Maximum-Flow and Minimum-Cut Sensitivity Oracles for Directed Graphs

Mridul Ahi
Department of Mathematics, IIT Delhi, India

Keerti Choudhary
Department of Computer Science and Engineering, IIT Delhi, India

Shlok Pande
Department of Computer Science and Engineering, IIT Delhi, India

Pushpraj
Department of Computer Science and Engineering, IIT Delhi, India

Lakshay Saggi
Department of Computer Science and Engineering, IIT Delhi, India

Abstract

This paper addresses the problem of designing fault-tolerant data structures for the (s,t)-max-flow and (s,t)-min-cut problems in unweighted directed graphs. Given a directed graph $G=(V,E)$ with a designated source s, sink t, and an (s,t)-max-flow of value λ, we present constructions for max-flow and min-cut sensitivity oracles, and introduce the concept of a fault-tolerant flow family, which may be of independent interest. Our main contributions are as follows.

1. *Fault-Tolerant Flow Family:* We construct a family $\mathcal{B}$ of $2\lambda+1$ (s,t)-flows such that for every edge e, $\mathcal{B}$ contains an (s,t)-max-flow of $G-e$. This covering property is tight up to constants for single failures and provably cannot extend to comparably small families for $k\geq 2$, where we show an $\Omega(n)$ lower bound on the family size, independent of λ.
2. *Max-Flow Sensitivity Oracle:* Using the fault-tolerant flow family, we construct a single as well as dual-edge sensitivity oracle for (s,t)-max-flow that requires only $O(\lambda n)$ space. Given any set F of up to two failing edges, the oracle reports the updated max-flow value in $G-F$ in $O(n)$ time. Additionally, for the single-failure case, the oracle can determine in constant time whether the flow through an edge x changes when another edge e fails.
3. *Min-Cut Sensitivity Oracle for Dual Failures:* Recently, Baswana et al. (ICALP'22) designed an $O(n^2)$-sized oracle for answering (s,t)-min-cut size queries under dual edge failures in constant time, along with a matching lower bound. We extend this by focusing on graphs with small min-cut values λ, and present a more compact oracle of size $O(\lambda n)$ that answers such min-cut size queries in constant time and reports the corresponding (s,t)-min-cut partition in $O(n)$ time. We also show that the space complexity of our oracle is asymptotically optimal in this setting.
4. *Min-Cut Sensitivity Oracle for Multiple Failures:* We extend our results to the general case of k edge failures. For any graph with (s,t)-min-cut of size λ, we construct a k-fault-tolerant min-cut oracle with space complexity $O_{\lambda,k}(n\log n)$ that answers min-cut size queries in $O_{\lambda,k}(\log n)$ time. This also leads to improved fault-tolerant (s,t)-reachability oracles, achieving $O(n\log n)$ space and $O(\log n)$ query time for up to $k=O(1)$ edge failures.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Network flows

Keywords and phrases Fault tolerance, Data structures, Minimum cuts, Maximum flows

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.5

Related Version *Full Version*: https://arxiv.org/abs/2512.00153

Funding *Keerti Choudhary*: Partially supported by Google India Research Awards.
Lakshay Saggi: Supported by Prime Minister's Research Fellowship (PMRF).

Acknowledgements The authors would like to thank Thatchaphol Saranurak, Surender Baswana, and Koustav Bhanja for helpful discussions.

1 Introduction

The increasing scale and complexity of modern networks necessitate the development of efficient algorithms for solving fundamental graph problems. Among these problems, the *max-flow* and *min-cut* problems have received significant attention due to their broad applicability in transportation networks, data propagation, resource allocation, and numerous other domains. Since the seminal work of Ford and Fulkerson [31], extensive research has focused on computing max-flow in networks [8, 12, 23, 30, 32, 35, 36, 42, 44, 54, 55], culminating in the near-linear-time algorithm by Chen et al. [16]. While the problem of computing max-flow and min-cut in a graph has been extensively studied in the past 70 years for static graphs, little is known about designing efficient data-structures for these problems for networks prone to failures.

In recent years, there has been significant interest in fault-tolerant data structures, motivated by the fact that real-world networks are often vulnerable to node or link failures. In the fault-tolerant model, we assume that the total number of failures at any point remains bounded by a parameter k, which depends on the robustness of the network. Typically, k is much smaller than the total number of vertices in the network. Over the past two decades, many efficient fault tolerant data-structures have been developed for various classical graph problems, including spanners [21, 46, 47], distance preservers [11, 38, 48], distance oracles [9, 14, 18, 26], reachability [5, 13, 17], connectivity [6, 27, 28, 33, 49], etc.

Despite these advances, the domain of max-flow and min-cut problems has seen limited progress in the context of network failures. This paper aims to address this gap by studying the problem of flows and cuts in directed networks susceptible to edge failures. We introduce simple yet powerful techniques for the aforementioned problems, which lead to state-of-the-art results for the specific problems which we study. Our objective is to compute a compact data structure, referred to as a *sensitivity* or *fault-tolerant* oracle, which, after the failure of any set of edges in the input graph G efficiently determines the max-flow and min-cut with respect to a source s and sink t. This problem is formalized as follows.

▶ **Question 1.** *Given a directed, unweighted graph G with a designated source s and a designated sink t, design a data-structure that given any query set of edges $F \subseteq E$ of size k, outputs the (s,t)-max-flow and (s,t)-min-cut on deletion of edges in F from graph G.*

1.1 Our Contributions

Max-flow Sensitivity Oracle for Single and Dual failures

The first result we discuss pertains to the max-flow problem in directed graphs. To date, no efficient sensitivity oracles for max-flow are known, even for handling a single edge failure. While it is possible to determine efficiently whether the (s,t)-max-flow value decreases after an edge failure – thanks to the work of Picard and Queyranne [50], which yields an $O(n)$-sized data structure with $O(1)$ query time (see Theorem 4.6 of [4]) – the problem of efficiently reporting the updated (s,t)-max-flow after an edge failure remains unresolved. The complexity of this problem arises from the potential need to reroute flow through an alternative path in the residual graph, especially if the failed edge was carrying flow in the original (s,t)-max-flow of G.

To tackle this, we first demonstrate that for any directed graph G, there exists a small family of flows that effectively captures the maximum flows of the graph $G-e$ for all potential edge failures $e \in E$. Specifically, we obtain the following result.

▶ **Theorem 1.1.** *For any directed graph $G = (V, E)$ with an (s, t)-max-flow of value λ, there exists a family $\mathcal{B}$ of $2\lambda + 1$ (s, t)-flows in G satisfying the following property:*

For each edge e, the family $\mathcal{B}$ includes an (s, t)-max-flow for the graph $G - e$.

We further prove an $\Omega(n)$ lower bound on the cardinality of $\mathcal{B}$, independent of λ, for $k \geqslant 2$ failures.

It is important to note that the above theorem provides a compact flow family resilient to a single edge failure, which can be amenable to various algorithmic applications. However, it does not directly imply a compact data structure for efficiently reporting the maximum flow in a fault-prone network. We thus ask the following question.

▶ **Question 2.** *Is it possible to design a compact data structure that, upon the failure of any set $F \subseteq E$, possibly of size one, can efficiently update the (s, t)-max-flow of the graph?*

To the best of our knowledge, this problem has not been addressed in the existing literature. We answer this question affirmatively by presenting the following sensitivity oracle for handling both single and dual edge failures, which leverages the flow-covering result from Theorem 1.1.

▶ **Theorem 1.2.** *For any directed graph $G = (V, E)$ with n vertices, there exists an (s, t)-max-flow f and a data structure of size $O(\lambda n)$, where λ is the flow value, such that upon the failure of any set $F \subseteq E$ of up to two edges, the data structure can implicitly report the updated (s, t)-max-flow f' for $G - F$ in $O(n)$ time, that is, it can identify all edges e in $G - F$ for which $f'(e) \neq f(e)$ in $O(n)$ time.*

Moreover, for the case of single edge failure ($|F| = 1$), the data structure can determine for any edge e in $G - F$ whether $f'(e)$ is 0 or 1 in $O(1)$ time.

We note that the bound in Theorem 1.2 is essentially tight, as $\Omega(\lambda n)$ is also a lower bound on the number of edges required to represent a maximum flow from source to sink.

Min-Cut Sensitivity Oracle for Dual failures

We next discuss our result on min-cut sensitivity oracle for two edge failures. Baswana et al. [4] presented an $O(n^2)$-sized data structure to answer min-cut queries after two edge failures in $O(1)$ time. They also gave a conditional lower bound of $\widetilde{\Omega}(n^2)$ on the oracle size, based on the *Directed Reachability Hypothesis*. Recently, Bhanja [10] improved this lower bound by proving an unconditional bound of $\Omega(n^2)$ on the size of the oracle. This lower bound holds for directed as well as undirected graphs.

Given that the result of [4] has a matching upper and lower bound of $\Theta(n^2)$, any improvement to the dual edge failure oracle seems unlikely. However, in many applications, the focus is towards networks with cuts of small size. Similar scenarios have been previously studied by Abboud et al. [1], Akmal and Jin [2] for finding all pairs of vertices in graphs with small edge connectivity. This naturally raises the following question.[1]

[1] We remark that the data-structure of [4] takes quadratic size even when $\lambda = 1$. For details, see the full version.

▶ **Question 3.** *Given a directed graph G with an (s,t)-min-cut of size $O(n^{1-\epsilon})$ for some $\epsilon > 0$, can we construct a sub-quadratic sized oracle for efficiently reporting the (s,t)-min-cut size after dual edge failures?*

We answer this question in the affirmative by presenting the following result, where the size of the oracle scales linearly with the min-cut size λ.

▶ **Theorem 1.3.** *For any directed graph G with n vertices and an (s,t)-min-cut of size λ, there exists a dual fault-tolerant min-cut oracle of size $O(\lambda n)$ that, given any set F of two edge failures, reports the size of the (s,t)-min-cut in the graph $G - F$ in $O(1)$ time. Furthermore, the oracle reports an (s,t)-min-cut partition in $G - F$ in $O(n)$ time.*

Our min-cut oracle for dual failures employs a different set of tools compared to those in [4]. In particular, we show the robustness of our flow-covering result of Theorem 1.1 by combining it with the seemingly unrelated 1-fault-tolerant strong-connectivity oracle of Georgiadis et al. [34], that takes linear space and answers strong-connectivity queries after single edge failure in constant time. This combination not only allows us to design an alternate min-cut sensitivity oracle but also introduces a new set of ideas which are of independent interest.

