Fault-Tolerant Edge-Disjoint s-t Paths - Beyond Uniform Faults

David Adjiashvili, Felix Hommelsheim, Moritz Mühlenthaler, Oliver Schaudt

Fault-Tolerant Edge-Disjoint s-t Paths – Beyond Uniform Faults

David Adjiashvili
Department of Mathematics, ETH Zürich, Switzerland

Felix Hommelsheim
Department of Mathematics and Computer Science, Universität Bremen, Germany

Moritz Mühlenthaler
Laboratoire G-SCOP, Grenoble INP, Univ. Grenoble-Alpes, France

Oliver Schaudt
Department of Mathematics, RWTH Aachen University, Germany

Abstract

The Edge-disjoint s-t Paths Problem (s-t EDP) is a classical network design problem whose goal is to connect for some $k \geq 1$ two given vertices of a graph under the condition that any $k-1$ edges of the graph may fail. We extend the simple uniform failure model of the s-t EDP as follows: the edge set of the graph is partitioned into *vulnerable*, and *safe* edges, and a set of at most k vulnerable edges may fail, while safe edges do not fail. In particular we study the *Fault-Tolerant Path* (FTP) problem, the counterpart of the Shortest s-t Path problem in this non-uniform failure model as well as the Fault-Tolerant Flow (FTF) problem, the counterpart of s-t EDP. We present complexity results alongside exact and approximation algorithms for both problems. We emphasize the vast increase in complexity of the problems compared to s-t EDP.

2012 ACM Subject Classification Theory of computation → Routing and network design problems; Theory of computation → Network flows; Mathematics of computing → Approximation algorithms

Keywords and phrases graph algorithms, network design, fault tolerance, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.SWAT.2022.5

Related Version *Full Version*: https://arxiv.org/abs/2009.05382

1 Introduction

The *Minimum-Cost Edge-Disjoint s-t Path Problem* (s-t EDP) is a classical network design problem defined as follows. Given an edge-weighted directed graph $D = (V, A)$, two terminal vertices $s, t \in V$ and an integer parameter $k \in \mathbb{Z}_{\geq 0}$, find k edge-disjoint paths connecting s and t with minimum total cost. Equivalently, the problem s-t EDP asks for the minimum cost of connecting two nodes in a network, given that any $k-1$ edges can "fail" and hence be a-posteriori removed from the graph. The assumption here is that faults are *uniform* in the sense that every edge in the graph is equally vulnerable. Our goal is to advance the understanding of network design problems in the presence of *non-uniform* faults. To this end we study a natural generalization of s-t EDP called the *Fault-Tolerant Path* (FTP) problem, in which we partition the set of edges into *vulnerable* and *safe* edges. The task is to find a minimum-cost subgraph of a given graph that contains an s-t path after removing any k vulnerable edges from the graph. Formally, the problem FTP is defined as follows.

Fault-Tolerant Path (FTP)

Instance: edge-weighted directed graph $D = (V, A)$, two vertices $s, t \in V$, set $M \subseteq A$ of *vulnerable* edges, and integer $k \in \mathbb{Z}_{\geq 0}$.

Task: Find minimum-cost set $S \subseteq A$ such that $S \setminus F$ contains an s-t path for every $F \subseteq M$ with $|F| \leq k$.

Observe that if $M = A$ then FTP is exactly s-t EDP. We also study a generalization of s-t EDP with a simpler but still non-uniform fault model: The problem Fault-Tolerant Flow (FTF) asks for $\ell \geq 1$ fault-tolerant disjoint s-t paths, assuming that only a single edge can be a-posteriori removed from the graph:

Fault-Tolerant Flow (FTF)

Instance: edge-weighted directed graph $D = (V, A)$, two vertices $s, t \in V$, set $M \subseteq A$ of *vulnerable* edges, and integer $\ell \in \mathbb{Z}_{\geq 0}$.

Task: Find minimum cost set $S \subseteq A$ such that $S \setminus f$ contains ℓ disjoint s-t paths for every $f \in M$.

1.1 Results

Consider the following well-known polynomial-time algorithm for s-t EDP: Assign unit capacities to all edges in G and find a minimum-cost s-t flow of value k. The integrality property of the LP formulation of the *Minimum-Cost s-t Flow* (MCF) problem guarantees that there is always an integer extreme-point. Such a point corresponds to a set of edges of an optimal solution and can be found in polynomial time (see for example [18]). It is natural to ask whether this approach works also for FTP, which generalizes s-t EDP. We give a negative answer by showing that FTP is NP-hard and hence the existence of a polynomial time algorithm for FTP is unlikely. In fact, the existence of constant-factor approximation algorithms is unlikely even when input graphs are directed acyclic graphs. On the positive side we provide polynomial-time algorithms for arbitrary graphs and $k = 1$ as well as directed acyclic graphs and fixed k.

We furthermore investigate the approximatiblity of FTP using its fractional relaxation FRAC-FTP, which is defined as follows.

Fractional FTP (FRAC-FTP)

Instance: edge-weighted directed graph $D = (V, A)$, two vertices $s, t \in V$, set $M \subseteq A$ of the edges, and integer $k \in \mathbb{Z}_{\geq 0}$.

Task: Find minimum cost capacity vector $x : A \to [0, 1]$ such that for every $F \subseteq M$ with $|F| \leq k$, the maximum s-t flow in $G_F = (V, A \setminus F)$ capacitated by x is at least one.

Observe that by adding the requirement that $x \in \{0, 1\}^A$ to FRAC-FTP, we obtain FTP. Recall that for MCF the value of an optimal integer solution is equal to the value of an optimal fractional solution. We show that in contrast to MCF the integrality gap of FRAC-FTP is bounded by $k + 1$ and that this bound is essentially tight in the sense that there is an infinite family of instances with integrality gap arbitrarily close to $k + 1$. This result also leads to a simple LP-based $(k + 1)$-approximation algorithm for FTP, which we

then combine with an algorithm for the case $k = 1$ to obtain a k-approximation algorithm for FTP. Note that FTP also admits the following simple $(k + 1)$-approximation algorithm: Replace each safe edge with $k + 1$ parallel edges and find $k + 1$ edge-disjoint paths from s to t with minimum cost. It is not clear however how to obtain a k-approximation based on this algorithm, so our LP-based analysis is justified.

The second problem we study is FTF, which asks for $\ell \geq 1$ disjoint s-t paths in the presence of non-uniform single-edge faults. Observe that in the special case of uniform faults (every edge is vulnerable) an optimal solution is a minimum-cost s-t flow of value $k + \ell$ which can be computed in polynomial time. We show that as before the presence of non-uniform faults makes the problem much harder. In fact, it is as hard to approximate as FTP, despite the restriction to single-edge faults (the same result holds for FTF on undirected graphs). On the positive side, we give a simple polynomial-time $(\ell + 1)$-approximation algorithm for FTF which computes a MCF with appropriately chosen capacities.

Note that our positive results for FTP imply a polynomial-time algorithm for FTF when $\ell = 1$. Together with the hardness of FTF in general this motivates the questions how the complexity of FTF depends on the number ℓ of disjoint paths. To this end, we fix ℓ and study the corresponding slice *Fault-Tolerant ℓ-Flow* of FTF. Our main result is a 2-approximation algorithm for this problem. In a nutshell, the algorithm first computes a minimum-cost ℓ-flow and then makes the resulting ℓ disjoint paths fault tolerant by solving the corresponding *augmentation problem*. We solve the augmentation problem by reducing it to a shortest path problem; it is basically a dynamic programming algorithm in disguise. However, the reduction is quite involved: in order to construct the instance of Shortest s-t-Path, we solve at most $n^{2\ell}$ instances of the Min-cost Strongly Connected Subgraphs problem on ℓ terminal pairs, all of which can be done in polynomial time for fixed ℓ. Hence, the overall running time is polynomial for fixed ℓ.

In the light of our approximation results for Fault-Tolerant ℓ-Flow, one may wonder whether the problem may even admit a polynomial-time algorithm (assuming $\mathsf{P} \neq \mathsf{NP}$, say). An indication in this direction is that for a number of problems with a similar flavor, including robust paths [3], robust matchings [16] or robust spanning trees [2], hardness results were obtained by showing that the corresponding augmentation problems are hard. However, our results mentioned above show that this approach does not work for FTF. We show that a polynomial-time algorithm for Fault-Tolerant ℓ-Flow implies polynomial-time algorithms for 1-2-connected Directed 2 Steiner Tree. Whether this problem is NP-hard or not is a long-standing open question.

