


Faster Algorithm for Second (s,t)-Mincut and Breaking Quadratic Barrier for Dual Edge Sensitivity for (s,t)-Mincut

Surender Baswana 

Department of Computer Science & Engineering, IIT Kanpur, India

Koustav Bhanja 

Weizmann Institute of Science, Rehovot, Israel

Anupam Roy 

Department of Computer Science & Engineering, IIT Kanpur, India

Abstract

Let G be a directed graph on n vertices and m edges. In this article, we study (s, t) -cuts of second minimum capacity and present the following algorithmic and graph-theoretic results.

- Second (s,t)-mincut:** Vazirani and Yannakakis [ICALP 1992] designed the first algorithm for computing an (s, t) -cut of second minimum capacity using $\mathcal{O}(n^2)$ maximum (s, t) -flow computations. We present the following algorithm that improves the running time significantly. For directed integer-weighted graphs, there is an algorithm that can compute an (s, t) -cut of second minimum capacity using $\tilde{\mathcal{O}}(\sqrt{n})$ maximum (s, t) -flow computations with high probability.¹ To achieve this result, a close relationship of independent interest is established between (s, t) -cuts of second minimum capacity and global mincuts in directed weighted graphs.
- Minimum+1 (s,t)-cuts:** Minimum+1 (s, t) -cuts have been studied quite well recently [Baswana, Bhanja, and Pandey, ICALP 2022 & TALG 2023], which is a special case of second (s, t) -mincut. We present the following structural result and the first nontrivial algorithm for minimum+1 (s, t) -cuts.
 - Algorithm:** For directed multi-graphs, we design an algorithm that, given any maximum (s, t) -flow, computes a minimum+1 (s, t) -cut, if it exists, in $\mathcal{O}(m)$ time.
 - Structure:** The existing structures for storing and characterizing all minimum+1 (s, t) -cuts occupy $\mathcal{O}(mn)$ space [Baswana, Bhanja, and Pandey, TALG 2023]. For undirected multi-graphs, we design a directed acyclic graph (DAG) occupying only $\mathcal{O}(m)$ space that stores and characterizes all minimum+1 (s, t) -cuts. This matches the space bound of the widely-known DAG structure for all (s, t) -mincuts [Picard and Queyranne, Math. Prog. Studies 1980].
- Dual Edge Sensitivity Oracle:** The study of minimum+1 (s, t) -cuts often turns out to be useful in designing dual edge sensitivity oracles – a compact data structure for efficiently reporting an (s, t) -mincut after insertion/failure of any given pair of query edges. It has been shown recently [Bhanja, ICALP 2025] that any dual edge sensitivity oracle for (s, t) -mincut in undirected multi-graphs must occupy $\Omega(n^2)$ space in the worst-case irrespective of the query time. Interestingly, for undirected unweighted simple graphs, we break this quadratic barrier while achieving a non-trivial query time as follows. There is an $\mathcal{O}(n\sqrt{n})$ space data structure that can report an (s, t) -mincut in $\mathcal{O}(\min\{m, n\sqrt{n}\})$ time after the insertion/failure of any given pair of query edges.

To arrive at our results, as one of our key techniques, we establish interesting relationships between (s, t) -cuts of capacity (minimum+ Δ), $\Delta \geq 0$, and maximum (s, t) -flow. We believe that these techniques and the graph-theoretic result in 2.(b) are of independent interest.

2012 ACM Subject Classification Theory of computation \rightarrow Data structures design and analysis; Theory of computation \rightarrow Dynamic graph algorithms; Theory of computation \rightarrow Network flows

¹ $\tilde{\mathcal{O}}(\cdot)$ hides poly-logarithmic factors.



© Surender Baswana, Koustav Bhanja, and Anupam Roy;
licensed under Creative Commons License CC-BY 4.0

33rd Annual European Symposium on Algorithms (ESA 2025).

Editors: Anne Benoit, Haim Kaplan, Sebastian Wild, and Grzegorz Herman; Article No. 68; pp. 68:1–68:19

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases mincut, second mincut, compact structure, fault tolerant, sensitivity oracle, dual edges, st mincut, global mincut, characterization

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.68

Related Version *Full Version:* <https://arxiv.org/abs/2507.01366>

Funding The research work of Surender Baswana is partially supported by Tapas Mishra Memorial Chair at IIT Kanpur, India. The research work of Koustav Bhanja is partially supported by Merav Parter’s European Research Council (ERC) grant under the European Union’s Horizon 2020 research and innovation programme, grant agreement No. 949083.