We further show that the bound of λn on the size of the oracle in Theorem 1.3 is tight by presenting the following result.

▶ **Theorem 1.4.** *For any positive integers n, λ with $n \geqslant \lambda$, there exists a directed graph G on n vertices, with source s, sink t, and an (s,t)-min-cut of size λ, such that any dual fault-tolerant min-cut oracle for G requires $\Omega(\lambda n)$ space.*

Fault-Tolerant Min-Cut Oracle for general k failures

We now discuss the problem of designing fault-tolerant min-cut oracle for graphs under the general setting of up to k failures. Our goal is to construct an oracle of $F(k,\lambda) \cdot o(n^2)$ space and $F'(k,\lambda)$ query time. Before presenting our results for this scenario, it is essential to set the context by discussing a broader landscape.

The task of building a k-fault-tolerant (s,t)-min-cut oracle is notably more complex than designing a k-fault-tolerant (s,t)-reachability oracle[2]. At present, optimal-size (s,t)-reachability oracles in the fault-tolerant setting have been achieved only for single and dual failures. See Table 1 for a brief summary of existing results. For $k = 1$, the dominator trees introduced by Lengauer and Tarjan [45] provides a linear-size data structure with constant query time. Choudhary [17] extended this by presenting an optimal dual fault-tolerant single source reachability oracle. For general k, the situation becomes significantly more challenging. The only currently known results are an oracle with $O(2^k n)$ query time due to fault-tolerant reachability preservers of [5]; and an all-pairs reachability oracle by Brand and Saranurak [56], which achieves a size of $O(n^2 \log n)$ and has $O(k^\omega)$ query time.

This naturally leads us to the following open question.

▶ **Question 4.** *Does there exist a linear-size oracle with constant query time for the (s,t)-reachability problem under $k \geqslant 3$ failures?*

[2] This is because (s,t)-reachability can be reduced to (s,t)-min-cut by adding a dummy source with an out-edge to the original source, limiting the maximum flow to one.

Table 1 A summary of existing fault-tolerant reachability oracles.

Problem	Failures	Size of the oracle	Query time	Reference
s-Reachability	1	$O(n)$	$O(1)$	Lengaur & Tarjan [45]
s-Reachability	2	$O(n)$	$O(1)$	Choudhary [17]
All-Pairs Reachability	k	$O(n^2 \log n)$	$O(k^\omega)$	Brand & Saranurak [56]
s-Reachability	k	$O(2^k n)$	$O(2^k n)$	Baswana et al. [5]

In this work, we affirmatively address this question by providing near-optimal results for (s,t)-reachability under any constant number of failures. In fact, we solve the more general problem of the k-fault-tolerant min-cut, as formalized in the following theorem.

▶ **Theorem 1.5.** *For any n vertex directed graph G with an (s,t)-min-cut of size λ, there exists a k-fault-tolerant minimum cut oracle requiring $O(2^{O(L^4 \log L)} n \log n)$ space, where $L = \lambda + k$. This oracle can compute the size of the (s,t)-min-cut in the graph $G - F$, for any set F of k edges, in $O(2^{O(L^4 \log L)} \log n)$ time. Furthermore, the oracle can report an (s,t)-min-cut in $G - F$ in an additional computation time of $O(kn)$.*

Although the space complexity and query time of our oracle grow significantly with respect to k, the oracle becomes highly efficient for small values of k. Similar assumptions have been made in prior work. For instance, recent advancements [29, 41, 56, 57] in fault-tolerant distance oracles for undirected graphs have resulted in a breakthrough by Dey and Gupta [19] where they design an oracle with $O(k^4 n^2)$ space and query time of $(k \log n)^{O(k^2)}$.

A summary of existing results on (s,t)-min-cut sensitivity oracles, along with our new contributions, is presented in Table 2.

Table 2 A summary of (s,t)-min-cut sensitivity oracles for directed graphs.

Failures	Size of the oracle	Query time	Reference
1	$O(n)$	$O(1)$	Picard et al. [50]
2	$O(n^2)$	$O(1)$	Baswana et al. [4]
2	$O(\lambda n)$	$O(1)$	Theorem 1.3
general k	$O(2^{O((\lambda+k)^4 \log(\lambda+k))} n \log n)$	$O(2^{O((\lambda+k)^4 \log(\lambda+k))} \log n)$	Theorem 1.5

As a direct consequence of Theorem 1.5, we obtain the following corollary.

▶ **Corollary 1.6.** *For any directed graph $G = (V, E)$ with n vertices, a source s, a sink t, and any constant integer $k \geqslant 1$, there exists an (s,t)-reachability oracle that takes $O(n \log n)$ space and determines in $O(\log n)$ time whether a path exists from s to t in the graph $G - F$ for any query set $F \subseteq E$ of at most k edges.*

1.2 Other Related Works

This subsection provides an overview of additional related works. For directed weighted graphs, Baswana and Bhanja [3] proposed a novel min-cut oracle that efficiently handles single edge failure. Their oracle takes $O(n)$ space and can return the size of the (s,t)-min-cut after failure of any edge in constant time. Additionally, they provide an $O(n^2)$ space data structure for finding the resulting cut in $O(n)$ time. This result, as well as the earlier work of Baswana and Bhanja [3] for dual failures in the unweighted setting, holds for both directed as well as undirected graphs.

We now turn to discuss the existing results specifically for the setting of undirected graphs. Baswana and Pandey [7] designed the first all-pairs single-edge sensitivity oracle for undirected unweighted graphs that takes $O(n^2)$ space and answers any query in O(1) time. Currently, the problem of extending this result to dual failures or weighted graphs remains open.

Additionally, single fault-tolerant oracles for Steiner and global min-cuts exist as byproducts of the cactus representation [20], which stores all global min-cuts, and the connectivity carcass [22, 24], which stores Steiner min-cuts, as noted in [7, 10]. Recently, Bhanja [10] gave efficient sensitivity oracles for Steiner and global min-cuts under single edge failure in undirected weighted graphs.

A closely related problem to fault-tolerant min-cut is dynamic min-cut, where the goal is to maintain min-cut information as the graph undergoes edge insertions or deletions. For global min-cuts, early work such as Thorup's tree-packing technique [51] laid the foundation, while recent advances using expander-decomposition-based methods [37, 40] have significantly improved the query time. For all-pairs min-cuts, Jin and Sun [39] achieve state-of-the-art results with a subpolynomial update time of $n^{o(1)}$ for graphs with small min-cut size. Further recent developments for flows in incremental and decremental settings can be found in [15, 52, 53].

1.3 Organization of the Paper

We introduce relevant notations, definitions, and some key properties in Section 2. In Section 3, we provide an overview of our techniques. Our results for max-flow sensitivity oracles are presented in Section 5. In Section 6, we review important structural properties of min-cuts that play a crucial role in computing min-cut sensitivity oracles. The results for dual fault-tolerant and k-fault-tolerant min-cut sensitivity oracles are presented in Section 7 and Section 8, respectively. We conclude with a discussion of open problems in Section 9. Proofs omitted from the main text and our lower bound results are discussed in the full version.

2 Preliminaries

Let $G = (V, E)$ be an unweighted directed multi-graph with n vertices and m edges. Hereon, we use the terms *graph* and *multi-graph* interchangeably. Let $s \in V$ be a source vertex, and $t \in V$ be a sink vertex. We use the notation $\textsc{max-flow}(s, t, G)$, or equivalently $\textsc{min-cut}(s, t, G)$, to denote the value of the maximum-flow or minimum-cut in G with respect to source s and sink t. For any edge $e \in E$, we use $\overleftarrow{e}$ to denote the edge obtained by reversing the direction of e. For any vertex $x \in V$, we denote its set of out-neighbors by $\text{OUT}(x)$ and its set of in-neighbors by $\text{IN}(x)$. For any subset of vertices $A \subseteq V$, the subgraph of G induced by vertices in A is denoted by $G[A]$. For any set F of edges in G, the graph obtained by deleting the edges in F from G is denoted by $G - F$.

A set C of edges in G is called an (s, t)-cut if there is no s to t path in the graph $G - C$. Such a cut C is called a *min-cut* if it minimizes the number of edges in the cut, i.e., it is the smallest set of edges whose removal disconnects t from s. Any (s, t)-min-cut partitions the vertex set into two disjoint subsets: one containing the source vertex, denoted as A_C, and the other containing the sink vertex, denoted as B_C. Thus, an (s, t)-min-cut can equivalently be represented by this partition (A_C, B_C).

Next we define the notion of nearest-cut, farthest-cut, $(\min + k)$-cut, and critical edges.

▶ **Definition 2.1.** *An (s,t)-min-cut C^* is said to be the Nearest Min-Cut (Farthest Min-Cut) if for each (s,t)-min-cut C we have $A_{C^*} \subseteq A_C$ ($A_{C^*} \supseteq A_C$). We denote such a C^* with $NMC(s,t)$ ($FMC(s,t)$).*

▶ **Definition 2.2** (($Min+k$)-cut)**.** *A set $\mathcal{E}$ of edges is said to be a $(\min+k)$ (s,t)-cut if $\mathcal{E}$ is an (s,t)-cut of size k more than the size of minimum (s,t)-cut in G, and $\mathcal{E}$ is a **minimal** (s,t)-cut (i.e., no proper subset of $\mathcal{E}$ is an (s,t)-cut).*

▶ **Definition 2.3** (Critical edge)**.** *An edge e in G is said to be* critical *if it is contained in some (s,t)-min-cut of G. In other words, e is a critical edge if it carries a unit flow with respect to every (s,t)-max-flow of G. Any edge which is not a critical edge is a non-critical edge.*

For any flow f from source s to sink t in a graph G, we use notation $\text{NULL}_G(f)$ to represent the collection of those edges in G that carry zero flow under f. Further, $\text{NULL}_G(f, \min+1)$ represents the collection of those edges in $\text{NULL}_G(f)$ that are contained in some $(\min+1)$ (s,t)-cut in G.

The following lemma will be used crucially in our max-flow and min-cut sensitivity oracles.