1.2 Related Work

The shortest path problem is a classical problem in the area of combinatorial optimization and as such, it has received considerable attention also in the context of fault tolerance, see for example [4, 10, 13, 6, 19, 20]. Most of the variants of the Shortest Path Problem studied in these references, as well as FTP and FTF, are subsumed by the Capacitated Survivable Network Design Problem, which due to its generality is hard to approximate even within a factor of $2^{\log^{1-\delta}(n)}$ on directed graphs for any $\delta > 0$ under standard complexity assumptions [7]. The problems FTP and FTF also fit in the framework of *bulk-robustness* introduced by Adjiashvili, Stiller and Zenklusen [3]. In this model, we are given a set of *failure scenarios*, that is, a set of subsets of resources that may fail simultaneously. The task is to find a minimum-cost subset of the resources such that a desired property (e.g., connectivity of a graph) is maintained, no matter which failure scenario materializes. Adjiashvili, Stiller and Zenklusen considered bulk-robust counterparts of the Shortest Path and Minimum

Matroid basis problems. For bulk-robust shortest paths on undirected graphs they give a $O(k + \log n)$-approximation algorithm, where k is the maximum size of a failure scenario. However, the running-time of this algorithm is exponential in k. Note that their bulk-robust shortest path problem generalizes FTP, and therefore the same approximation guarantee holds for FTP. Our approximation algorithm for FTP significantly improves on this bound, on both the approximation guarantee and the running-time. Furthermore, Adjiashvili [1] obtained an LP-based $O(k^2)$-approximation algorithm for bulk-robust shortest paths on planar graphs.

Uniform failure models have been considered for other classical connectivity problems, such as the Minimum Spanning Tree problem: Here, if any k edges of the input graph may fail we obtain the Minimum k-Edge Connected Spanning Subgraph (k-ECSS) problem. For k-ECSS, Gabow, Goemans, Tardos and Williamson [12] gave a polynomial time $(1 + \frac{c}{k})$-approximation algorithm for k-ECSS, for some fixed constant c. The authors also show that for some constant $c' < c$, the existence of a polynomial time $(1 + \frac{c'}{k})$-approximation algorithm implies $\mathsf{P} = \mathsf{NP}$. The more general Generalized Steiner Network problem admits a polynomial 2-approximation algorithm due to Jain [17]. This is also the best known bound for weighted 2-ECSS. Non-uniform single-edge failures for the minimum spanning tree problem have been considered in [2] and a 2-approximation algorithm for this problem has been given recently by Boyd et al. [5]. A problem of a similar flavor but with uniform single-edge faults is Robust Matching Augmentation, which asks for a minimum-cost subgraph such that after the removal of any single edge, the resulting graph contains a perfect matching [16]. This problem is as hard to approximate as Directed Steiner Forest, which is known to admit no $\log^{2-\varepsilon}$-approximation algorithm unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathsf{polylog}(n)})$ [15]. The approximation hardness of FTF is a consequence of this result.

1.3 Notation

We mostly consider directed graphs, which we denote by (V, A), where V is the set of vertices set and A the set of arcs. Undirected graphs are denoted by (V, E) where E is the edge set. When we consider edge-weighted graphs we assume throughout that the weights are non-negative. Let $G = (V, A)$ be a digraph with vulnerable arcs $M \subseteq A$. We denote by $\overline{M} := A \setminus M$ the set of safe arcs. Furthermore, for any set $\emptyset \neq X \subsetneq V$ of vertices of G, we denote by $\delta(X)$ the set of arcs $vw \in A$ such that $v \in X$ and $w \notin X$.

1.4 Organization

The remainder of this paper is organized as follows. In Section 2, we present our results on FTP. We first show that FTP on undirected graphs is a special case of FTP on directed graphs. We provide exact polynomial algorithms for two special cases of FTP in Section 2.1. In Section 2.2 we relate FTP and FRAC-FTP by proving a tight bound on the integrality gap and show how this result leads to a k-approximation algorithm for FTP. In Section 2.3 we study the approximation hardness of FTP. Section 3 contains the results on the problem FTF. Approximation hardness of FTF is shown in Section 3.1. Section 3.2 contains the approximation algorithms for FTF with and without fixed flow value ℓ. Furthermore, in Section 3.3, we relate the complexity of FTF with fixed ℓ to other problems of open complexity. Section 4 concludes the paper and contains open problems.

2 Fault-Tolerant Paths

Assuming non-negative edge-weights, the shortest path problem on undirected graphs is a special case of the same problem on directed graphs: we may replace each undirected edge by two anti-parallel directed edges and conclude that any shortest path in the resulting digraph corresponds to a shortest path in the original undirected graph. We show that this observation extends to FTP. For this purpose we show that any solution to an FTP instance on an undirected graph admits an orientation, such that in each failure scenario a directed s-t paths remains (assuming that if an undirected edge fails, both corresponding anti-parallel arcs fail). As a consequence, the positive results for FTP on directed graphs given in sections 2.1 and 2.2 also hold for FTP on undirected graphs.

▶ **Proposition 1.** *Let $X \subseteq E$ be a feasible solution to an instance of FTP on an undirected graph (V, E). Then there is an orientation $\overrightarrow{X}$ of X such that $(V, \overrightarrow{X} - F)$ contains a directed s-t path for every $F \subseteq M$ with $|F| \leq k$.*

Proof. Let us assume for a contradiction that there is no such orientation. A set Y of (undirected and directed) edges is a *partial orientation of* X if there is a partition of X into sets X_1 and X_2 such that $Y = X_1 \cup \overrightarrow{X_2}$, where $\overrightarrow{X_2}$ is an orientation of X_2. Let Y be a partial orientation of X that maximizes the number of directed edges such that $(V, \overrightarrow{X} - F)$ contains a directed s-t path for every $F \subseteq M$ with $|F| \leq k$. By our assumption, there is at least one undirected edge $e = vw$ in Y. Furthermore, there are two sets $S_1, S_2 \subseteq V$ of vertices, such that $\{s\} \subseteq S_1, S_2 \subseteq V \setminus \{t\}$, $v \in S_1 \setminus S_2$, and $w \in S_2 \setminus S_1$. Note that $vw \in \delta(S_1)$ and $wv \in \delta(S_2)$.

Since e is needed in both directions for Y to be feasible, there is some $F \subseteq M$, $|F| \leq k$ such that $X \setminus F$ contains an s-t path that must leave S_1 via vw. Therefore, the cut $\delta(S_1)$ contains at most $k+1$ edges and all of them except possibly e are vulnerable. The same holds for $\delta(S_2)$ and therefore we have $|\delta(S_1)| = |\delta(S_2)| = k+1$. From the feasibility of Y and the fact that all edges in $\delta(S_1)$ and $\delta(S_2)$ except possibly e are vulnerable, it follows that $|\delta(S_1 \cap S_2)| \geq k+1$ and $|\delta(S_1 \cup S_2)| \geq k+1$. By the submodularity of the cut function $|\delta(\cdot)|$ we have

$$2k+2 = |\delta(S_1)| + |\delta(S_2)| \geq |\delta(S_1 \cap S_2)| + |\delta(S_1 \cup S_2)| \geq 2k+2 \tag{1}$$

so we have equality throughout. Furthermore, $|\delta(\cdot)|$ satisfies the following identity

$$|\delta(S_1)| + |\delta(S_2)| = |\delta(S_1 \cap S_2)| + |\delta(S_1 \cup S_2)| + |A(S_1 \setminus S_2, S_2 \setminus S_1)| + |A(S_2 \setminus S_1, S_1 \setminus S_2)| \ ,$$

but the observation that e is an edge connecting $S_1 \setminus S_2$ and $S_2 \setminus S_1$, together with the fact the we have equality in (1) yields a contradiction to the previous identity. ◀

2.1 Exact Algorithms

In this section we give polynomial-time algorithms for FTP on arbitrary graphs, where at most one edge can fail ($k = 1$) and FTP on directed acyclic graphs (DAGs) for fixed k. We start with the following useful observation.

▶ **Lemma 2.** *Let $I = (D = (V, A), s, t, M, k)$ be an FTP-instance and $X \subseteq A$. Then the following statements are equivalent.*

1. *X is a feasible solution to I.*
2. *The network (V, X) with capacities $c_e = 1$ if $e \in M$ and $c_e = \infty$ otherwise admits an s-t flow of value at least $k+1$.*

Proof. Let $X \subseteq E(D)$. First, suppose that X is a feasible solution to the FTP instance (D, M, k). Suppose for a contradiction that the network $((V, X), c)$ capacitated by $c_e = 1$ if $e \in M$ and $c_e = \infty$ otherwise admits no s-t flow of value at least $k + 1$. Then, by the Max-Flow Min-Cut Theorem, there is some capacitated cut $\delta(V')$ for some $V' \subseteq V$ with $s \in V'$ and $t \notin V'$ such that $c(\delta(V')) < k + 1$. By the definition of c, this implies that $\delta(V')$ does not contain any safe edge. But then $F \coloneqq \delta(V')$ is a cut in (V, X) of size at most k, a contradiction to the feasibility of X.

