Since selectors cannot leave L_3, the rest of the row is blocked.

For every pair of rows there are now two verifiers instead of one. The wires in each gadget are doubled as shown in Fig. 3b, so that the two verifiers can come through. The inter-row edges are the same as in the basic construction. Claim 2.2b and the rest of the proof follows as in Theorem 2.4a.

For part (b), there is only one source node and three sink nodes: t for selectors, t' for verifiers, and t''. Keep the construction from part (a), merge s and s'. The problem is that now selectors could start on edges designed for verifiers and vice versa. To fix it, we add $k(k-1)/2$ new terminal pairs st'' of demand .2, call them "helpers", and change demand for each verifier to $.4$. Now if two helpers leave s on the same edge, there will not be enough edges from s to pack all the selectors and verifiers. Therefore the only way a helper can leave s is together with two verifiers. So, for each edge sv designed for verifiers, add edge vt''. Since there are $k(k-1)/2$ such edges and each helper has to use one of them, all verifiers must use them, too. □



(a) The i-th row
(n=3; only one jump edges are shown)



(b) A single gadget (in- and out-edges are shown for only one double wire)

Figure 3: Additional wiring for selectors.

3 Algorithmic results

In this section G is an acyclic directed graph on n vertices and H is the demand graph with k edges. Let us consider the edge-disjoint paths problem when $G + H$ is Eulerian. It is interesting to note that in any solution every edge of G must be used by some (and hence by exactly one) $s_i t_i$ path.

We will show that this problem is fixed-parameter tractable. While the problem instance fixes a specific matching between sources and sinks, our algorithm solves the problem for all possible sink-source matchings at once.

Let us introduce the relevant notation. Let S_k be the group of permutations on $[k] = \{1 \dots k\}$. Assuming a fixed numbering $s_1 t_1 \dots s_k t_k$ of terminal pairs, if for some permutation $\pi \in S_k$ there are edge-disjoint $s_i \to t_{\pi(i)}$ paths, $i \in [k]$, then we say that these paths form a *feasible* path allocation and *realize* π, and that π is *feasible*. Let $\Pi(G, H)$ be the set of all feasible permutations. By abuse of notation we consider it to be a $k!$-bit vector.

Theorem 3.1 *Suppose G is acyclic and $G+H$ is Eulerian. Then it is possible to compute $\Pi(G,H)$ in time $O(m + k^{O(1)}\, k!\, n)$. In particular, this solves the directed edge-disjoint paths problem on (G,H).*

Proof: Let $\Pi = \Pi(G,H)$. Recall that in an acyclic graph there must be at least one vertex of zero in-degree; let u be one such vertex in G. Let $v_1 \dots v_r$ be the vertices adjacent to u. Since $G+H$ is Eulerian, there are exactly r sources sitting in u, say $s_{i_1} \dots s_{i_r}$. Therefore each feasible path allocation induces k edge-disjoint paths on $G' = G - u$ such that there is a (unique) path that starts at each v_i. This observation suggests to consider a smaller problem instance (G', H') where we obtain the new demand graph H' from H by moving each s_{i_j} from u to v_j. Note that G' is acyclic and $G'+H'$ is Eulerian. The idea is to compute Π from $\Pi' = \Pi(G', H')$ by gluing the uv_i paths with the paths in the problem instance (G', H').

Formally, let H' be the demand graph associated with terminal pairs $s'_1 t_1 \dots s'_k t_k$ where $s'_i = v_j$ if $i = i_j$ for some j, and $s'_i = s_i$ otherwise. Let $I = \{i_1 \dots i_r\}$ and $S_I \subset S_k$ be the subgroup of all permutations on I extended to identity on $[k] - I$. Letting $\circ$ denote function composition,[3] we claim that

$$\Pi = \{\pi' \circ \sigma : \pi' \in \Pi' \text{ and } \sigma \in S_I\}. \tag{1}$$

Indeed, let $\sigma \in S_I$ and $\pi' \in \Pi'$; let $P'_1 \dots P'_k$ be paths that realize π' in (G', H'). Then

$$P_i = (s_i,\, s'_{\sigma(i)}) \cup P'_{\sigma(i)}$$

is a path from s_i to $t_{\pi(\sigma(i))}$ for all i. (Note that $P_i = P'_i$ for $i \notin I$.) Paths $P_1 \dots P_k$ are edge-disjoint, so $\pi' \circ \sigma \in \Pi$.