▶ **Lemma 2.4.** *For any integer (s,t)-max-flow f in an unweighted graph G, the cardinality of the set $\text{NULL}_G(f, \min+1)$ is at most $2n$.*

Below we state an assumption used in our max-flow and min-cut sensitivity oracles.

▶ **Assumption 2.5.** *In our sensitivity oracles we assume that each vertex of the graph G lies on some simple (s,t)-path. This assumption is justified as only edges incident to such vertices can contribute to an (s,t)-min-cut, even in the presence of edge failures.*

3 Technical Overview

In this section, we provide a detailed exposition of the core technical ideas of our paper.

3.1 Fault-tolerant Flow Family

Let us begin by revisiting the (s,t)-reachability problem under single failure. It is well-known that for any two vertices s and t in a graph G such that t is reachable from s, there exist two paths P and Q intersecting *only* at (s,t)-cuts of size one. See Figure 1. Consequently, if an edge e fails but does not disrupt (s,t)-reachability, at least one of these paths, P or Q, serves as a certificate of (s,t)-reachability.

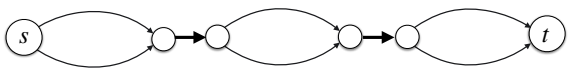


Figure 1 A pair of two (s,t) paths intersecting at (s,t)-cuts of size one.

One of the main contributions of our work is extending this result to (s,t)-min-cuts of size λ (greater than 1). Specifically, we construct a family $\mathcal{A}$ of $\lambda+1$ max-flows such that for any non-critical edge e (i.e. e satisfies the condition that failure of e *does not* reduce the min-cut size), there exists a flow $f \in \mathcal{A}$ satisfying $f(e) = 0$. This flow serves as a max-flow certificate for the graph $G - e$.

The construction for $\mathcal{A}$ proceeds through a sophisticated flow decomposition technique. First, an auxiliary graph $H = (V, E, c)$ is constructed where critical edges are assigned capacity $\lambda + 1$ while non-critical edges are assigned capacity λ. We argue that the (s, t)-max-flow f in the capacitated graph H has value $\lambda(\lambda + 1)$. To compute the family $\mathcal{A} = \{f_1, f_2, \ldots, f_{\lambda+1}\}$, we iteratively extract valid (s, t)-flows from f in a total of $\lambda + 1$ iterations.

These flows satisfy the property that for each edge e in H,

$$f(e) = f_1(e) + f_2(e) + \cdots + f_{\lambda+1}(e).$$

To ensure that each iteration produces a valid maximum flow for G, we employ a careful application of circulation theory with lower bounds, where we put appropriate upper and lower bounds on the flow that can be removed from an edge during any iteration. We prove that these bounds guarantee that the extracted flow in each iteration is indeed a valid (s, t)-max-flow in G.

Since $f = f_1 + f_2 + \cdots + f_{\lambda+1}$ and the capacity of non-critical edges in G is set to λ, it follows that for each non-critical edge e there is at least one flow $f_i \in \mathcal{A}$ that satisfies $f_i(e) = 0$.

3.2 Max-flow Sensitivity Oracle

Building upon the fault-tolerant flow family, we develop compact sensitivity oracles for max-flow problem under both single and dual edge failures.

Single Edge Failure

Observe that a straightforward way to store edges carrying non-zero flow for each $f_i \in \mathcal{A}$ would require $O(\lambda^2 n)$ space, since each flow can saturate up to λn edges. However, we show that it is possible to store all flows implicitly in $O(\lambda n)$ space by carefully constructing $\mathcal{A}$ such that the number of edges on which any two flows differ is linear in n. To handle the failure of critical edges, the family $\mathcal{A}$ is further extended to compute a compact family $\mathcal{B}$ within the same size bounds, such that $\mathcal{B}$ contains a max-flow of $G - e$ for each edge e, whether critical or non-critical.

Our oracle supports two types of queries:

- Flow through specific edge: Given a failing edge e and a query edge x, determine $f_e(x)$ in $O(1)$ time, where f_e is a canonical max-flow of $G - e$.
- Flow through all edges: Given a failing edge e, the oracle implicitly reports the updated max-flow in $O(n)$ time.

Dual Edge Failure

A natural extension of Theorem 1.1 is to ask whether the flow family can be generalized to handle multiple edge failures, thereby yielding max-flow sensitivity oracles for dual or higher failures. Unfortunately, this is not the case, as we prove an $\Omega(n)$ lower bound on the cardinality of $\mathcal{B}$ for $k \geqslant 2$ failures (for details see the full version).

However, we overcome this obstacle by exploiting the following key property: for any (s, t)-flow f in G with $f(e) = 0$, the graph $G_f - e$ is the residual network of $(G - e)$ under f. Although seemingly straightforward, this observation provides a bridge between fault-tolerant flow families and our max-flow (as well as min-cut) sensitivity oracles under dual edge failures.

For instance, when edges e, e' are deleted, the updated max-flow can be efficiently obtained if there exists a maximum flow f satisfying $f(e) = 0$. In this case, re-routing reduces to adjusting the flow through e' using graph $G_f - e$, rather than recomputing $(G - e)_f$. The

existence of such flows are guaranteed by the family $\mathcal{B}$. Moreover, we show that by sparsifying G_f using the 1-fault-tolerant strong connectivity preservers of Georgiadis et al. [34], this re-routing can be performed in $O(n)$ time.

3.3 Reporting Min-Cut after Dual Failures

We next describe the construction of an oracle to determine the min-cut size after two edge failures.

Existing Approach. As observed in [4], a challenging scenario arises when both failing edges are non-critical but their deletion reduces the min-cut size by one (i.e., the failing edges belong to a minimal cut of size $\lambda + 1$). To tackle this, [4] explicitly stored a family of $(\lambda + 1)$-sized minimal cuts in appropriate auxiliary graphs, in a total of $O(n^2)$ space.

Our Idea. We take a detour from existing ideas and instead exploit the following insight based on max-flows: If an edge e is non-critical, then there exists a flow f such that $f(e) = 0$, making f an (s, t)-max-flow in $G - e$. Then, upon deletion of another edge e', the max-flow value decreases by one if and only if:

1. $f(e') = 1$, and
2. Flow through e' cannot be rerouted via an alternate path,i.e., there is no cycle containing $(e')^{rev}$ in the residual graph $G_f - e$. See Figure 2.

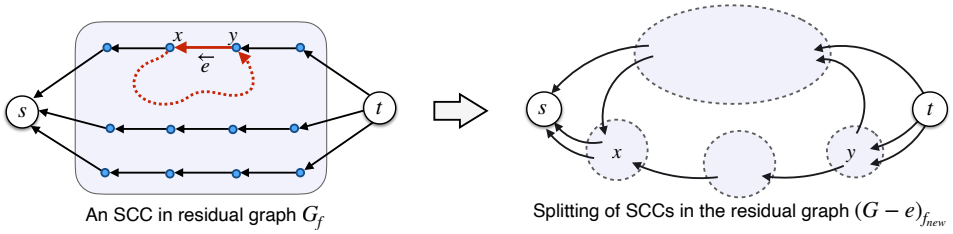


Figure 2 The SCCs in the residual graph G_f may split on deletion of e, which makes some intra-SCC edges as inter-SCC in the updated residual graph.

To exploit this observation, the first challenge is to bound the number of (s, t)-max-flows one will need. For this, the family $\mathcal{A}$ comes to our rescue and bounds the number of such flows by $\lambda + 1$. Another challenge is constructing an efficient and compact oracle to check if the above two conditions hold. We show that the first condition can be verified using our data structure over $\mathcal{A}$, while the second condition can be checked using a 1-fault-tolerant SCC data structure from [34] applied to the residual graph G_f. Both structures use linear space per flow in $\mathcal{A}$. This leads to an efficient $O(\lambda n)$-space data structure for min-cut sensitivity oracle for dual failures. Our oracle to report the cut partitioning is more challenging and is deferred to the full version.

3.4 Handling More than Two Edge Failures

We now discuss our ideas for handling general k failures. Given any set F of failing edges of size at most k, we first consider the following simpler problem: Does the size of the (s, t)-min-cut decrease by exactly k when the edges in F fail? This problem turns out to be relatively easier because it requires checking if all k edges simultaneously contribute to some (s, t)-min-cut in G. We design a data structure $\mathcal{O}_{\text{MINCUT}}(s, t, G)$ of size $O(\lambda n)$ that verifies this in $O(k^2)$ time. Additionally, it reports a min-cut in the graph $G - F$ in $O(kn)$ time.

Even though the simpler subproblem is easier to handle, the general problem of finding an (s,t)-min-cut in $G-F$ is significantly more challenging.

For any failing set F containing up to k edges, it can be observed that:

$$|\textsc{min-cut}(s,t,G-F)| = \min_{0\leqslant i<|F|} \min_{C\in(\lambda+i)\text{-cut}} (\lambda+i-|C\cap F|).$$

This suggests that studying minimal cuts of size up to $\lambda+k$ is crucial for answering min-cut queries. But these cuts can interact with each other in a number of ways, making it difficult to structurally analyse them.

To address the complexity of minimal cuts of size up to $\lambda+k$, we leverage a recent result by Kim et al. [43], which provides a powerful graph augmentation technique. Specifically, their result shows that one can augment a graph with additional random edges $\mathcal{E}\subseteq V\times V$ so that a given minimal cut C of size at most $\lambda+k$ becomes a valid minimum-cut in the augmented graph with probability $1/g(\lambda+k)$, for some computable function g. Intuitively, this augmentation allows us to transform the problem from studying minimal cuts to just minimum cuts. This augmentation, along with the data-structure $\mathcal{O}_{MINCUT}(s,t,G+\mathcal{E})$ computed for each sampled set $\mathcal{E}$, helps us handle multiple failures effectively and obtain the desired oracle.

4 A family of Fault-tolerant Flows

Let $G=(V,E)$ be the input directed graph with unit edge capacities, s be the source vertex, and t be the sink. Let λ be the value of (s,t)-max-flow in G. We present in this section construction of a compact family $\mathcal{B}$ of $2\lambda+1$ (s,t)-flows in G with the property that, for each edge e in G, there exists a maximum flow for the graph $G-e$ lying in family $\mathcal{B}$.

We will first establish the following theorem.