Now, suppose that X is not a feasible solution to the FTP instance (D, M, k). Then there is some capacitated cut $\delta(V')$ for some $V' \subseteq V$ with $s \in V'$ and $t \notin V'$ such that $c(\delta(V')) \leq k$. But then, by the Max-Flow Min-Cut Theorem, the network $((V, X), c)$ admits no s-t flow of value at least $k + 1$. ◀

We consider the restriction of FTP to $k = 1$. An *s-t bipath* in the graph $D = (V, A)$ is a union of two (not necessarily disjoint) s-t paths $P_1, P_2 \subseteq A$. In the context of 1-FTP we call a bipath $Q = P_1 \cup P_2$ *robust* if $P_1 \cap P_2 \cap M = \emptyset$. Note that every robust s-t bipath Q in G is a feasible solution to the 1-FTP instance. Indeed, consider any vulnerable edge $e \in M$. Since $e \notin P_1 \cap P_2$ it holds that either $P_1 \subseteq Q - e$, or $P_2 \subseteq Q - e$. It follows that $Q - e$ contains some s-t path. The next lemma shows that every feasible solution of the 1-FTP instance contains a robust s-t bipath.

▶ **Lemma 3.** *Every feasible solution S^* to an 1-FTP instance contains a robust s-t bipath.*

Proof. We assume without loss of generality that S^* is a minimal feasible solution with respect to inclusion. Let $Y \subseteq S^*$ be the set of bridges in (V, S^*). From feasibility of S^*, we have $Y \cap M = \emptyset$. Consider any s-t path P in S^*. Let $u_1, \cdots, u_r$ be be the set of vertices incident to $Y = P \cap Y$. Let u_i and u_{i+1} be such that $u_i u_{i+1} \notin Y$. (if such an edge does not exist, we have $Y = P$, which means that P is a robust s-t bipath). Note that S^* must contain two edge-disjoint u_i-u_{i+1} paths L_1, L_2. Taking as the set Y together with all such pairs of paths L_1, L_2 results in a robust bipath. ◀

We conclude from the previous discussion and Lemma 3 that all minimal feasible solutions to the 1-FTP instance are robust bipaths. This observation leads to the simple polynomial-time algorithm for 1-FTP that, using flow-techniques, computes for any pair of vertices u, v the length of i) a min-cost u-v path using only safe edges and ii) two edge-disjoint u-v paths of minimum cost. In a second step the algorithm computes a minimum-cost s-t path in a complete graph with respect to the minimum of the two computed costs. The resulting s-t path corresponds to a min-cost s-t bipath in the original graph and hence, by Lemma 3, an optimal robust s-t path.

▶ **Theorem 4.** *1-FTP admits a polynomial-time algorithm.*

Proof. To solve 1-FTP we need to find the minimum cost robust s-t bipath. To this end let us define two length functions $\ell_1, \ell_2 : V \times V \to \mathbb{R}_{\geq 0}$. For two vertices $u, v \in V$ let $\ell_1(u, v)$ denote the shortest path distance from u to v in the graph $(V, A \setminus M)$, and let $\ell_2(u, v)$ denote the cost of the shortest pair of edge-disjoint u-v paths in D. Clearly, both length functions can be computed in polynomial time (e.g. using flow techniques). Finally, set $\ell(u, v) = \min\{\ell_1(u, v), \ell_2(u, v)\}$. Construct the complete graph on the vertex set V and associate the length function ℓ with it. Observe that by definition of ℓ, any s-t path in this graph corresponds to a robust s-t bipath with the same cost, and vice versa. It remains to find the shortest s-t bipath by performing a single shortest s-t path in the new graph. For every edge uv in this shortest path, the optimal bipath contains the shortest u-v path in $(V, A \setminus M)$ if $\ell(u, v) = \ell_1(u, v)$, and the shortest pair of u-v paths in D, otherwise. ◀

We now consider the problem k-FTP (for fixed $k \in \mathbb{N}$) on layered graphs. The generalization to a directed acyclic graph is done via a standard transformation, which we describe later. Recall that a layered graph $D = (V, A)$ is a graph with a partitioned vertex set $V = V_1 \cup \cdots \cup V_r$ and a set of edges satisfying $A \subset \bigcup_{i\in[r-1]} V_i \times V_{i+1}$. We assume without loss of generality that $V_1 = \{s\}$ and $V_r = \{t\}$. For every $i \in [r-1]$ we let $A_i = A \cap V_i \times V_{i+1}$. We reduce k-FTP to a shortest path problem in a larger graph. The following definition sets the stage for the algorithm.

▶ **Definition 5.** *An* i-configuration *is a vector* $d \in \{0, 1, \cdots, k+1\}^{V_i}$ *satisfying* $\sum_{v\in V_i} d_v = k+1$. *We let* $supp(d) = \{v \in V_i : d_v > 0\}$. *For an* i*-configuration* d^1 *and an* $(i+1)$*-configuration* d^2 *we let*

$$V(d^1, d^2) = supp(d^1) \cup supp(d^2) \quad \text{and} \quad A(d^1, d^2) = A[V(d^1, d^2)].$$

We say that an i*-configuration* d^1 precedes *an* $(i+1)$*-configuration* d^2 *if the following flow problem is feasible. The graph is defined as* $H(d^1, d^2) = (V(d^1, d^2), A(d^1, d^2))$. *The demand vector* ν *and the capacity vector* c *are given by*

$$\nu_u = \begin{cases} -d^1_u & \text{if } u \in supp(d^1) \\ d^2_u & \text{if } u \in supp(d^2) \end{cases} \quad \text{and} \quad c_e = \begin{cases} 1 & \text{if } e \in M \\ \infty & \text{if } e \in E \setminus M, \end{cases}$$

respectively. If d^1 *precedes* d^2 *we say that the* link (d^1, d^2) *exists. Finally, the* cost $\ell(d^1, d^2)$ *of this link is set to be minimum value* $w(A')$ *over all* $A' \subseteq A(d^1, d^2)$, *for which the previous flow problem is feasible, when restricted to the set of edges* A'.

The algorithm constructs a layered graph $\mathcal{H} = (\mathcal{V}, \mathcal{A})$ with r layers $\mathcal{V}_1, \cdots, \mathcal{V}_r$. For every $i \in [r]$ the set of vertices $\mathcal{V}_i$ contains all i-configurations. Observe that since $V_1 = \{s\}$ and $V_r = \{t\}$, we have that $\mathcal{V}_1$ and $\mathcal{V}_r$ contain one vertex each, which we denote by c^s and c^t, respectively. The edges correspond to links between configurations. Every edge is directed from the configuration with the lower index to the one with the higher index. The cost is set according to Definition 5. The following lemma provides the required observation, which immediately leads to a polynomial-time algorithm.

▶ **Lemma 6.** *Every* c^s-c^t *path* P *in* $\mathcal{H}$ *corresponds to a fault-tolerant path* S *with* $w(S) \leq \ell(P)$, *and vise-versa.*

Proof. Consider first a fault-tolerant path $S \subseteq A$. We construct a corresponding c^s-c^t path in $\mathcal{H}$ as follows. Consider any $k+1$ s-t flow f^S, induced by S. Let $p^1, \cdots, p^l$ be a path decomposition of f^S and let $1 \leq \rho_1, \cdots, \rho_l \leq k+1$ (with $\sum_{i\in[l]} \rho_i = k+1$) be the corresponding flow values.

Since D is layered, the path p^j contains exactly one vertex v^j_i from V_i and one edge e^j_i from A_i for every $j \in [l]$ and $i \in [r]$. For every $i \in [r]$ define the i-configuration d^i with

$$d^i_v = \sum_{j\in[l]: v=v^i_j} \rho_i,$$

if some path p^j contains v, and $d^i_v = 0$, otherwise. The fact that d^i is an i-configuration follows immediately from the fact that f^S is a $(k+1)$-flow. In addition, for the same reason d^i precedes d_{i+1} for every $i \in [r-1]$. From the latter observations and the fact that $d^1 = c^s$ and $d^r = c^t$ it follows that $P = d^1, d^2, \cdots, d^r$ is a $c^s - c^t$ path in $\mathcal{H}$ with cost $\ell(P) \leq w(S)$.

Consider next an $c^s - c^t$ path $P = d^1, \cdots, d^r$ with cost $\ell(P) = \sum_{i=1}^{r-1} \ell(d^i, d^{i+1})$. The cost $\ell(d^i, d^{i+1})$ is realized by some set of edges $R_i \subseteq A(d^i, d^{i+1})$ for every $i \in [r-1]$. From Definition 5, the maximal s-t flow in the graph $D' = (V, R)$ is at least $k+1$, where

$R = \cup_{i\in[r-1]} R_i$. Next, Lemma 2 guarantees that there exists some feasible solution $S \subseteq R$, the cost of which is at most $\ell(P)$. In the latter claim we used the disjointness of the sets R_i, which is due the layered structure of the graph G. This concludes the proof of the lemma. ◀

Finally, we observe that the number of configurations is bounded by $O(n^{k+1})$, which implies that k-FTP can be solved in polynomial time on layered graphs.