Conversely, let $\pi \in \Pi$ and $P_1 \dots P_k$ be paths that realize it. The same paths restricted to G' realize some $\pi' \in \Pi'$. For each $i \in I$ the path P_i goes through some s'_j, say through $s'_{\sigma(i)}$. Let $\sigma(i) = i$ for $i \notin I$. Then $\sigma \in S_I$ and $\pi = \pi' \circ \sigma$. Claim proved.

We compute Π by iterating (1) n times on smaller and smaller graphs. To choose u we maintain the set of vertices of zero in-degree; recomputing it after each iteration takes time $O(k)$. To compute (1) in time $k^{O(1)}k!$ we project each $\pi' \in \Pi'$ to $J = [k] - I$. Specifically, we define the projection $\pi'|_J$ as the permutation on $[k]$ which coincides with π' on J and equals identity on I. We compute $\Pi'' = \{\pi'|_J : \pi' \in \Pi'\}$ in time $k^{O(1)}k!$ by iterating through Π'.[4] Clearly,

$$\Pi = \{\pi'' \circ \sigma : \pi'' \in \Pi'' \text{ and } \sigma \in S_I\}, \tag{2}$$

which can be computed in the claimed running time since $|\Pi'' \times S_I| \le (k-r)!\, r! \le (k-1)!$. □

Let us define the *imbalance* of a graph as $\frac{1}{2}\sum_{\text{nodes } v} |\texttt{indegree}(v) - \texttt{outdegree}(v)|$. Suppose $G+H$ is 'nearly' Eulerian in the sense that its imbalance is small. We can add b new terminal pairs st where s, t are new vertices, along with a few edges from s to G and from G to t, to get a new problem instance (G', H') such that G' is acyclic and $G'+H'$ is Eulerian. It is easy to see that the problem instances (G,H) and (G', H') are in fact equivalent, in the sense that one has a solution if and only if another does (see Theorem 2 in Vygen [22]). Thus we obtain the following result:

Theorem 3.2 *If G is acyclic and the imbalance of $G+H$ is b then the edge-disjoint paths problem on (G,H) can be solved in time $O(m + (k+b)^{O(1)}\, (k+b)!\, n)$.*

Now we will extend the argument of Theorem 3.1 to the unsplittable flow problem. Recall that an instance of the unsplittable flow problem is a triple (G, H, w) where w is a function from $E(G \cup H)$ to positive reals. Let $d_i = w(t_i s_i)$ be the demand on the i-th terminal pair.

[3] That is, $(f \circ g)(x) = f(g(x))$ for all x.

[4] One could, of course, optimize the exponent in $k^{O(1)}$ by using appropriate data structures, etc. However, we think that such optimization is not essential to this paper.

We will need a more complicated version of $\Pi(G, H)$. Let σ and τ be onto functions from $[k]$ to source and sink nodes respectively. Say (σ, τ) is a *feasible pair* if $\sum_{\sigma_i=s} d_i = \sum_{s_i=s} d_i$ for each source node s and $\sum_{\tau_i=t} d_i = \sum_{t_i=t} d_i$ for each sink node t. In other words, a feasible pair rearranges s_i's on the source nodes and t_i's on the sink nodes without changing the total demand on each source or sink node. Say paths $P_1 \dots P_k$ *realize* a feasible pair (σ, τ) if these paths form a solution to the unsplittable flow problem on G with terminal pairs $\sigma_i\tau_i$ and demands w_i. Let $\Pi(G, H, w)$ be the set of all such feasible pairs.

Theorem 3.3 *Let (G, H, w) be an instance of the unsplittable flow problem such that G is acyclic and $G + H$ is Eulerian under w. Then we can compute $\Pi = \Pi(G, H, w)$ in time $O(m + k^{4k}n)$. In particular, this solves the unsplittable flow problem on (G, H, w).*

Proof: (Sketch) The proof is similar to that of Theorem 3.1. Again, letting u be a vertex of zero in-degree in G, the idea is to compute Π from a problem instance on a smaller graph $G' = G - u$.