▶ **Theorem 4.1.** *Given any n-vertex directed graph $G=(V,E)$ with (s,t)-max-flow of value λ, we can compute in polynomial time a family $\mathcal{A}=\{f_1,\ldots,f_{\lambda+1}\}$ of $\lambda+1$-max-flows such that for any non-critical edge $e\in E$, there exists a flow f_i satisfying $f_i(e)=0$.*

We compute an auxiliary graph $H=(V,E,c)$, where for each edge $e\in E$, $c(e)$ is defined as follows:

$$c(e)=\begin{cases}\lambda+1 & \text{if } e \text{ is critical in } G,\\ \lambda & \text{otherwise.}\end{cases}$$

▶ **Lemma 4.2.** $\textsc{max-flow}(s,t,H)=\lambda(\lambda+1)$.

Proof. Consider an (s,t)-min-cut C in G. As each edge in C has a capacity of $\lambda+1$ in H, the capacity of this cut in H is $\lambda(\lambda+1)$. So, the (s,t)-min-cut value in H is at most $\lambda(\lambda+1)$.

Next, observe that any (s,t)-cut C that is not a min-cut in G must contain at least $\lambda+1$ edges. Since each edge in H has a capacity at least λ, the capacity of any cut C that is not a min-cut in G is at least $\lambda(\lambda+1)$ in H. This proves that $\textsc{max-flow}(s,t,H)=\lambda(\lambda+1)$. ◀

Let f be integral (s,t)-max-flow of value $\lambda(\lambda+1)$ in H. The main idea behind computing the family $\mathcal{A}$ is to iteratively peel off from f, λ units of flow, in a total of $(\lambda+1)$ rounds. In order to proceed, we present the following lemma.

▶ **Lemma 4.3.** *Let h be an integral (s,t)-flow in H of value λi (assuming $i \in \mathbb{Z}^+$) such that $h(e) \leqslant i$, for each $e \in E$. Then we can compute a max-flow, say f_i, in G with 0/1 values satisfying*

$$f_i(e) = \begin{cases} 0 & \text{if } h(e) = 0, \\ 1 & \text{if } h(e) = i. \end{cases} \quad \text{for each edge } e \in E.$$

Proof. To compute the flow f_i, we construct an instance $\widetilde{G} = (V, E, d, \ell, \mu)$ of the circulation problem with upper and lower limits as described below:

1. $d(s) = -\lambda$, $d(t) = \lambda$ and $d(v) = 0\ \forall v \neq s, t$
2. For each $e \in E$, $\mu(e) = \min\{1, h(e)\}$
3. For each $e \in E$, $\ell(e) = 1$ if $h(e) = i$, and $\ell(e) = 0$ otherwise.

It suffices to show that $\widetilde{G}$ has a circulation. A circulation exists in $\widetilde{G}$ if and only if, for every cut (A, B) in $\widetilde{G}$, the following inequality holds.

$$d(B) + \ell(B, A) \ \leqslant\ \mu(A, B),$$

where,

$$d(B) = \sum_{v \in B} d(v), \quad \ell(B, A) = \sum_{\substack{(x,y)\in E,\\ x\in B,\ y\in A}} \ell(x, y), \quad \text{and} \quad \mu(A, B) = \sum_{\substack{(x,y)\in E,\\ x\in A,\ y\in B}} \mu(x, y).$$

Consider a cut (A, B) in H. Let $\mathbb{1}_{t\in B}$ be an indicator variable that takes the value 1 if t lies in B, and 0 otherwise. Similarly, define $\mathbb{1}_{s\in B}$. So, the net flow under h from A to B is

$$\lambda i(\mathbb{1}_{t\in B} - \mathbb{1}_{s\in B}).$$

Let α be the number of edges in H lying in set $B \times A$ that carry a flow i under h. Thus, the edges in H lying in $A \times B$ must be carrying a flow of at least

$$(\alpha i)\ +\ (\lambda i)(\mathbb{1}_{t\in B} - \mathbb{1}_{s\in B}).$$

As each edge in H carries a flow at most i under h, the number of edges in the set $A \times B$ that were carrying a non-zero flow with respect to h must be at least $\alpha + \lambda(\mathbb{1}_{t\in B} - \mathbb{1}_{s\in B})$. This implies

$$\mu(A, B) \geqslant \alpha + \lambda(\mathbb{1}_{t\in B} - \mathbb{1}_{s\in B}).$$

Next recall that there are α edges in H from B to A that carry a flow i under h, and for each such edge e, we have $\ell(e) = 1$. Thus, $\ell(B, A) = \alpha$. Further, $d(B) = \lambda(\mathbb{1}_{t\in B} - \mathbb{1}_{s\in B})$. This establishes that $d(B) + \ell(B, A) \ \leqslant\ \mu(A, B)$, for every cut (A, B) in $\widetilde{G}$. Therefore, a circulation exists in $\widetilde{G}$. ◀

▶ **Lemma 4.4.** *We can compute in polynomial time a family $\mathcal{A} = \{f_1, \ldots, f_{\lambda+1}\}$ of $(\lambda + 1)$ integral max-flows in G satisfying $f = f_1 + \cdots + f_{\lambda+1}$.*

Proof. We begin with the (s,t)-max-flow f in H, which satisfies the conditions of Lemma 4.3 for $i = \lambda + 1$. So, $f(e) \leqslant \lambda + 1$ for all edges $e \in E$, and value of f is $\lambda(\lambda + 1)$. On applying Lemma 4.3 to flow $h_{\lambda+1} := f$ in H, we obtain the flow $f_{\lambda+1}$.

This process is then repeated for each i from λ down to 1. For each i, we compute f_i by considering the flow $h_i := f - (f_{i+1} + \cdots + f_{\lambda+1})$ in the graph H. It is important to note that $0 \leqslant h_i(e) \leqslant i$ for all edges $e \in E$. This is due to the fact that, in any previous invocation of Lemma 4.3, say during round j (where $j > i$), if $h_j(e) = j$, then $f_j(e) = 1$; similarly, if $h_j(e) = 0$, then $f_j(e) = 0$. Given that value of h_i is λi, we can compute f_i by applying Lemma 4.3 to the flow h_i in H. This ultimately results in a sequence of $\lambda + 1$ integral max-flows, $f_1, f_2, \ldots, f_{\lambda+1}$, in G which satisfy the equation $f = f_1 + f_2 + \cdots + f_{\lambda+1}$. ◀

Consider the family $\mathcal{A} = \{f_1, \ldots, f_{\lambda+1}\}$ obtained from Lemma 4.4. We now show that for any non-critical edge e, there exists a flow $f_i \in \mathcal{A}$ such that $f_i(e) = 0$. Consider a non-critical edge e in G. Recall that, by the construction of H, we have $c(e) = \lambda$. Since $f(e) \leqslant c(e)$, it follows that

$$f_1(e) + \cdots + f_{\lambda+1}(e) \leqslant \lambda.$$

Thus, there must exist some flow $f_i \in \mathcal{A}$ such that $f_i(e) = 0$. This concludes the proof of Theorem 4.1.

Construction of family $\mathcal{B}$

We next extend our construction to obtain the following result.

▶ **Theorem 4.5.** *For any graph G with (s,t)-max-flow λ, we can compute in polynomial time a family $\mathcal{B}$ of $2\lambda+1$ (s,t)-flows in G satisfying the following:*

For each edge e, the family $\mathcal{B}$ contains an (s,t)-max-flow for $G - e$.

Proof. Let $\mathcal{A} = \{f_1, \ldots, f_{\lambda+1}\}$ be the family of flows obtained from Theorem 4.1. Then, for any non-critical edge e, there exists a flow $f_i \in \mathcal{A}$ such that $f_i(e) = 0$.

Next, fix an (s,t)-flow $\tilde{f} \in \mathcal{A}$. Let $\mathcal{P} = \{P_1, \ldots, P_\lambda\}$ be a family of λ edge-disjoint (s,t)-paths satisfying that for each edge $e \in E$, we have $\tilde{f}(e) = 1$ if and only if e is contained in one of the paths in $\mathcal{P}$. Define a collection of λ flows, $g_1, \ldots, g_\lambda$, where each flow g_i is obtained by cancelling the flow along path P_i from $\tilde{f}$. For any critical edge e, if $P_i \in \mathcal{P}$ is the path containing e, then g_i is an (s,t)-max-flow for the graph $G - e$.

Thus, the family $\mathcal{A} \cup \{g_1, \ldots, g_\lambda\}$ consists of the required set of (s,t)-flows in G. ◀

We remark that the bound of $(2\lambda+1)$ on the size of the family $\mathcal{B}$ in Theorem 4.5 is existentially tight. Consider any graph G having an (s,t) minimal-cut of size $(\lambda+1)$, say $C_{\lambda+1}$, that does not contain any critical edges. (A simple construction for such a G is a graph with three vertices s, x, t, with λ parallel edges from s to x, and $\lambda+1$ parallel edges from x to t). Additionally, let C_λ be an (s,t)-minimum cut in G. For each potential failure $e \in C_\lambda$, there exists an (s,t)-max-flow in $G - e$ of value $\lambda - 1$, which saturates all edges in $C_\lambda - e$. Similarly, for each failure $e \in C_{\lambda+1}$, there exists an (s,t)-max-flow in $G - e$ of value λ, which saturates all edges in $C_{\lambda+1} - e$. It is easy to observe that these flows must be distinct. This provides us a lower bound of $(2\lambda+1)$ on the size of family $\mathcal{B}$.

5 Maximum-flow Sensitivity Oracle

In this section, we present our sensitivity oracle for reporting the maximum flow in the presence of single and dual edge failures. Let $G = (V, E)$ be an n-vertex directed graph with source s and sink t, and let λ be the value of the (s,t)-max-flow in G.

We begin by transforming G into another graph $\mathcal{G} = (V, \mathcal{E})$ as follows. Initialize $\mathcal{G}$ as G, and next iteratively remove those edges e from $\mathcal{G}$ that do not lie in any minimal-cut of size λ or $\lambda+1$ in $\mathcal{G}$.

▶ **Observation 5.1.** *$\mathcal{G}$ is a minimal subgraph of G satisfying the following:*

1. *MAX-FLOW(s,t,G) = MAX-FLOW$(s,t,\mathcal{G})$, and*
2. *MAX-FLOW$(s,t,G-e)$ = MAX-FLOW$(s,t,\mathcal{G}-e)$, for each $e \in E(G)$.*

Throughout this section, we use $\mathcal{A}$ and $\mathcal{B}$ to denote the families of flows obtained by applying Theorem 4.1 and Theorem 4.5 to the graph $\mathcal{G}$.