To obtain the same result for directed acyclic graphs we perform the following transformation of the graph. Let $v_1, \cdots, v_n$ be a topological sorting of the vertices in D. Replace every edge $e = v_iv_j$ $(i < j)$ with a path $p_e = v_i, u^e_{i+1}, \cdots, u^e_{j-1}, v_j$ of length $j-i+1$ by subdividing it sufficiently many times. Set the cost of the first edge on the path to $w'(v_iu^e_{i+1}) = w(v_iv_j)$ and set the costs of all other edges on the path to zero. In addition, create a new set of faulty edges M', which contains all edges in a path p_e if $e \in M$. It is straightforward to see that the new instance of FTP is equivalent to the original one, while the obtained graph after the transformation is layered. We summarize the result as follows.

▶ **Theorem 7.** *There is a polynomial-time algorithm for k-FTP restricted to instances with a directed acyclic graph.*

2.2 Integrality Gap and Approximation Algorithms

In this section we study the natural fractional relaxation FRAC-FTP of FTP and prove a tight bound on its integrality gap. That is, we bound the worst-case ratio of the value of an optimal solution of an FTP instance and the corresponding optimal value of FRAC-FTP. This result also suggests a simple approximation algorithm for FTP with ratio $k+1$. We then combine this algorithm with the algorithm for 1-FTP to obtain a k-approximation algorithm.

Fractional FTP and Integrality Gap

We give the following bound on the integrality gap of FRAC-FTP.

▶ **Theorem 8.** *The integrality gap of FRAC-FTP is at most $k+1$. Furthermore, there exists an infinite family of instances of FTP with integrality gap arbitrarily close to $k+1$.*

Proof. Consider an instance $I = (D, s, t, M, k)$ of FTP. Let x^* denote an optimal solution to the corresponding FRAC-FTP instance, and let $OPT = w(x^*)$ be its cost. Define a vector $y \in \mathbb{R}^A$ as follows.

$$y_e = \begin{cases} (k+1)x_e & \text{if } e \notin M \\ \min\{1, (k+1)x_e\} & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, it holds that $w(y) \leq (k+1)OPT$. We claim that every s-t cut in D with capacities y has capacity of at least $k+1$. Consider any such cut $C \subset A$, represented as the set of edges in the cut. Let $M' = \{e \in M : x^*_e \geq \frac{1}{k+1}\}$ denote the set of faulty edges attaining high fractional values in x^*. Define $C' = C \cap M'$. If $|C'| \geq k+1$ we are clearly done. Otherwise, assume $|C'| \leq k$. In this case consider the failure scenario $F = C'$. Since x^* is a feasible solution it must hold that $\sum_{e\in C\setminus C'} x^*_e \geq 1$. Since for every edge $e \in C \setminus C'$ it holds that $y_e = (k+1)x^*_e$ we obtain

$$\sum_{e\in C\setminus C'} y_e \geq k+1$$

as desired. From our observations it follows that the maximum flow in D with capacities y is at least $k+1$. Finally, consider the minimum cost $(k+1)$-flow z^* in D with capacities defined by

$$c_e = \begin{cases} k+1 & \text{if } e \notin M \\ 1 & \text{otherwise.} \end{cases}$$

From integrality of c and the minimum-cost flow problem we can assume that z^* is integral. Note that $y_e \leq c_e$ for every $e \in A$, hence any feasible $(k+1)$-flow with capacities y is also a feasible $(k+1)$-flow with capacities c. From the previous observation it holds that $w(z^*) \leq w(y) \leq (k+1)OPT$. From Lemma 2 we know that the support of z^* is a feasible solution to the FTP instance. This concludes the proof of the upper bound of $k+1$ for the integrality gap.

To prove the same lower bound we provide an infinite family of instances, containing instances with integrality gap arbitrarily close to $k+1$. Consider a graph with $p \gg k$ parallel edges with unit cost connecting s and t, and let $M = A$. An optimal solution to this FTP instance chooses any subset of $k+1$ edges. At the same time, the optimal solution to FRAC-FTP assigns a capacity of $\frac{1}{p-k}$ to every edge. This solution is feasible, since in every failure scenario, the number of edges that survive is at least $p-k$, hence the maximum s-t flow is at least one. The cost of this solution is $\frac{p}{p-k}$. Taking p to infinity yields instances with integrality gap arbitrarily close to $k+1$. ◀

The proof of Theorem 8 leads to a simple $(k+1)$-approximation algorithm for FTP. However, simply creating k copies of each vulnerable arc and finding a minimum-cost s-t flow of value at least $k+1$ gives a $(k+1)$-approximation as well.

▶ **Proposition 9.** *FTP admits a polynomial-time $(k+1)$-approximation algorithm.*

A k-Approximation Algorithm

We propose an LP-based k-approximation algorithm for FTP that is a refinement of the LP-based approximation algorithm of Proposition 9. Intuitively, the reason why the algorithm of Proposition 9 gives an approximation ratio of $k+1$ is that the capacity of the edges in $A \setminus M$ is set to $k+1$. Therefore, if an s-t flow z^* uses such an edge to its full capacity, the cost incurred is $k+1$ times the cost of the edge. Hence, the best possible lower bound on the cost $w(z^*)$ is $(k+1)\,\mathrm{OPT}_{FRAC}$, where OPT_{FRAC} denotes the optimal value of the corresponding FRAC-FTP instance. To improve the algorithm we observe that each edge which carries a flow of $k+1$ according to z^* is a cut-edge in the obtained solution.

Let $I = (D, s, t, M)$ be an instance of FTP. We begin our analysis by considering a certain canonical flow defined by minimal feasible solutions.

▶ **Definition 10.** *Consider an inclusion-wise minimal feasible solution $S \subseteq A$ to I. A* flow f^S induced by S *is any integral s-t $(k+1)$-flow in D respecting the capacity vector*

$$c_e^S = \begin{cases} 1 & \text{if } e \in S \cap M \\ k+1 & \text{if } e \in S \setminus M \\ 0 & \text{if } e \in A \setminus S. \end{cases}$$

Consider an optimal solution $X^* \subseteq A$ to I and a corresponding induced flow f^*. Define

$$X_{PAR} = \{e \in X^* : f^*(e) \leq k\} \text{ and } X_{BRIDGE} = \{e \in X^* : f^*(e) = k+1\} .$$

As we argued before, every edge in X_{BRIDGE} must be a bridge in $H = (V, X^*)$ disconnecting s and t. Let e_u denote the tail vertex of an edge $e \in A$. Since every edge $e \in X_{BRIDGE}$ constitutes an s-t cut in H, it follows that the vertices in $U = \{e_u : e \in X_{BRIDGE}\} \cup \{s, t\}$ can be unambiguously ordered according to the order in which they appear on any s-t path in H, traversed from s to t. Let $s = u_1, \cdots, u_q = t$ be this order. Except for s and t, every vertex in U constitutes a cut-vertex in H. Divide H into $q-1$ subgraphs $H^1, \cdots, H^{q-1}$ by letting $H^i = (V, Y_i)$ contain the union of all u_i-u_{i+1} paths in H. We observe the following:

▶ **Proposition 11.** *For every $i \in \{1, 2, \ldots, q-1\}$ the set $Y_i \subseteq A$ is an optimal solution to the FTP instance $I_i = (G, u_i, u_{i+1}, M)$.*

Consider some $i \in \{1, 2, \ldots, q-1\}$ and let f_i^* denote the flow f^*, restricted to edges in H^i. Note that f_i^* can be viewed as a u_i-u_{i+1} $(k+1)$-flow. Exactly one of the following cases can occur. Either H^i contains a single edge $e \in A \setminus M$, or $\max_{e \in Y_i} f_i^*(e) \leq k$. In the former case, the edge e is the shortest u_i-u_{i+1} path in $(V, A \setminus M)$. In the latter case we use an algorithm that is similar to the one of Proposition 9 to obtain a k-approximation of the optimal FTP solution on instance I_i. Concretely, the algorithm defines the capacity vector $c'(e) = k$ if $e \notin M$ and $c'(e) = 1$, otherwise, and finds an integral minimum-cost u_i-u_{i+1} $(k+1)$-flow Y^* in D, and returns the support $Y \subseteq A$ of the flow as the solution. The existence of the flow f_i^* guarantees that $w(y^*) \leq w(f_i^*)$, while the fact that the maximum capacity in the flow problem is bounded by k gives $w(Y) \leq kw(y^*)$. It follows that this algorithm approximates the optimal solution to the FTP instance I_i to within a factor k.

The final algorithm uses the algorithm for 1-FTP as a blueprint. However, instead of finding two edge-disjoint u-v paths, the new algorithm solves the aforementioned flow problem. We summarize the main result of this section as follows.

▶ **Theorem 12.** *FTP admits a polynomial-time k-approximation algorithm.*

2.3 Approximation Hardness

We complement our algorithmic results by showing approximation hardness for FTP. An instance of the problem *Directed m-Steiner Tree* (m-DST) is given by a weighted directed graph $D = (V, A)$, a *source* node $s \in V$, a set $T \subseteq V$ of *terminals* and an integer $m \leq |T|$. The goal is to find a minimum-cost arboresence $X \subseteq A$ rooted at s that connects s to m terminals. Halperin and Krauthgamer [15] showed that m-DST cannot be approximated within a factor $\log^{2-\epsilon} m$ for every $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathsf{polylog}(n)})$. We show that the problem m-DST is a special case of FTP.