We derive a problem instance (G', H', w') on k terminal pairs s'_it_i with demands d_i and capacities given by w. Again, letting $v_1 \dots v_r$ be the nodes adjacent to u in G, the new demand graph H' is obtained from H by moving all sources from u to v_i's, arranging them in any (fixed) way such that $G' + H'$ is Eulerian under w'. It is easy to see that we get such arrangement from any set of paths that realizes some feasible pair. If such arrangement exists we can find it using enumeration; else Π is empty.

Similarly to (1), we compute Π from $\Pi(G', H', w')$ by gluing the uv_i paths with paths in G', except now we only consider uv_i paths that respect the capacity constraints on edges uv_i. □

Let us return to the general acyclic digraphs. We extend the $n^{O(k)}$ algorithm of [7] from disjoint paths to unsplittable flows.

Theorem 3.4 *The unsplittable flow problem on acyclic digraphs can be solved in time $O(knm^k)$.*

Proof: We extend the pebbling game from [7]. For each $i \in [k]$ add nodes s'_i and t'_i and infinite-capacity edges s'_is_i and $t_it'_i$. Define the pebbling game as follows. Pebbles $p_1 \dots p_k$ can be placed on edges. Each pebble p_i has weight d_i. The *capacity constraint* is that at any moment the total weight of all pebbles on a given edge e is at most w_e. If a pebble p_i sits on edge e, define the *level* of p_i to be the maximal length of a path that starts with e. Pebble p_i can move from edge uv to edge vw if and only if p_i has the highest level among all pebbles and the capacity constraint on vw will be satisfied after the move. Initially each p_i is on s'_is_i. The game is won if and only if each p_i is on $t_it'_i$.

It is easy to see that the pebbling game has a winning strategy if and only if there is a solution to the unsplittable flow problem (paths in the unsplittable flow problem correspond to trajectories of pebbles). The crucial observation is that if some pebbles visit an edge e then at some moment all these pebbles are on e.

Let G_{state} be the state graph of the pebbling game, with nodes corresponding to possible configurations and edges corresponding to legal moves. The algorithm is to search G_{state} to determine whether the winning configuration is reachable from the starting one. The running time follows since there are m^k possible configurations and at most kn legal moves from each. □

Finally, we consider the case when the demand graph is just a set of parallel edges.

Lemma 3.5 *If the demand graph is a set of parallel edges and all capacities are 1, the unsplittable flow problem on directed or undirected graphs can be solved in time $O(e^k)$ plus one max-flow computation.*[5]

Proof: Let s, t be the source and the sink node respectively. Consider some minimal st-edge-cut C of G and suppose its capacity is greater than the total demand (else there is no solution). Any solution to the unsplittable flow problem solves a bin-packing problem where the demands are packed on the edges of C. If such a packing exists, it can be found by enumeration in time $O(\frac{k^k}{k!}) = O(e^k)$. By a well-known Menger's

[5]Recall that without the restriction on capacities the problem is $W[1]$-hard on acyclic digraphs.

theorem there exist $|C|$ edge-disjoint st-paths. We can route the unsplittable flow on these paths using the packing above. □

4 Conclusions and further directions

We resolve a long-standing open question on tractability of the edge-disjoint paths problem on acyclic digraphs: whether this problem is fixed-parameter tractable in the number of terminal pairs (denoted k), i.e. whether it admits an algorithm with running time of the form $f(k)\, n^{O(1)}$. We prove that this problem is $W[1]$-hard, hence unlikely to be fixed-parameter tractable. Our result fits nicely between the NP-completeness result of [7] and their $n^{O(k)}$ algorithmic result for the same problem. We further strengthen our result by restricting the demand graph. We also consider extensions to (a very restricted version of) the unsplittable flow problem. For both the edge-disjoint paths problem and the unsplittable flow problem, we complement our hardness results with algorithmic (fixed-parameter tractable) results for the case when the input graph G is acyclic and $G + H$ is Eulerian or nearly Eulerian, where H is the demand graph.

Let us mention two directions for further work:

- Our hardness proof for the unsplittable flow problem with parallel demands (Theorem 2.4(b)) requires the input graph to have at least two different capacities. The case when all capacities are 1 is still NP-complete [10]; we conjecture that it is fixed-parameter tractable.
- Given our hardness result for edge-disjoint paths, we hope to address the case when the input graph G is acyclic and planar. This problem is still NP-complete [22], but becomes polynomial if furthermore $G + H$ is planar (this follows from [16], e.g. see [23]). Note that the directed *node*-disjoint paths problem on planar graphs is solvable in polynomial time for every fixed k [19].