So, for any flow $f \in \mathcal{B}$, $\text{NULL}_{\mathcal{G}}(f)$ represents the collection of those edges in $\mathcal{G}$ that carry zero flow under f. The lemma below bounds the size of set $\text{NULL}_{\mathcal{G}}(f)$, for any flow $f \in \mathcal{B}$.

▶ **Lemma 5.2.** *For any flow $f \in \mathcal{B}$, the cardinality of the set $\text{NULL}_{\mathcal{G}}(f)$ is at most $3n$. Furthermore, the number of edges in $\mathcal{G}$ is at most $O(\lambda n)$.*

Proof. Consider any max-flow $f \in \mathcal{A}$. By Lemma 2.4, the size of the set $\text{NULL}_{\mathcal{G}}(f, \min+1)$ is at most $2n$. Observe that each edge in $\mathcal{G}$ is contained in either an (s,t)-min-cut or an (s,t)-$(\min+1)$-cut in $\mathcal{G}$. Thus, $\text{NULL}_{\mathcal{G}}(f) = \text{NULL}_{\mathcal{G}}(f, \min+1)$, as the critical edges in $\mathcal{G}$ cannot carry zero flow. This proves a bound of $2n$ on the cardinality of the set $\text{NULL}_{\mathcal{G}}(f)$.

Now, let $\tilde{f}$ be the representative flow in $\mathcal{A}$ used to compute the flows $g_1, \ldots, g_\lambda \in \mathcal{B} \setminus \mathcal{A}$. Then each flow in $\mathcal{B} \setminus \mathcal{A}$ is obtained by cancelling one-unit flow along an (s,t)-path in $\mathcal{G}$ from $\tilde{f}$. For each g_i, it follows that $\text{NULL}_{\mathcal{G}}(g_i)$ differs from $\text{NULL}_{\mathcal{G}}(\tilde{f})$ in at most n edges, since cancellation of flow along a single path changes the number of edges in the NULL set by at most n. Given that $|\text{NULL}_{\mathcal{G}}(\tilde{f})|$ is bounded by $2n$, we conclude that $|\text{NULL}_{\mathcal{G}}(g_i)|$ is bounded by $3n$, for each $i \in [1, \lambda]$.

We now prove the second part. The number of critical edges in $\mathcal{G}$ is at most λn, as the value of the (s,t)-max-flow in $\mathcal{G}$ is λ. Since each non-critical edge in $\mathcal{G}$ is contained in an (s,t)-$(\min+1)$-cut, the non-critical edges are contained in the union $\bigcup_{f \in \mathcal{A}} \text{NULL}_{\mathcal{G}}(f, \min+1)$. Hence, the total number of edges in $\mathcal{G}$ is bounded by $O(\lambda n)$. ◀

As a corollary of Lemma 5.2, note that the following extension of Theorem 4.5 is immediate.

▶ **Theorem 5.3.** *For any graph G with (s,t)-max-flow λ, we can compute in polynomial time a family $\mathcal{B}$ of $2\lambda+1$ (s,t)-flows in G satisfying the following:*

1. *For each edge e, the family $\mathcal{B}$ contains an (s,t)-max-flow for $G - e$.*
2. *For any two flows $f, f' \in \mathcal{B}$, the flows f and f' disagree on at most $O(n)$ edges.*

5.1 Reporting Maximum Flow under Single Edge Failure

We now present a sensitivity oracle for answering (s,t)-max-flow queries in the presence of a single edge failure. Our oracle stores the following:

- A dictionary of the edges in $E(\mathcal{G})$. Additionally, for each flow $f \in \mathcal{B}$, the oracle stores a dictionary of the set $\text{NULL}_{\mathcal{G}}(f)$.
- Label of a canonical (s,t)-maximum flow of the graph $\mathcal{G}$, denoted $\tilde{f}$, where $\tilde{f} \in \mathcal{A}$.
- For each edge $e \in E(\mathcal{G})$, the label of a canonical flow $f_e \in \mathcal{B}$ such that f_e is an (s,t)-max-flow in $\mathcal{G} - e$.

Due to Lemma 5.2, the cardinality of the set $\text{NULL}_{\mathcal{G}}(f)$ is at most $3n$, for any $f \in \mathcal{B}$; and the number of edges in $\mathcal{G}$ is at most $O(\lambda n)$. Therefore, the size of the data structure is $O(\lambda n)$.

Before presenting the query algorithm, we state the following observation.

▶ **Observation 5.4.** *For any edge $e \in E(G)$ and any flow $f \in \mathcal{B}$, the flow through e under f can be determined in $O(1)$ time.*

Proof. Consider a flow $f \in \mathcal{B}$. Observe that $f(e) = 1$ if and only if e lies in $\mathcal{G}$ but is not contained in $\text{NULL}_{\mathcal{G}}(f)$. Since we store dictionaries for both $E(\mathcal{G})$ and $\text{NULL}_{\mathcal{G}}(f)$, querying whether e lies in $E(\mathcal{G}) \setminus \text{NULL}_{\mathcal{G}}(f)$ requires just $O(1)$ time. ◀

Algorithm 1 QUERY-EDGE-FLOW(e, x).

```
// Query flow through edge x
   after failure of e
if f̃(e) = 0 then
  | Return f̃(x)
else
  | Let f_e be the canonical flow in B
  |  satisfying f_e is max-flow in G − e;
  | Return f_e(x);
```

Algorithm 2 REPORT-FLOW-DIFF(e).

```
// Report edges whose flow is
   altered after failure of e
if f̃(e) = 0 then
  | Return ∅
else
  | Let f_e be the canonical flow in B
  |  satisfying f_e is max-flow in G − e;
  | Return NULL_G(f̃) ⊕ NULL_G(f_e);
```

Our max-flow sensitivity oracle supports the following two natural operations upon failure of an edge in the graph.

Query flow through a given edge x

For a failing edge e, if $\tilde{f}(e) = 0$ then there is no change in the flow, so we simply return $\tilde{f}(x)$. Otherwise, we use the precomputed flow f_e (a max-flow in $\mathcal{G} - e$) to return $f_e(x)$. Both these steps require only $O(1)$ time due to Observation 5.4. See Algorithm 1.

Report the set of edges whose flow value changes

Consider a failing edge e in G. If $\tilde{f}(e) = 0$, then there is no change in the flow, so we simply return the empty-set. If not, we retrieve the label of flow $f_e \in \mathcal{B}$ that is an (s,t)-max-flow in $\mathcal{G} - e$. Next, to identify the edges over which $\tilde{f}$ and f_e differ, we simply traverse the sets $\text{NULL}_{\mathcal{G}}(\tilde{f})$ and $\text{NULL}_{\mathcal{G}}(f_e)$, and output their symmetric difference. This operation takes $O(n)$ time, as by Lemma 5.2, the size of both the sets involved is at most $3n$. See Algorithm 2.

This completes our discussion on the max-flow sensitivity oracle for single failure.

5.2 Reporting Maximum Flow under Dual Edge Failures

We now address the problem of designing a sensitivity oracle for reporting the (s,t)-max-flow after the failure of any two edges. The foundation of our approach is the fault-tolerant flow family $\mathcal{B}$ and 1-FT-SCC certificate of [34].

Before presenting the query procedure, we state some structural observations, which are essential for efficiently reporting flows.

▶ **Observation 5.5.** *Suppose H is a directed graph, f is an (s,t)-max-flow in H, and e is an edge with $f(e) = 1$. Then the flow through e can be rerouted along an alternate path if and only if there exists a cycle in the residual graph H_f that contains e^{rev}.*

▶ **Observation 5.6.** *For any graph H with (s,t)-max-flow f and any edge e satisfying $f(e) = 0$, the residual graph $(H - e)_f$ is same as $H_f - e$.*

Given a cycle C in H_f, we define $K(C)$ to be the collection of edges in G corresponding to C. More specifically, $K(C) = (E(C) \cap E(G)) \cup \{e^{\text{rev}} \mid e \in E(C) \setminus E(G)\}$. Switching flow through a cycle C in the residual graph corresponds to toggling the set of saturated edges along $K(C)$.

Query Algorithm

Given a pair of failing edges e and e', the query algorithm proceeds as follows. First, we retrieve a canonical flow f in $\mathcal{B}$ that is a maximum (s,t)-flow in $G-e$. If this flow already avoids both failures, i.e., $f(e)=0$ and $f(e')=0$, then it is also a maximum flow in $G-\{e,e'\}$, and so the change in the edges carrying flow is simply $X = \text{NULL}_{\mathcal{G}}(\tilde{f}) \oplus \text{NULL}_{\mathcal{G}}(f)$.

Now, suppose $f(e')=1$. By Observation 5.6, we have $G_f - e$ is the residual graph of $G-e$ with respect to f. We next determine if there exists a directed cycle in $G_f - e$ containing $(e')^{\text{rev}}$, if so, we can reroute the flow along this cycle.

If, however, no such cycle exists, then flow through e' cannot be rerouted and its endpoints must belong to different strongly connected components in $G_f - e$. In this situation, we must reduce the total flow by one unit, which is achieved by canceling a unit of flow along an (s,t) path that traverses e'. To accomplish this, we take the help of graph $G_f + (s,t) - e$ (where a direct edge (s,t) is artificially introduced). In this setting, the desired (s,t) path containing e' corresponds to a cycle in $G_f + (s,t) - e$ containing edge $(e')^{\text{rev}}$. See Algorithm 3.

Algorithm 3 Report-Flow-Diff(e,e').

Let f be a flow in $\mathcal{B}$ maximizing $val(f)$ such that $f(e)=0$;
Compute $X = \text{NULL}_{\mathcal{G}}(\tilde{f}) \oplus \text{NULL}_{\mathcal{G}}(f)$;
if $f(e')=0$ **then** return X;
else if $\exists$ *a cycle* C_1 *in* $G_f - e$ *containing* $(e')^{rev}$ **then** return $X \oplus K(C_1)$;
else find a cycle C_2 in $G_f + (s,t) - e$ containing $(e')^{rev}$, and return $X \oplus K(C_2)$;

Use of SCC Preservers

To realize the query algorithm efficiently, the only additional structure we need is, for each $f \in \mathcal{B}$, a way to detect cycles in the residual graphs $G_f - e$ and $G_f + (s,t) - e$. This is achieved by storing, for each f, the 1-fault-tolerant SCC preserver of Georgiadis et al. [34]. which is a subgraph with $O(n)$ edges maintaining all strongly connected components under single-edge failures. With this, checking for the existence of cycles passing through an edges takes $O(n)$ time; and this does not inflate the overall space usage.