Given an m-DST instance we construct an instance of FTP as follows. The graph D is augmented by $|T|$ new arcs A' of cost 0 connecting every terminal to a new node t. Finally, we let $M = A'$ and $k = m - 1$. It is readily verified that any fault-tolerant s-t path in the graph so obtained corresponds to a feasible solution to the m-DST instance of the same cost (we may assume that all arcs in A' are in some solution to the FTP instance). This implies the following conditional approximation lower bound for FTP.

▶ **Proposition 13.** *FTP admits no polynomial-time approximation algorithms with ratio $\log^{2-\epsilon} k$ for every $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{polylog(n)})$.*

The reduction can be easily adapted to yield a k^ϵ-approximation algorithm for FTP for the special case that $M \subseteq \{e \in A : t \in e\}$ using the algorithm of Charikar et. al. [8].

We end this discussion by showing that FTP contains as a special case a more general Steiner problem, which we call *Simultaneous Directed m-Steiner Tree* (m-SDST). An input to m-SDST specifies two arc-weighted digraphs $D_1 = (V, A_1, w_1)$ and $D_2 = (V, A_2, w_2)$ on

the same set V of vertices, a source s, a set $T \subseteq V$ of terminals, and an integer $m \leq |T|$. The goal is to find a subset $U \subseteq T$ of m terminals and two arborescences $S_1 \subseteq A_1$ and $S_2 \subseteq A_2$ connecting s to U in the respective graphs, so as to minimize $w_1(S_1) + w_2(S_2)$. m-SDST is seen to be a special case of FTP via the following reduction. Given an instance of m-SDST, construct a graph $D = (V', A)$ as follows. Take a disjoint union of D_1 and D_2, where the direction of every arc in D_2 is reversed. Connect every copy of a terminal $u \in T$ in D_1 to its corresponding copy in D_2 with an additional zero-cost arc e_u. Finally, set $M = \{e_u : u \in T\}$ and $k = m - 1$. A fault-tolerant path connecting the copy of s in D_1 to the copy of s in D_2 corresponds to a feasible solution to the m-SDST instance with the same cost, and vice-versa.

3 Fault-Tolerant Flows

In this section we present our results on the problem FTF. We first give an approximation hardness result and then investigate the complexity of FTF for fixed flow values ℓ. Our main result is a polynomial-time algorithm for the corresponding augmentation problem, which we use to obtain a 2-approximation for Fault-Tolerant ℓ-Flow. We conclude by showing that a polynomial-time algorithm for Fault-Tolerant ℓ-Flow implies polynomial-time algorithms for two problems whose complexity status is open.

3.1 Approximation Hardness of FTF

We show that FTF is as hard to approximate as Directed Steiner Forest by using an approximation hardness result from [16] for the problem Weighted Robust Matching Augmentation. The problem Weighted Robust Matching Augmentation asks for the cheapest edge-set (assuming non-negative costs) to add to a bipartite graph such that the resulting graph is bipartite and contains a perfect matching after a-posteriori removing any single edge. The idea of our reduction is similar to that of the classical reduction from the Bipartite Maximum Matching problem to the Max s-t Flow problem. Note that we may assume that both parts of the input graph have the same size. We add to the graph (U, W, E) on n vertices of a Weighted Robust Matching Augmentation instance I two terminal vertices s and t, and connect s to each vertex of U and each vertex of W to t by an arc of cost 0. Now we add all possible arcs from U to W, marking those as vulnerable that correspond to an edge in E; the costs are according to I. Observe that a fault-tolerant $n/2$-flow corresponds to a feasible solution to the Weighted Robust Matching Augmentation instance after deleting s and t.

▶ **Lemma 14.** *A polynomial-time $f(\ell)$ approximation algorithm for FTF implies a polynomial-time $f(n/2)$-approximation algorithm for Weighted Robust Matching Augmentation, where n is the number of vertices in the Weighted Robust Matching Augmentation instance.*

Proof. In the following it will be convenient to denote by $\overline{E}$ the edge-set of the bipartite complement of a bipartite graph with edge-set E. Let $I = (G, c)$ be an instance of Weighted Robust Matching Augmentation where $G = (U, W, E)$ is a balanced bipartite graph on n vertices and $c \in \mathbb{Z}^{\overline{E}}_{\geq 0}$. Our reduction is similar to the classical reduction from the perfect matching problem in bipartite graphs to the Max s-t Flow problem. We construct in polynomial-time an instance $I' = (D', c', s, t, M)$ of FTF as follows. To obtain the digraph $D' = (V, A)$, we add to the vertex set of G two new vertices s and t and add all arcs from s to U and from W to t. Furthermore, we add all arcs from U to W and consider those that correspond to an edge in E as vulnerable. That is, we let $M \coloneqq \{uw : u \in U, w \in W, uw \in E\}$. To complete the construction of I', we let $\ell = n/2$, and let the arc-costs c' be given by

$$c'_{uw} := \begin{cases} c_{uw} & \text{if } uw \in E(G), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

For $X \subseteq E \cup \overline{E}$ we write $q(X)$ for the corresponding set of arcs of D'. Similarly, for a set $Y \subseteq A$ of arcs we write $q^{-1}(Y)$ for the corresponding set of undirected edges of G. Observe that for a feasible solution X to I, the arc set $q(X) \cup A_s \cup A_t$ is feasible for I', where A_s (resp., A_t) is the set of arcs leaving s (resp., entering t). Furthermore, a feasible solution Y to I' corresponds to a feasible solution $q^{-1}(Y \setminus (A_s \cup A_t))$ to I. Also note that, by the choice of c', we have that the cost of two corresponding solutions is the same. It follows that since $\ell = n/2$, any polynomial-time $f(\ell)$-factor approximation algorithm for Fault-Tolerant ℓ-Flow implies a polynomial-time $f(n/2)$-factor approximation algorithm for Weighted Robust Matching Augmentation, where $n = |U + W|$. ◀

We combine Lemma 14 with two hardness results from [16] and [15] to obtain the following approximation hardness result for FTF.

▶ **Theorem 15.** *FTF admits no polynomial-time* $\log^{2-\varepsilon}(\ell)$*-factor approximation algorithm for every* $\varepsilon > 0$*, unless* $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathsf{polylog}(n)})$.

Proof. We give a polynomial-time cost-preserving reduction from Directed Steiner Forest to FTF via Weighted Robust Matching Augmentation. The intermediate reduction step from Directed Steiner Forest to Weighted Robust Matching Augmentation is given in [16, Prop. 6.1]. Consider an instance I of Directed Steiner Forest on a weighted digraph $D = (V, A)$ on n vertices with k terminal pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. According to the reduction given in the proof of [16, Prop. 18], we obtain an instance of Weighted Robust Matching Augmentation on a graph of at most $2(n + k) + 2(n - k) = 4n =: n'$ vertices. By the arguments their proof, a $f(n')$-approximation algorithm for Weighted Robust Matching Augmentation yields a $f(4n)$-approximation algorithm for Directed Steiner Forest. We apply Proposition 14 to conclude that an $f(\ell)$-approximation algorithm for FTF yields a $f(2n)$-approximation algorithm for Directed Steiner Forest. According to the result of Halperin and Krauthgamer [15], the problem Directed Steiner Forest admits no polynomial-time $\log^{2-\varepsilon} n$-approximation algorithm for every $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathsf{polylog}(n)})$. We conclude that FTF admits no polynomial-time $\log^{2-\varepsilon}(\ell/2)$-factor approximation algorithm under the same assumption. ◀

3.2 Approximation Algorithms

We first present a simple polynomial-time $(\ell + 1)$-approximation algorithm for FTF, which is similar to the LP-based $(k+1)$-approximation for FTP. The algorithm computes a minimum-cost s-t flow of value $\ell + 1$ on the input graph with the following capacities: each vulnerable arc receives capacity 1 and any other arc capacity $1 + 1/\ell$. The solution then consists of all arcs in the support of the flow. To see that for this choice of capacities we obtain a feasible solution, recall that the value of any s-t cut upper-bounds the value of any s-t flow. Therefore, each s-t cut C has value at least $\ell + 1$, so C contains either at least ℓ safe arcs or at least $\ell + 1$ arcs. To prove the approximation guarantee, we show that any optimal solution to an FTF instance contains an s-t flow of value $\ell + 1$ and observe that we over-pay safe arcs by a factor of at most $(1 + 1/\ell)$.