Acknowledgments. We thank Jon Kleinberg, Martin Pál and Mark Sandler for valuable discussions and help with the write-up. We thank the journal referees for their comments.

References

[1] G. Baier, E. Köhler and M. Skutella, "On the k-Splittable Flow Problem," *Algorithmica*, **42**(3-4): 231-248 (2005). Preliminary version in *Proc. 10th Annual European Symposium on Algorithms*, 2002.

[2] Y. Dinitz, N. Garg and M. Goemans "On the Single-Source Unsplittable Flow Problem," *Combinatorica* **19**(1): 17-41 (1999). Preliminary version in *Proc. 39th Annual Symposium on Foundations of Computer Science*, 1998.

[3] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto and F. Rosamund, "Cutting Up is Hard to Do: the Parameterized Complexity of k-Cut and Related Problems," *Computing: The Australasian Theory Symposium*, 2003.

[4] R.G. Downey and M.R. Fellows, *Parameterized Complexity*, Springer-Verlag (1999).

[5] S. Even, A. Itai and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Computing*, **5**:691-703 (1976). Preliminary version in *Proc. 16th Annual Symposium on Foundations of Computer Science*, 1975.

[6] J. Flum and M. Grohe, *Parameterized Complexity Theory*, Springer, 2006.

[7] S. Fortune, J. Hopcroft and J. Wyllie, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, **10**:111-121 (1980).

[8] B. Korte, L. Lovász, H-J. Prömel, A. Schrijver, eds., *Paths, Flows and VLSI-Layouts*, Springer-Verlag (1990).

[9] R.M. Karp, "On the computational complexity of combinatorial problems," *Networks*, 5:45-68, 1975.

[10] J. Kleinberg, "Single-source unsplittable flow," *Proc. 37th Annual Symposium on Foundations of Computer Science*, 1996.

[11] J. Kleinberg, "Decision algorithms for unsplittable flow and the half-disjoint paths problem," *Proc. 30th Annual ACM Symposium on the Theory of Computing*, 1998.

[12] J. Kleinberg, "Approximation Algorithms for Disjoint Paths Problems," *Ph.D. Thesis*, M.I.T., 1996.

[13] D.E. Knuth, "Wheels within wheels," *J. of Combinatorial Theory, Series B*, 16:42-46 (1974).

[14] S. G. Kolliopoulos, "Edge-disjoint Paths and Unsplittable Flow," in *Handbook of Approximation Algorithms and Metaheuristics*, ed. T. F. Gonzalez, Chapman & Hall/CRC, 2007.

[15] S.G. Kolliopoulos and C. Stein, "Improved approximation algorithms for unsplittable flow problems," *SIAM J. Comput.* **31**(3): 919-946 (2001). Preliminary version in *Proc. 38th Annual Symposium on Foundations of Computer Science*, 1997.

[16] C.L. Lucchesi and D.H. Younger, "A minimax relation for directed graphs," *J. London Mathematical Society* **17**:369-374 (1978).

[17] N. Robertson and P.D. Seymour, "Graph minors XIII. The disjoint paths problem," *J. Combinatorial Theory Ser. B* **63**:65-110 (1995).

[18] A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency*, Springer-Verlag, Berlin, 2003.

[19] A. Schrijver, "Finding k disjoint paths in a directed planar graph," *SIAM J. Computing* **23**:780-788 (1994).

[20] Y. Shiloach, "A polynomial solution to the undirected two paths problem," *J. of the ACM* **27**:445-456 (1980).

[21] M. Skutella, "Approximating the single source unsplittable min-cost flow problem," *Mathematical Programming Ser. B* **91**(3):493-514 (2002). Preliminary version in *Proc. 41th Annual Symposium on Foundations of Computer Science*, 2000.

[22] J. Vygen, "NP-completeness of some edge-disjoint paths problems," *Discrete Appl. Math.* **61**:83-90 (1995).

[23] J. Vygen, "Disjoint paths," *Rep. #94846*, Research Inst. for Discrete Math., U. of Bonn (1998).