▶ **Theorem 5.7** (Georgiadis et al. [34]). *For any directed graph H, there exists a subgraph H_0 on $O(n)$ edges such that for any edge $e \in E(H)$, the strongly connected components of $H-e$ and $H_0 - e$ are identical.*

The data structure thus consists of (a) dictionaries for $E(\mathcal{G})$ and sets $\text{NULL}_{\mathcal{G}}(f)$ for $f \in \mathcal{B}$, and (b) the 1-FT-SCC preservers for G_f and $G_f + (s,t)$ for all f. Since $|\mathcal{B}| = O(\lambda)$, the total space remains $O(\lambda n)$. We now discuss the time complexity of Algorithm 3. Identifying the flow f just takes $O(1)$ time. Then, finding the cycle in the appropriate SCC preserver further takes $O(n)$ time, as number of edges in any FT SCC-preserver is $O(n)$. Hence, the total time to report all edges where the flows differ is $O(n)$.

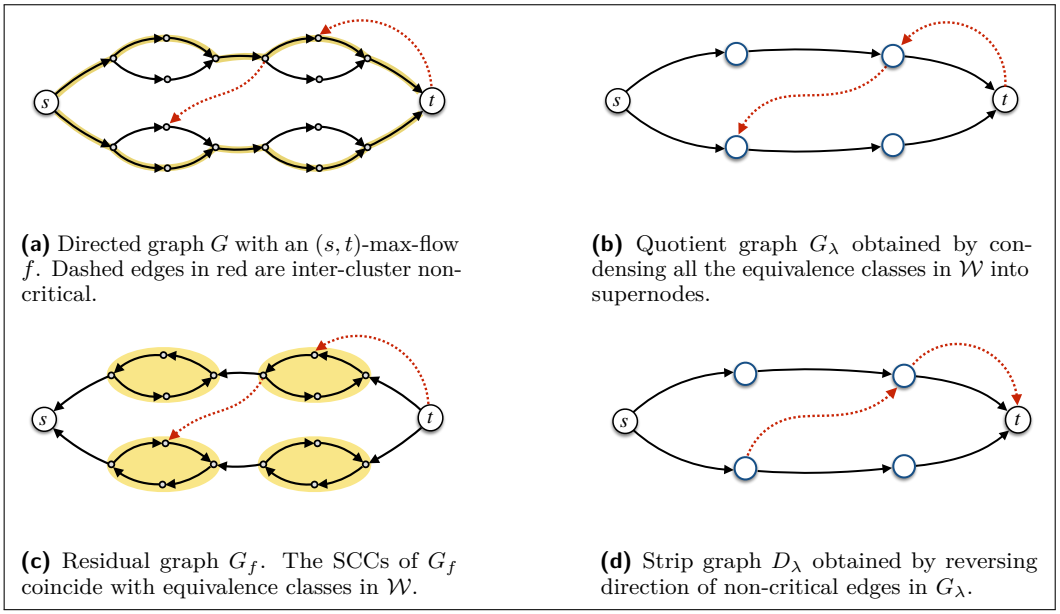
▶ **Theorem 5.8.** *There exists an $O(\lambda n)$-space oracle that, for any pair of failing edges e, e', efficiently reports all edges whose flow value is altered in a maximum (s,t)-flow after failure of e and e'.*

6 An Overview of Structural Properties of (s,t)-Min-Cuts

In this section, we discuss some structural properties of (s,t)-min-cuts that will be essential for the computation of min-cut sensitivity oracles.

Consider a relation on vertex-set of G under which any two vertices x and y are said to be related if and only if they are not separated by any (s,t)-min-cut in G. Let $\mathcal{W}$ be the collection of equivalence classes of V induced by this relation, and for any $v \in V$, let $\boldsymbol{v}$ denote the equivalence class of v in G. We say an edge (x,y) in G is *inter-cluster* if $\boldsymbol{x} \neq \boldsymbol{y}$, and *intra-cluster* otherwise.

Observe that a critical edge is always inter-cluster; however, in directed graphs the converse is not necessarily true (see Section 6).



(a) Directed graph G with an (s,t)-max-flow f. Dashed edges in red are inter-cluster non-critical.

(b) Quotient graph G_λ obtained by condensing all the equivalence classes in $\mathcal{W}$ into supernodes.

(c) Residual graph G_f. The SCCs of G_f coincide with equivalence classes in $\mathcal{W}$.

(d) Strip graph D_λ obtained by reversing direction of non-critical edges in G_λ.

Figure 3 Depiction of graphs G_λ and D_λ. Note that the strip graph D_λ is precisely the DAG of the SCCs of G_f, but with all edge directions reversed.

Throughout this paper, we use G_λ to denote the quotient graph of G induced by the relation defined above, and D_λ to refer to the graph obtained by reversing the direction of all non-critical inter-cluster edges in G_λ, where λ denotes the value of the (s,t)-max-flow in the graph G.

Picard and Queyranne [50], Dinitz and Vainshtein [25] referred to D_λ as *strip graph* and presented the following properties of D_λ (under Assumption 2.5).

▶ **Property 6.1** ([50]). *For any (s,t)-max-flow f in G, the SCCs of residual graph G_f correspond to the equivalence classes $\mathcal{W}$ of G. Moreover, D_λ is essentially the DAG obtained by reversing the edges of graph obtained by contracting the SCCs of G_f into supernodes.*

Figure 3 presents a depiction of the graphs G_λ and D_λ, along with the alternate characterization of D_λ as described in the above property.

▶ **Property 6.2** ([25, 50]). *An (s,t)-cut C in G is a minimum cut if and only if C comprises of critical inter-cluster edges, and the edges of the cut intersect any path in D_λ at most once.*

7 Dual Fault-Tolerant Min-cut oracle via Fault-Tolerant Flows

We present here a construction of a dual fault-tolerant min-cut oracle that, for any graph G with an (s,t)-min-cut of size λ, uses $O(n\lambda)$ space and can report the size of the min-cut upon the occurrence of failures in constant time.

In the first subsection, we show how to handle failing sets that do not contain any critical edges by employing the fault-tolerant flow family $\mathcal{A}$ computed in Theorem 4.1. In the subsequent subsection, we discuss how to handle failing sets that contain one or more critical edges.

7.1 Handling failure of non-critical edges

Let $\mathcal{A} = \{f_1, \dots f_{\lambda+1}\}$ be collection of $(\lambda+1)$ (s,t)-max-flows obtained by applying Theorem 4.1 on graph G.

▶ **Lemma 7.1.** *Let $e_1, e_2 \in E$ and $f \in \mathcal{A}$ be a max-flow such that $f(e_1) = 0$ and $f(e_2) = 1$. Then, $\textsc{max-flow}(s,t,G-\{e_1,e_2\}) = \lambda - 1$ if and only if the endpoints of e_2 are not strongly connected in $G_f - e_1$, where G_f is the residual graph corresponding to flow f.*

Proof. Since $\textsc{max-flow}(s,t,G-e_1) = \lambda$, the size of (s,t)-min-cut in $G-\{e_1,e_2\}$ is $\lambda-1$ if and only if e_2 is a critical edge in $G-e_1$.

Observe that critical edges in a graph are those inter-cluster edges that are saturated with respect to every (s,t)-max-flow. Likewise, non-critical inter-cluster edges are those inter-cluster edges that carry a zero flow with respect to every (s,t)-max-flow. Indeed, if (A,B) is an (s,t)-min-cut separating endpoints of a non-critical edge e, then e would be directed from set B to set A. Since edges directed from sink-side to source-side of an (s,t)-min-cut carry a zero flow with respect to every max-flow, we must have $f(e) = 0$, for every (s,t)-max-flow f.

Since $f(e_2) = 1$, edge e_2 is a critical edge in $G-e_1$ if and only if e_2 is inter-cluster edge in $G-e_1$, that is, the endpoints of e_2 lie in different SCCs in the residual graph $(G-e_1)_f = G_f - e_1$. This proves the claim. ◀

In order to use the above lemma to answer min-cut queries we need an efficient data structure for strong-connectivity upon edge failures in residual graphs. For this, we use the following result by Georgiadis, Italiano, and Parotsidis [34].

▶ **Lemma 7.2** (Georgiadis et al. [34])**.** *For any n vertex directed graph $G = (V,E)$, there exists an $O(n)$ sized data structure $\mathcal{O}_{SCC}(G)$ that, given any pair of vertices $x, y \in V$ and an edge $e \in E$, answers in $O(1)$ time whether or not x and y are strongly connected in $G-e$.*

We now describe the construction of oracle that answers min-cut queries on failure of two non-critical edges. Our oracle stores the following information.

- The fault-tolerant strong-connectivity oracle $\mathcal{O}_{SCC}(G_f)$ of graph G_f, for each $f \in \mathcal{A}$.
- For each non-critical edge e contained in a $(\min+1)$-cut, label of a flow in $\mathcal{A}$, denoted f_e, under which edge e carries zero flow.
- A dictionary of the set $\bigcup_{f\in\mathcal{A}} \textsc{null}_G(f, \min+1)$. Additionally, for each flow $f \in \mathcal{A}$, the oracle stores a dictionary of the set $\textsc{null}_{\mathcal{G}}(f, \min+1)$.

It is easy to verify that the oracle takes $O(\lambda n)$ space.

Query Oracle

Consider a pair of non-critical edges, e and e'. We first verify whether e as well as e' are contained in some $(\min+1)$-cut, i.e., lie in the union $\bigcup_{f\in\mathcal{A}} \text{NULL}_G(f, \min+1)$. Checking their membership in this union suffices as for each non-critical edge, there exists a flow $f \in \mathcal{A}$ under which the edge carries zero flow.

If either e or e' is not contained in $\bigcup_{f\in\mathcal{A}} \text{NULL}_G(f, \min+1)$, the min-cut size remains unchanged, as there will be no $(\lambda+1)$-minimal cut containing these edges.