▶ **Theorem 16.** *FTF admits a polynomial-time* $(\ell + 1)$*-approximation algorithm.*

Proof. Let I be an instance of FTF on a digraph $D = (V, A))$ with weight $c \in \mathbb{Z}^A_{\geq 0}$, terminals s and t, vulnerable arcs M and desired flow value ℓ. We consider an instance $I' = (D, c, s, t, \ell + 1, g)$ of MCF, where the arc capacities g are given by

$$g_e := \begin{cases} 1 & \text{if } e \in M, \text{ and} \\ 1 + \frac{1}{k} & \text{otherwise} \end{cases}$$

An optimal solution to I' can be computed computed in polynomial-time by standard techniques. We saw in the discussion at the beginning of Section 3.2 that the set of arcs of positive flow in a solution to I' yields a feasible solution to I.

It remains to bound the approximation ratio. Let Y^* be an optimal solution to I of cost $\mathrm{OPT}(I)$. We first show that Y^* contains $\ell + 1$ disjoint s-t paths.

▷ Claim 1. Y^* contains an s-t flow of value $\ell + 1$ with respect to the capacities g.

Proof. First observe that in any feasible solution to I, every s-t cut contains either at least ℓ safe arcs or at least $\ell + 1$ arcs. Now, an s-t cut Z in Y^* having at least ℓ safe arcs satisfies $g(Z) \geq (1 + \frac{1}{\ell}) \cdot \ell = \ell + 1$. On the other hand, an s-t cut Z' in Y^* containing at least $\ell + 1$ arcs satisfies $g(Z') \geq \ell + 1$. Hence, each s-t cut in Y^* has capacity at least $\ell + 1$. By the max-flow-min-cut theorem there is an s-t flow of value at least $\ell + 1$. ◁

The theorem now follows from the next claim.

▷ Claim 2. An optimal solution to I' has cost at most $(\ell + 1) \cdot \mathrm{OPT}(I)$.

Proof. Let $f^* \in \mathbb{Q}^A$ be an optimal s-t flow with respect to the capacities g. Furthermore, let Y be the set of arcs of positive flow, that is $Y := \{e \in A \mid f^*_e > 0\}$. Let $Y_M = Y \cap M$ be the vulnerable arcs in Y and let $Y_S = Y \setminus Y_M$ be the safe arcs. First, we may assume that each arc $e \in Y$ has flow value at least $f^*_e \geq 1/\ell$, since each arc has capacity either 1 or $1 + \frac{1}{\ell}$. This is true since we could scale the arc capacities g by a factor ℓ, which allows us to compute (in polynomial time) an integral optimal solution with respect to the scaled capacity function, using any augmenting paths algorithm for MCF. In addition, observe that we may pay a factor of at most $1 + \frac{1}{\ell}$ too much for each safe arc since the capacity of the safe arc is $1 + \frac{1}{\ell}$. Therefore, we may bound the cost of a safe arc $e \in Y_S$ by $\ell \cdot (1 + \frac{1}{\ell}) \cdot c_e \cdot f_e$ and the cost of each vulnerable arc $e \in Y_M$ by $\ell \cdot c_e \cdot f_e$, where f_e is the flow-value of arc e according to the solution Y. Hence, we obtain

$$\begin{aligned} c(Y) &= c(Y_S) + c(Y_M) \\ &\leq \ell \cdot \left((1 + \frac{1}{\ell}) \cdot \sum_{e \in Y_S} c_e \cdot f^*_e + \sum_{e \in Y_M} c_e \cdot f^*_e \right) \\ &\leq \ell \cdot (1 + \frac{1}{\ell}) \cdot \left(\sum_{e \in Y_S} c_e \cdot f^*_e + \sum_{e \in Y_M} c_e \cdot f^*_e \right) \\ &\leq (\ell + 1) \cdot \mathrm{OPT}(I) \ , \end{aligned}$$

where the first inequality follows from the two arguments above and the last inequality follows from Claim 1. ◁

◀

Note that we cannot simply use the dynamic programming approach as in the algorithm for 1-FTP to obtain an ℓ-approximation for FTF, since a solution to FTF in general does not have cut vertices, which are essential for the decomposition approach for the k-approximation for FTP.

A 2-approximation for Fault-Tolerant ℓ-Flow

We now show that for a fixed number ℓ of disjoint paths a much better approximation guarantee can be obtained. That is, we give a polynomial-time 2-approximation algorithm for Fault-Tolerant ℓ-Flow (FTℓF) (however, its running time is exponential in ℓ). The algorithm first computes a minimum-cost s-t flow of value ℓ and then augments it to a feasible solution by solving the following *augmentation problem*.

Fault-Tolerant ℓ-Flow Augmentation

Instance: arc-weighted directed graph $D = (V, A)$, nodes $s, t \in V$, arc-set $X_0 \subseteq A$ that contains ℓ disjoint s-t paths, and set $M \subseteq A$ of vulnerable arcs.

Task: Find minimum weight set $S \subseteq A \setminus X_0$ such that for every $f \in M$ the set $(X_0 \cup S) \setminus f$ contains ℓ disjoint s-t paths.

Our main technical contribution is that Fault-Tolerant ℓ-Flow Augmentation can be solved in polynomial time for fixed ℓ. Our algorithm is based on a dynamic programming approach and it involves solving many instances of the problem Directed Steiner Forest, which asks for a cheapest subgraph connecting ℓ given terminal pairs. This problem admits a polynomial-time algorithm for fixed ℓ [11], but is W[1]-hard when parameterized by ℓ, so the problem is unlikely to be fixed-parameter tractable [14]. Roughly speaking, we traverse the ℓ disjoint s-t paths computed previously in parallel, proceeding one arc at a time. In order to deal with vulnerable arcs, at each step, we solve an instance of Directed Steiner Forest connecting the ℓ current vertices (one on each path) to ℓ destinations on the same path by using backup paths. That is, we decompose a solution to the augmentation problem into instances of Directed Steiner Forest connected by safe arcs. An optimal decomposition yields an optimal solution to the instance of the augmentation problem. We find an optimal decomposition by dynamic programming. Essentially, we give a reduction to a shortest path problem in a graph that has exponential size in ℓ.

Let us fix an instance I of Fault-Tolerant ℓ-Flow Augmentation on a digraph $D = (V, A)$ with arc-weights $c \in \mathbb{Z}^A_{\geq 0}$ and terminals s and t. Let $P_1, P_2, \ldots, P_\ell$ be ℓ disjoint s-t paths contained in X_0. We assume without loss of generality that X_0 is the union of $P_1, P_2, \ldots, P_\ell$. If X_0 contains an arc e that is not on any of the ℓ paths, we remove e from X_0 and assign to it weight 0.

We now give the reduction to the shortest path problem. We construct a digraph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$; to distinguish it clearly from the graph D of I, we call the elements in V (A) of D *vertices* (*arcs*) and elements of $\mathcal{V}$ ($\mathcal{A}$) *nodes* (*links*). We order the vertices of each path P_i, $1 \leq i \leq \ell$, according to their distance to s on P_i. For two vertices x_i, y_i of P_i, we write $x_i \leq y_i$ if x_i is at least as close to s on P_i as y_i. Let us now construct the node set $\mathcal{V}$. We add a node v to $\mathcal{V}$ for every ℓ-tuple $v = (x_1, ..., x_\ell)$ of vertices in $V(X_0)$ satisfying $x_i \in P_i$, for every $i \in \{1, 2, \ldots, \ell\}$. Note that the corresponding vertices of a node are not necessarily distinct, since the ℓ *edge*-disjoint paths $P_1, P_2, \ldots, P_\ell$ may share vertices. We also define a (partial) ordering on the nodes in $\mathcal{V}$. From now on, let $v_1 = (x_1, ..., x_\ell)$ and $v_2 = (y_1, ..., y_\ell)$ be two nodes of $\mathcal{V}$. We write $v_1 \leq v_2$ if $x_i \leq y_i$ for every $1 \leq i \leq \ell$. Additionally, let $Q_i(x, y)$ be the sub-path of P_i from a vertex x of P_i to a vertex y of P_i.

We now construct the link set $\mathcal{A} := \mathcal{A}_1 \cup \mathcal{A}_2$ of $\mathcal{D}$ as the union of two link-sets $\mathcal{A}_1$ and $\mathcal{A}_2$, defined as follows. We add to $\mathcal{A}_1$ a link v_1v_2, if v_1 precedes v_2 and the subpaths of each P_i from x_i to y_i contain no vulnerable arc. That is, we let

$$\mathcal{A}_1 := \{v_1v_2 \mid v_1, v_2 \in \mathcal{V},\ v_1 \leq v_2,\ Q_i(x_i, y_i) \cap M = \emptyset \text{ for } 1 \leq i \leq \ell\} \ .$$

Algorithm 1 : Exact algorithm for Fault-Tolerant ℓ-Flow Augmentation.

Input: instance I of Fault-Tolerant ℓ-Flow Augmentation on a digraph $D = (V, A)$
1: Construct the graph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$
2: Find a shortest path $\mathcal{P}$ in $\mathcal{D}$ from $(s, \dots, s)$ to $(t, \dots, t)$
3: For each link $vw \in \mathcal{P} \cap \mathcal{A}_2$ add the arcs of an optimal solution to $I(v, w)$ to Y
4: **return** Y

We now define the link set $\mathcal{A}_2$. Let $v_1, v_2 \in \mathcal{V}$ such that v_1 precedes v_2. If there is some $1 \leq i \leq \ell$, such that $Q_i(x_i, y_i)$ contains at least one vulnerable arc, then we first need to solve an instance of Directed Steiner Forest on ℓ terminal pairs in order to compute the cost of the link v_1v_2. We construct an instance $I(v_1, v_2)$ of Directed Steiner Forest as follows. The terminal pairs are $(x_i, y_i)_{1 \leq i \leq \ell}$. The input graph is given by $D' = (V, A')$, where $A' = (A \setminus X_0) \cup \bigcup_{1 \leq i \leq \ell} \overleftarrow{Q_i}(x_i, y_i)$, where $\overleftarrow{Q_i}(x_i, y_i)$ are the arcs of $Q_i(x_i, y_i)$ in reversed direction. The arc costs are given by

$$c'_e := \begin{cases} c_e & \text{if } e \in A \setminus X_0, \text{ and} \\ 0 & \text{if } e \in \overleftarrow{Q_i}(x_i, y_i) \text{ for some } i \in \{1, 2, \dots, \ell\}. \end{cases}$$

That is, for $1 \leq i \leq \ell$, we reverse the path $Q_i(x_i, y_i)$ connecting x_i to y_i and make the corresponding arcs available at zero cost. We then need to connect x_i to y_i without using arcs in X_0. Since the number of terminal pairs is at most ℓ which is a constant, the Directed Steiner Forest instance $I(v_1, v_2)$ can be solved in polynomial time by the algorithm of Feldman and Ruhl given in [11]. Let $\mathrm{OPT}(I(v_1, v_2))$ be the cost of an optimal solution to $I(v_1, v_2)$. We add a link v_1v_2 to $\mathcal{A}_2$ if the computed solution of $I(v_1, v_2)$ is strongly connected. This completes the construction of $\mathcal{A}_2$. For a link $e \in \mathcal{A}$ we let the weight $w_e = 0$ if $e \in \mathcal{A}_1$ and $w_e = \mathrm{OPT}(I(v_1, v_2))$ if $e \in \mathcal{A}_2$.

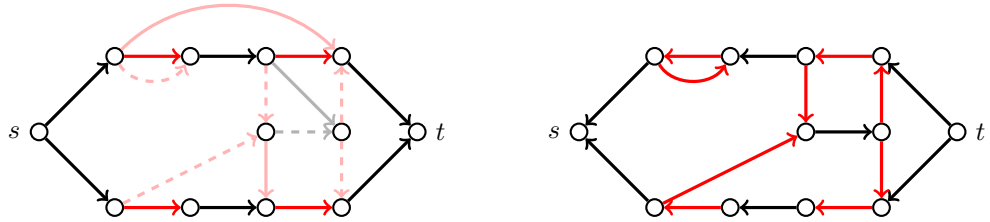
We now argue that a shortest path $\mathcal{P}$ from node $s_1 = (s, \dots, s) \in \mathcal{V}$ to node $t_1 = (t, \dots, t) \in \mathcal{V}$ in $\mathcal{D}$ corresponds to an optimal solution to I. For every link $vw \in \mathcal{P}$, we add the optimal solution to $I(v, w)$ computed by the Feldman-Ruhl algorithm to our solution Y. The algorithm runs in polynomial time for a fixed number ℓ of disjoint s-t paths, since it computes at most n^ℓ Min-cost Strongly Connected Subgraphs on ℓ terminal pairs, which can be done in polynomial time by [11]. Proving that the final algorithm is optimal is quite technical and requires another auxiliary graph and a technical lemma.

▶ **Theorem 17.** *The set Y computed by Algorithm 1 is an optimal solution to the instance I of Fault-Tolerant ℓ-Flow Augmentation. Furthermore, the running time is bounded by $O(|A||V|^{6\ell-2} + |V|^{6\ell-1} \log |V|)$.*

From Theorem 17 we obtain a polynomial-time 2-approximation algorithm for FTℓF: Let $\mathrm{OPT}(I)$ be the cost of an optimal solution to an instance I of FTℓF. The algorithm first computes a minimum-cost s-t flow X_0 and then runs the algorithm for the augmentation problem using X_0 as initial arc-set. The algorithm returns the union of the arc-sets computed in the two steps. By Theorem 17 we can augment X_0 in polynomial time to a feasible solution $X_0 \cup Y$ to I. Since we pay at most $\mathrm{OPT}(I)$ for the sets X_0 and Y, respectively, the total cost is at most $2\,\mathrm{OPT}(I)$.

▶ **Corollary 18.** *FTℓF admits a polynomial-time 2-approximation algorithm.*

The remainder of this section is devoted to sketching the proof of Theorem 17. For this purpose we need another auxiliary graph that we use as a certificate of feasibility. For a graph $H = (V, A^*)$ such that $X_0 \subseteq A^* \subseteq A$, we denote the corresponding residual graph



(a) Graph D and X_0 consisting of two disjoint paths. **(b)** Residual graph $D_{X_0}(X_0 \cup Y)$.

Figure 1 Illustration of the structure of feasible solutions to Fault-Tolerant ℓ-Flow Augmentation. Unsafe arcs are red, safe arcs are black. In Fig. 1a: edges of X_0 are black and red; edges of $A - X_0$ are light gray and light red. Dashed edges belong to Y.

by $D_{X_0}(A^*) = (V, A')$. The arc-set A' is given by $A' := \{uv \in A^* \mid uv \notin X_0\} \cup \{vu \in A^* \mid uv \in X_0\}$. An illustration of this graph is given in Figure 1. We first show that in a feasible solution $Y \subseteq A \setminus X_0$, each vulnerable arc in X_0 is contained in a strongly connected component of $D_{X_0}(X_0 \cup Y)$.

▶ **Lemma 19.** *Let $Y \subseteq A \setminus X_0$. Then Y is a feasible solution to I if and only if each vulnerable arc $f \in M \cap X_0$ is contained in a strongly connected component of $D_{X_0}(X_0 \cup Y)$.*

Proof. We first prove the "if" part, so let $f = uv$ be a vulnerable arc in X_0 that is contained in a strongly connected component of $D_{X_0}(X_0 \cup Y)$. Since $f \in X_0$, the arc f is reversed in $D_{X_0}(X_0 \cup Y)$ and since f is on a cycle C in $D_{X_0}(X_0 \cup Y)$, there is a path P from u to v in $D_{X_0}(X_0 \cup Y)$. Let P' be the path corresponding to P in $X_0 \cup Y$. Note that P' is not a directed path in D and that an arc e on P' is traversed forward if $e \in P' \cap Y$ and traversed backward if $e \in P' \cap X_0$. We partition P' into two disjoint parts $P'_{X_0} = P' \cap X_0$ and $P'_Y = P' \cap Y$. We now argue that $(X_0 - P'_{X_0} - f) \cup P'_Y$ contains ℓ disjoint s-t paths. Clearly, we have $(X_0 - P'_{X_0} - f) \cup P'_Y \subseteq X_0 \cup Y$. Furthermore, by our assumption that X_0 is the union of ℓ s-t edge-disjoint paths, for each vertex $v \in V - \{s, t\}$, we have $\delta^+(v) = \delta^-(v)$ and $\delta^+(s) = \delta^-(t) = \ell$. Since C is a cycle in $D_{X_0}(X_0 \cup Y)$ the degree constraints also hold for $(X_0 - P'_{X_0} - f) \cup P'_Y$. Hence $(X_0 - P'_{X_0} - f) \cup P'_Y$ is the union of ℓ disjoint s-t paths.

We now prove the "only if" part. Let $f = uv \in X_0$ be a vulnerable arc and suppose f is not contained in a strongly connected component of $D_{X_0}(X_0 \cup Y)$. Let $L \subseteq V$ be the set of vertices that are reachable from u in $D_{X_0}(X_0 \cup Y)$ and let $R = V - L$. Note that $s \in L$, since u is on some s-t path in X_0 and $t \in R$, since otherwise there is a path from u to v in $D_{X_0}(X_0 \cup Y)$ (since every arc in X_0 is reversed in $D_{X_0}(X_0 \cup Y)$). Let $L' = \{x_1, \ldots, x_\ell\} \subseteq L$, $x_i \in P_i$ for $1 \leq i \leq \ell$, be the vertices of L that are closest to t in X_0. We now claim that $\delta^+(L)$ is a cut of size ℓ in $X_0 \cup Y$ containing f. Since f is vulnerable this contradicts the feasibility of $X_0 \cup Y$. We have $f \in \delta^+(L)$ in $X_0 \cup Y$, since otherwise f is contained in a strongly connected component of $D_{X_0}(X_0 \cup Y)$. By the construction of L, we have $Y \cap \delta^+(L) = \emptyset$. Since X_0 is the union of ℓ disjoint paths, the set $\delta^+(L)$ has size at most ℓ, proving our claim, since this implies that $X_0 \cup Y$ is not feasible. ◀

Next, we sketch the proof of Theorem 17.