Next, assume both e and e' are contained in $\bigcup_{f\in\mathcal{A}} \text{NULL}_G(f, \min+1)$. Recall $f_e \in \mathcal{A}$ denotes the max-flow under which e carries zero flow. By Lemma 7.1, the (s,t)-min-cut size decreases by one if and only if the endpoints of e' are in different strongly connected components (SCCs) in $G_{f_e} - e$, and $f_e(e') = 1$.

The first condition, which checks whether the endpoints of e' belong to different SCCs, can be verified in $O(1)$ time using the $\mathcal{O}_{SCC}(G_{f_e})$ oracle. The second condition, $f(e') = 1$, is verifiable by checking that e' is not present in the dictionary of $\text{NULL}_G(f_e, \min+1)$, a check that can also be performed in constant time. Thus, the query time of the oracle is $O(1)$.

Algorithm 4 ReportMinCut(e, e').

```
// Reports size of (s,t)-min-cut after failure of non-critical edges e, e'
if Either e or e' is a critical edge then
    Return "Invalid input";
if Both e and e' are contained in ⋃_{f∈A} NULL_G(f, min+1) then
    if e' ∉ NULL_G(f_e, min+1) and endpoints of e' are in different SCC in G_{f_e} − e then
        Return (λ − 1);
Return (λ);
```

7.2 Handling failing sets containing at least one critical edge

To handle the failure of one or more critical edges, we introduce a general data structure that, given any set of k edge failures, determines whether the (s,t)-min-cut size in G decreases by exactly k in $O(k^2)$ time. This is formalized as follows.

▶ **Theorem 7.3.** *For any n-vertex directed graph G with source s, sink t, and (s,t)-min-cut of size λ, there exists an $O(\lambda n)$-sized oracle, $\mathcal{O}_{MINCUT}(s,t,G)$, that, for any set F of k edges, determines in $O(k^2)$ time whether the (s,t)-min-cut size decreases by k upon the failure of F.*

Furthermore, if the min-cut size decreases by exactly k, the oracle can compute and report an (s,t)-min-cut in $G - F$ in $O(kn)$ time.

Now consider a failing set $F = \{e, e'\}$ comprising two edges in G, where at least one edge in F, say e, is a critical edge. Upon the failure of F, the size of the (s,t)-min-cut decreases by at most two and at least one (since the min-cut size in $G - e$ is exactly $\lambda - 1$). Furthermore, using Theorem 7.3, we can construct a data structure of size $O(\lambda n)$ that verifies, in constant time, whether the min-cut size decreases by exactly two when F fails.

Combined with the discussion in Section 7.1, this gives an $O(n\lambda)$-sized data structure that, for any set F of two edge failures, reports the size of the (s,t)-min-cut in $G - F$ in constant time. The details of our data structure for reporting a min-cut after dual failures is deferred to the full version of the paper. This concludes the proof of Theorem 1.3.

8 Fault-tolerant Min-cut oracle for k edge failures

In this section, we present our (s,t)-min-cut sensitivity oracle resilient to k failures. We begin with the following lemma.

▶ **Lemma 8.1.** *For any failing set F of size k, MIN-CUT$(s,t,G-F)$ is given by*

$$\min\left\{|C|-|F_0| \;\middle|\; C \text{ is minimal cut in } G \text{ of size at most } \lambda+k, F_0 = F\cap C\right\}.$$

The above lemma highlights the significance of $\lambda+i$ minimal-cuts for answering min-cut queries under failures. In order to use the this lemma, one would require a data structure that can check, for every subset $F_0 \subseteq F$, whether there exists an (s,t)-minimal cut C of size at most $\lambda+k$ containing F_0. While we provide a data structure in Theorem 7.3 to efficiently compute an (s,t)-min-cut containing a given subset F_0, extending this to handle all $\lambda+k$ minimal cuts is non-trivial.

To address this, we leverage the result by Kim et al. [43], which shows how to augment graphs with additional edges to transform a large collection of minimal cuts into min-cuts. This allows us to efficiently utilize data-structure developed for min-cuts in Theorem 7.3.

For any set $\mathcal{E} \subseteq V\times V$, let $G+\mathcal{E}^\infty$ denote the graph obtained by adding edges in $\mathcal{E}$ to G, with infinite capacity on all edges in $\mathcal{E}$. We use the following result by Kim et al. [43].

▶ **Theorem 8.2** ([43])**.** *There exists a randomized polynomial-time algorithm that, given a directed graph $G=(V,E)$, two vertices $s,t\in V$, and an integer L, outputs a set $\mathcal{E}\subseteq V\times V$ such that for every (s,t)-minimal-cut $Z\subseteq E$ of size at most L, with probability $2^{-O(L^4\log L)}$, Z remains an (s,t)-cut in $G+\mathcal{E}^\infty$, and furthermore, Z is an (s,t)-min-cut in $G+\mathcal{E}^\infty$.*

Oracle construction

Let $L=\lambda+k$ and $\rho = 2^{O(L^4\log L)}\cdot(4L\log_e n)$. We perform ρ independent rounds, and in each round, we compute a sample $\mathcal{E}$ by invoking Theorem 8.2. Let $\mathcal{C}$ be a collection of those samples $\mathcal{E}\in V\times V$ for which $G+\mathcal{E}^\infty$ has an (s,t)-min-cut of size at most L. For each $\mathcal{E}\in\mathcal{C}$, compute the oracle $\mathcal{O}_{MINCUT}(s,t,G+\mathcal{E}^\infty)$ using the data structure from Theorem 7.3. Finally, verify in $m^{O(L)}$ time that for every (s,t)-minimal-cut Z of size at most L, there exists some $\mathcal{E}\in\mathcal{C}$ satisfying Z is an (s,t)-min-cut in $G+\mathcal{E}^\infty$. If not, re-compute $\mathcal{C}$. The total space required by our data structure is $O(\rho\cdot nL) = \left(2^{O(L^4\log L)}n\log n\right)$, since the graphs in family $\mathcal{C}$ have a min-cut of size at most L. Lemma 8.3 below shows that the number of repetitions needed to compute $\mathcal{C}$ is at most one with high probability.

▶ **Lemma 8.3.** *With probability at least $1-\frac{1}{n^2}$, for each (s,t)-minimal cut Z of size at most L, there exists $\mathcal{E}\in\mathcal{C}$ such that Z is an (s,t)-min-cut in $G+\mathcal{E}^\infty$.*

Proof. Consider an (s,t)-minimal cut Z. The probability that there does not exist an $\mathcal{E}\in\mathcal{C}$ such that Z is an (s,t)-min-cut in $G+\mathcal{E}^\infty$ is

$$\left(1-\frac{1}{2^{O(L^4\log L)}}\right)^{\rho} = \left(1-\frac{1}{2^{O(L^4\log L)}}\right)^{2^{O(L^4\log L)}\cdot(4L\log_e n)} \leqslant e^{-4L\log_e n} = \frac{1}{n^{4L}}.$$

The total number of minimal cuts of size at most L is bounded by n^{2L}. Therefore, the probability that there exists a minimal cut Z for which no $\mathcal{E}\in\mathcal{C}$ satisfies that Z is an (s,t)-min-cut in $G+\mathcal{E}^\infty$ is at most $n^{-2L}\leqslant n^{-2}$. This completes the proof. ◀

Query procedure

Our algorithm to determine the size of the minimum cut after k failures is presented in Algorithm 5. The procedure iterates over every subset $F_0 \subseteq F$ and every set $\mathcal{E} \in \mathcal{C}$, and checks using Theorem 7.3 whether the size of the min-cut in $G + \mathcal{E}^\infty$ decreases by exactly $|F_0|$ upon the failure of F_0.

It then reports the minimum value of $\text{MIN-CUT}(s, t, G + \mathcal{E}^\infty) - |F_0|$, taken over all subsets $F_0 \subseteq F$ and all $\mathcal{E} \in \mathcal{C}$ such that the (s,t)-min-cut size in $G + \mathcal{E}^\infty$ decreases by exactly $|F_0|$ upon the failure of F_0. The time complexity to compute the size of the min-cut in $G - F$ is therefore $O(k^2|\mathcal{C}|) = O\big(2^{O(L^4 \log L)} \log n\big)$, where $L = \lambda + k$.

Furthermore, we can compute an explicit (s,t)-min-cut in $G - F$. To achieve this, we return a min-cut in the graph $G + \mathcal{E}^\infty - F_0$ for the appropriate choice of $\mathcal{E}$ and F_0, as determined in the previous step. Computing and returning this cut requires an additional $O(kn)$ time using the oracle described in Theorem 7.3.

Algorithm 5 ReportMinCut(F).

Initialize $q = \lambda$;
foreach $F_0 \subseteq F$ ***and*** $\mathcal{E} \in \mathcal{C}$ **do**
  if *(s,t)-min-cut size in $G + (\mathcal{E})^\infty$ on failure of F_0 decreases by exactly $|F_0|$* **then**
    $q = \min\Big(q,\ \text{MIN-CUT}(s, t, G + \mathcal{E}^\infty) - |F_0|\Big)$
Return q;

We thus conclude with the following theorem.

▶ **Theorem 8.4.** *There exists a Las Vegas algorithm that, for any n-vertex directed graph G with an (s,t)-min-cut of size λ, computes a k-fault-tolerant minimum cut oracle of $O\big(2^{O(L^4 \log L)} n \log n\big)$ space. This oracle can determine the size of the (s,t)-min-cut in $G - F$ for any set F of k edges in $O\big(2^{O(L^4 \log L)} \log n\big)$ time, where $L = \lambda + k$.*

Furthermore, the oracle can report an (s,t)-min-cut in $G - F$ in an additional computation time of $O(kn)$.

9 Open Problems

Several interesting questions remain unresolved in this field. One immediate open question after our work is whether our max-flow sensitivity oracle can be extended to handle more than two failures. Specifically, can we obtain a compact oracle capable of updating the max-flow in a directed graph after $k > 2$ failures in $O_k(n)$ time?

Another natural open question is computing a compact min-cut sensitivity oracle for graphs with large cuts for $k > 2$ failures. While our min-cut sensitivity oracle for $k = 2$ has a linear dependence on λ, our oracle for $k > 2$ exhibits an exponential dependence on λ. We leave it open to close this gap and achieve a polynomial dependence on λ.