Proof of Theorem 17 (sketch). Let $\mathcal{P}$ be a shortest path in the $\mathcal{D}$ and let Y be the solution computed by Algorithm 1. Using Lemma 19 we can show the feasibility of Y.

▷ Claim 1. The solution Y computed by Algorithm 1 is feasible.

Let Y^* be an optimal solution to I of weight $\mathrm{OPT}(I)$. We now show that Y is optimal. Observe that the weight of Y is equal to $c'(\mathcal{P})$, so it suffices to show that $w(\mathcal{P}) \leq \mathrm{OPT}(I)$. To prove the inequality, we first introduce a partial ordering of the strongly connected components of $D_{X_0}(X_0 \cup Y^*)$. Using this ordering we can construct a path $\mathcal{P}'$ in $\mathcal{D}$ from $(s, \ldots, s)$ to $(t, \ldots, t)$ of cost $w(\mathcal{P}') = \mathrm{OPT}(I)$. We conclude by observing that a shortest path $\mathcal{P}$ has cost at most $w(\mathcal{P}')$.

▷ Claim 2. There is a path $\mathcal{P}'$ from $(s, \ldots, s)$ to $(t, \ldots, t)$ in $\mathcal{D}$ of cost at most $\mathrm{OPT}(I)$.

Proof of Claim 2 (sketch). We give an algorithm that constructs a path $\mathcal{P}'$ from $(s, \ldots, s)$ to $(t, \ldots, t)$ in $\mathcal{D}$ such that $\mathcal{P}'$ only uses links in $\mathcal{A}_2$ that correspond to strongly connected components of Y^* in $D_{X_0}(X_0 \cup Y^*)$. Starting from $s_1 = (s, \ldots, s) \in \mathcal{V}$, we perform the following two steps alternatingly until we reach $(t, ..., t) \in \mathcal{V}$.

1. From the current node u, we proceed by greedily taking links of $\mathcal{A}_1$ until we reach a node $v = (v_1, v_2, \ldots, v_\ell) \in \mathcal{V}$ with the property that each vertex v_i, $1 \leq i \leq \ell$, is either t or part of some strongly connected component of $D_{X_0}(X_0 \cup Y^*)$.
2. From the current node v, we take a link $vw \in \mathcal{A}_2$ to some node $w \in \mathcal{V}$, where the link vw corresponds to a strongly connected component Z of $D_{X_0}(X_0 \cup Y^*)$.

It can be shown that this algorithm indeed finds the desired solution. ◁

Finally, using the algorithm in [11] for finding a cost-minimal strongly connected subgraph on ℓ terminal pairs, we obtain that Algorithm 1 runs in time $O(mn^{6\ell-2} + n^{6\ell-1} \log n)$. ◀

3.3 Relation to a Problem of Open Complexity

In the previous section we obtained a 2-approximation for Fault-Tolerant ℓ-Flow. Ideally, one would like to complement this with a hardness (of approximation) result. However, since Fault-Tolerant ℓ-Flow Augmentation admits a polynomial-time algorithm according to Theorem 17, we cannot use the augmentation problem in order to prove NP-hardness of Fault-Tolerant ℓ-Flow; an approach that has been used successfully for instance for robust paths [3], robust matchings [16] and robust spanning trees [2]. Hence, there is some hope that Fault-Tolerant ℓ-Flow might actually be polynomial-time solvable. Here, we show that a polynomial-time algorithm for Fault Tolerant 2-Flow implies polynomial-time algorithms for 1-2-connected Directed 2 Steiner Tree, a problem whose complexity is open [9].

The problem 1-2-connected Directed 2 Steiner Tree is defined as follows. Given a digraph $D = (V, E)$ with costs $c \in \mathbb{Q}^A$, a root vertex $s \in V$, and two terminals $t_1, t_2 \in V$, the goal is to find a minimum cost set $X \subseteq E$ such that X contains two disjoint s-t_1 paths and one s-t_2 path.

▶ **Proposition 20.** *A polynomial-time algorithm for Fault Tolerant 2-Flow implies a polynomial-time algorithm for 1-2-connected Directed 2 Steiner Tree.*

Proof. Let I be an instance of 1-2-connected Directed 2 Steiner Tree on a graph $D = (V, A)$ with edge-weights $c \in \mathbb{Q}^E$, root $s \in V$, and terminals $T = \{t_1, t_2\}$. We construct an instance I' of Fault Tolerant 2-Flow as follows. We add to D two vertices u and t and four directed edges $\hat{A} = \{(s, u), (t_1, u), (u, t), (t_2, t)\}$. Let the resulting graph be D'. The edge weights c' of D' are given by

$$c'_e := \begin{cases} c_e & \text{if } e \in A(G), \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we let $M := A \cup \{(s, u), (t_1, u)\}$, that is, the edges incident to t are safe while all other edges are unsafe.

Let X be a feasible solution to I'. We have $\hat{A} \subseteq X$, since otherwise X is not feasible. We now show that there is at least one s-t_i path and there are at least two disjoint s-t_2 paths in $(V, X \setminus \hat{A})$. Assume first that there is no path from s to t_1 in $(V, X \setminus \hat{A})$. But then $\{(s,u),(v,t)\}$ is a cut of size two in D', where (s,u) is a vulnerable edge. This contradicts the feasibility of X. Now assume that there are no two disjoint s-t_2 paths in $(V, X \setminus \hat{A})$. It is not hard to see that then we have a contradiction to the feasibility of X. Finally, observe that there is a one-to-one correspondence between feasible solutions to I and I'. ◀

4 Conclusions and Future Work

We introduced the two problems FTP and FTF, which add a non-uniform fault model to the classical edge-disjoint s-t paths problem. This fault-model leads to a dramatic increase in the computational complexity. We gave polynomial-time algorithms for several classes of instances including the case $k = 1$ and DAGs with fixed k. Furthermore, we proved a tight bound on the integrality gap of a natural LP relaxation for FTP and obtained a polynomial k-approximation algorithm. For FTF, our main result is a 2-approximation algorithm for fixed ℓ. One of the main open problems is to see whether the approximation guarantee for FTP can be improved to the approximation guarantees of the best known algorithms for the Steiner Tree problem. Furthermore, it would be interesting to see if the methods employed in the current paper for 1-FTP and k-FTP on directed acyclic graphs can be extended to k-FTP on general graphs. Another intriguing open question is whether Fault-Tolerant ℓ-Flow is NP-hard, which is not known even for $\ell = 2$. We showed that a positive result in this direction implies polynomial-time algorithms for two Steiner problems whose complexity status is open.

References

1. David Adjiashvili. Non-uniform robust network design in planar graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2015.
2. David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 13–26. Springer, 2020.
3. David Adjiashvili, Sebastian Stiller, and Rico Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1-2):361–390, 2015. doi:10.1007/s10107-014-0760-6.
4. Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In *European Symposium on Algorithms*, pages 862–873. Springer, 2005.
5. Sylvia Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. A 2-approximation algorithm for flexible graph connectivity. *arXiv preprint*, 2021. arXiv:2102.03304.
6. Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.
7. Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. *Algorithmica*, 72(2):493–514, 2015.
8. Moses Charikar, Chandra Chekuri, To-yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.
9. Joseph Cheriyan, Bundit Laekhanukit, Guyslain Naves, and Adrian Vetta. Approximating rooted steiner networks. *ACM Transactions on Algorithms (TALG)*, 11(2):1–22, 2014.
10. Kedar Dhamdhere, Vineet Goyal, R Ravi, and Mohit Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 367–376. IEEE, 2005.

11 Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM Journal on Computing*, 36(2):543–561, 2006. doi:10.1137/S0097539704441241.

12 Harold N Gabow and Suzanne R Gallagher. Iterated rounding algorithms for the smallest k-edge connected spanning subgraph. *SIAM Journal on Computing*, 41(1):61–103, 2012.

13 Daniel Golovin, Vineet Goyal, Valentin Polishchuk, R Ravi, and Mikko Sysikaski. Improved approximations for two-stage min-cut and shortest path problems under uncertainty. *Mathematical Programming*, 149(1-2):167–194, 2015.

14 Jiong Guo, Rolf Niedermeier, and Ondřej Suchỳ. Parameterized complexity of arc-weighted directed Steiner problems. *SIAM Journal on Discrete Mathematics*, 25(2):583–599, 2011.

15 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 585–594, 2003. doi:10.1145/780542.780628.

16 Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. How to secure matchings against edge failures. *SIAM Journal on Discrete Mathematics*, 35(3):2265–2292, 2021.

17 Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001. doi:10.1007/s004930170004.

18 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

19 Gang Yu and Jian Yang. On the robust shortest path problem. *Computers and Operations Research*, 25(6):457–468, June 1998.

20 Paweł Zieliński. The computational complexity of the relative robust shortest path problem with interval data. *European Journal of Operational Research*, 158(3):570–576, November 2004. doi:10.1016/s0377-2217(03)00373-4.