A further open question is whether our (s,t)-reachability oracle (as well as the max-flow and min-cut sensitivity oracles) can be extended to the single-source setting. Currently, for $k = 1, 2$, there exist single-source reachability oracles [17, 45] with linear space that for any query vertex x and any set F of up to two failures answer whether x is reachable from source on failure of F in constant time. However, to date, no oracle exists for answering reachability queries from a fixed source in linear space and truly sub-linear query time, even for a constant $k > 2$.

References

1 Amir Abboud, Loukas Georgiadis, Giuseppe F. Italiano, Robert Krauthgamer, Nikos Parotsidis, Ohad Trabelsi, Przemyslaw Uznanski, and Daniel Wolleb-Graf. Faster algorithms for all-pairs bounded min-cuts. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019*, volume 132 of *LIPIcs*, pages 7:1–7:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.7.

2 Shyan Akmal and Ce Jin. An efficient algorithm for all-pairs bounded edge connectivity. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 11:1–11:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.11.

3 Surender Baswana and Koustav Bhanja. Vital Edges for (s,t)-Mincut: Efficient Algorithms, Compact Structures, & Optimal Sensitivity Oracles. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2024.17.

4 Surender Baswana, Koustav Bhanja, and Abhyuday Pandey. Minimum+1 (s, t)-cuts and dual edge sensitivity oracle. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 15:1–15:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.15.

5 Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: generic and optimal. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 509–518, 2016. doi:10.1145/2897518.2897648.

6 Surender Baswana, Keerti Choudhary, and Liam Roditty. An efficient strongly connected components algorithm in the fault tolerant model. *Algorithmica*, 81(3):967–985, 2019. doi:10.1007/S00453-018-0452-3.

7 Surender Baswana and Abhyuday Pandey. Sensitivity oracles for all-pairs mincuts. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 581–609. SIAM, 2022. doi:10.1137/1.9781611977073.27.

8 Aaron Bernstein, Joakim Blikstad, Thatchaphol Saranurak, and Ta-Wei Tu. Maximum Flow by Augmenting Paths in $n^{2+o(1)}$ Time . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2056–2077, Los Alamitos, CA, USA, October 2024. IEEE Computer Society. doi:10.1109/FOCS61266.2024.00123.

9 Aaron Bernstein and David Karger. Improved distance sensitivity oracles via random sampling. In *SODA'08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 34–43, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1347082.1347087.

10 Koustav Bhanja. Optimal sensitivity oracle for steiner mincut. In Julián Mestre and Anthony Wirth, editors, *35th International Symposium on Algorithms and Computation, ISAAC 2024, December 8-11, 2024, Sydney, Australia*, volume 322 of *LIPIcs*, pages 10:1–10:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ISAAC.2024.10.

11 Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 73:1–73:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.73.

12 Jan Van Den Brand, Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. A Deterministic Almost-Linear Time Algorithm for Minimum-Cost Flow . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 503–514, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi:10.1109/FOCS57990.2023.00037.

13 Diptarka Chakraborty and Keerti Choudhary. New extremal bounds for reachability and strong-connectivity preservers under failures. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 25:1–25:20, 2020. doi:10.4230/LIPIcs.ICALP.2020.25.

14 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. *f*-sensitivity distance oracles and routing schemes. In *18th Annual European Symposium on Algorithms - ESA (1)*, pages 84–96, 2010. doi:10.1007/978-3-642-15775-2_8.

15 Li Chen, Rasmus Kyng, Yang P. Liu, Simon Meierhans, and Maximilian Probst Gutenberg. Almost-linear time algorithms for incremental graphs: Cycle detection, sccs, s-t shortest path, and minimum-cost flow. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 1165–1173, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649745.

16 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-linear-time algorithms for maximum flow and minimum-cost flow. *Commun. ACM*, 66(12):85–92, November 2023. doi:10.1145/3610940.

17 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, pages 130:1–130:13, 2016. doi:10.4230/LIPIcs.ICALP.2016.130.

18 Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008. doi:10.1137/S0097539705429847.

19 Dipan Dey and Manoj Gupta. Nearly optimal fault tolerant distance oracle. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, pages 944–955, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3618260.3649697.

20 E. Dinic, Alexander Karzanov, and M. Lomonosov. The system of minimum edge cuts in a graph. *Studies in Discrete Optimizations, A.A. Fridman, ed., Nauka, Moscow, 290-306, in Russian*, January 1976.

21 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011*, pages 169–178, 2011. doi:10.1145/1993806.1993830.

22 Ye. Dinitz and A. Vainshtein. Locally orientable graphs, cell structures, and a new algorithm for the incremental maintenance of connectivity carcasses. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '95, pages 302–311, USA, 1995. Society for Industrial and Applied Mathematics.

23 Yefim Dinitz. *Dinitz' Algorithm: The Original Version and Even's Version*, pages 218–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. doi:10.1007/11685654_10.

24 Yefim Dinitz and Alek Vainshtein. The connectivity carcass of a vertex subset in a graph and its incremental maintenance. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 716–725. ACM, 1994. doi:10.1145/195058.195442.

25 Yefim Dinitz and Alek Vainshtein. The general structure of edge-connectivity of a vertex subset in a graph and its incremental maintenance. odd case. *SIAM J. Comput.*, 30(3):753–808, 2000. doi:10.1137/S0097539797330045.

26 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009*, pages 506–515, 2009. doi:10.1137/1.9781611973068.56.

27 Ran Duan and Seth Pettie. Connectivity oracles for failure prone graphs. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 465–474, 2010. doi:10.1145/1806689.1806754.

28 Ran Duan and Seth Pettie. Connectivity oracles for graphs subject to vertex failures. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 490–509, 2017. doi:10.1137/1.9781611974782.31.

29 Ran Duan and Hanlin Ren. Maintaining exact distances under multiple edge failures. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 1093–1101, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520002.

30 Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972. doi:10.1145/321694.321699.

31 D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 2010.

32 Yu Gao, Yang P. Liu, and Richard Peng. Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 516–527, 2022. doi:10.1109/FOCS52979.2021.00058.

33 Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1880–1899, 2017. doi:10.1137/1.9781611974782.123.

34 Loukas Georgiadis, Giuseppe F. Italiano, and Nikos Parotsidis. Strong connectivity in directed graphs under failures, with applications. *SIAM Journal on Computing*, 49(5):865–926, 2020. doi:10.1137/19M1258530.

35 Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, September 1998. doi:10.1145/290179.290181.

36 Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, October 1988. doi:10.1145/48014.61051.

37 Gramoz Goranci, Monika Henzinger, Danupon Nanongkai, Thatchaphol Saranurak, Mikkel Thorup, and Christian Wulff-Nilsen. *Fully Dynamic Exact Edge Connectivity in Sublinear Time*, pages 70–86. SIAM, 2023. doi:10.1137/1.9781611977554.ch3.

38 Manoj Gupta and Shahbaz Khan. Multiple source dual fault tolerant BFS trees. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, pages 127:1–127:15, 2017. doi:10.4230/LIPIcs.ICALP.2017.127.

39 Wenyu Jin and Xiaorui Sun. Fully dynamic s-t edge connectivity in subpolynomial time (extended abstract). In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 861–872, 2022. doi:10.1109/FOCS52979.2021.00088.

40 Wenyu Jin, Xiaorui Sun, and Mikkel Thorup. *Fully Dynamic Min-Cut of Superconstant Size in Subpolynomial Time*, pages 2999–3026. SIAM, 2024. doi:10.1137/1.9781611977912.107.

41 Adam Karczmarz and Piotr Sankowski. Sensitivity and Dynamic Distance Oracles via Generic Matrices and Frobenius Form . In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1745–1756, Los Alamitos, CA, USA, November 2023. IEEE Computer Society. doi:10.1109/FOCS57990.2023.00106.

42 Tarun Kathuria, Yang P. Liu, and Aaron Sidford. Unit Capacity Maxflow in Almost $O(m^{4/3})$ Time . In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 119–130, Los Alamitos, CA, USA, November 2020. IEEE Computer Society. doi:10.1109/FOCS46700.2020.00020.

43 Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Directed flow-augmentation. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 938–947. ACM, 2022. doi:10.1145/3519935.3520018.

44 Yin Tat Lee, Satish Rao, and Nikhil Srivastava. A new approach to computing maximum flows using electrical flows. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 755–764, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2488608.2488704.

45 Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979. doi:10.1145/357062.357071.

46 Tamás Lukovszki. New results of fault tolerant geometric spanners. In *Algorithms and Data Structures, 6th International Workshop, WADS '99, Proceedings*, pages 193–204, 1999. doi:10.1007/3-540-48447-7_20.

47 Merav Parter. Vertex fault tolerant additive spanners. In *Distributed Computing - 28th International Symposium, DISC 2014, Proceedings*, pages 167–181, 2014. doi:10.1007/978-3-662-45174-8_12.

48 Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013. doi:10.1007/978-3-642-40450-4_66.

49 Mihai Patrascu and Mikkel Thorup. Planning for fast connectivity updates. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 263–271, 2007. doi:10.1109/FOCS.2007.59.

50 Jean-Claude Picard and Maurice Queyranne. On the structure of all minimum cuts in a network and applications. *Math. Program.*, 22(1):121, 1982. doi:10.1007/BF01581031.

51 Mikkel Thorup. Fully-dynamic min-cut*. *Combinatorica*, 27(1):91–127, February 2007. doi:10.1007/s00493-007-0045-2.

52 Jan Van Den Brand, Li Chen, Rasmus Kyng, Yang P. Liu, Simon Meierhans, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-Linear Time Algorithms for Decremental Graphs: Min-Cost Flow and More via Duality . In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2010–2032, Los Alamitos, CA, USA, October 2024. IEEE Computer Society. doi:10.1109/FOCS61266.2024.00120.

53 Jan van den Brand, Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva, and Aaron Sidford. *Incremental Approximate Maximum Flow on Undirected Graphs in Subpolynomial Update Time*, pages 2980–2998. SIAM, 2024. doi:10.1137/1.9781611977912.106.

54 Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P. Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 543–556, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3519935.3520068.

55 Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and l1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 859–869, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451108.

56 Jan van den Brand and Thatchaphol Saranurak. Sensitive distance and reachability oracles for large batch updates. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–435, 2019. doi:10.1109/FOCS.2019.00034.

57 Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Trans. Algorithms*, 9(2), March 2013. doi:10.1145/2438645.2438646.