Acknowledgements This research work was partially carried out while Koustav Bhanja was pursuing his doctoral studies at the Department of CSE, IIT Kanpur, India.

1 Introduction

The concept of cut is fundamental in graph theory and has numerous real-world applications [1]. Let $G = (V, E)$ be a directed weighted graph on $n = |V|$ vertices and $m = |E|$ edges with a designated source vertex s and a designated sink vertex t . Every edge e of G is assigned with a non-negative real value as the capacity of the edge, denoted by $w(e)$. There are mainly two types of well-studied cuts in a graph – global cuts and (s, t) -cuts. A *global cut*, or simply a *cut*, of G is defined as a nonempty proper subset of V . A cut C is said to be an (s, t) -cut if $s \in C$ and $t \in \overline{C} (= V \setminus C)$. Every cut of G is associated with a capacity defined as follows. The *capacity* of a cut C , denoted by $c(C)$, is the sum of the capacities of every edge (x, y) satisfying $x \in C$ and $y \in \overline{C}$. A global cut (likewise (s, t) -cut) of the least capacity is called a *global mincut* (likewise (s, t) -mincut). Henceforth λ denotes the capacity of (s, t) -mincut.

The study of cuts from both structural and algorithmic perspectives has been an important field of research for the past six decades [21, 19, 14, 25, 36]. In this article, we provide the following two main results for (s, t) -cuts. (1) We present efficient algorithms for computing an (s, t) -cut of second minimum capacity in directed weighted graphs. After more than 30 years, our algorithms provide the first improvement by a polynomial factor over the existing result of Vazirani and Yannakakis [37]. To arrive at this result, we establish that computing an (s, t) -cut of second minimum capacity has the same time complexity as computing a global mincut. (2) We present a dual edge sensitivity oracle for (s, t) -mincut – a compact data structure that efficiently reports an (s, t) -mincut after the insertion or failure of any pair of edges. This is the first dual edge sensitivity oracle for (s, t) -mincut that occupies subquadratic space while achieving nontrivial query time in undirected unweighted simple graphs¹. This breaks the existing quadratic lower bounds [3, 6] on the space for any dual edge sensitivity oracle for (s, t) -mincut in undirected multi-graphs². We arrive at this result by designing a compact structure for storing and characterizing all *minimum+1* (s, t) -cuts (a special case of second minimum (s, t) -cut). This structure improves the space occupied by the existing best-known structure [3] by a linear factor.

We now present the problems addressed in this article, their state-of-the-art, and our results.

¹ A simple graph refers to a graph having no parallel edges.

² A multi-graph refers to an unweighted graph having parallel edges.

1.1 Faster Algorithm for Second Minimum (s,t) -cuts

An algorithmic graph problem aims at computing a structure that optimizes a given function. Examples of such classical problems are Minimum Spanning Tree, Shortest Path, Minimum Cut. Having designed (near) optimal algorithm for such problems, the next immediate objective is the following. Design an efficient algorithm that, given a structure S achieving optimal function value, computes a structure S' that differs from S and achieves the optimal or *next* optimal function value. There has been an extensive study on computing the second minimum spanning tree [8, 26, 29], the second shortest path [38, 18, 34], and second (s,t) -mincut [37]. The algorithms for these problems are so fundamental that they appear in textbooks on algorithms [13, 1]. In addition, they act as the foundation for generalizing the problem to compute k^{th} optimal structure, such as computing k^{th} minimum spanning tree, k^{th} shortest path, k^{th} (s,t) -mincut.

The problem of designing efficient algorithms for computing (s,t) -mincut (or equivalently, maximum (s,t) -flow) has been studied for more than six decades. This problem is now almost settled with the recent breakthrough result on almost linear time algorithm for maximum (s,t) -flow by Van et al. [36]. Interestingly, given an algorithm for maximum (s,t) -flow, we can compute all (s,t) -mincuts implicitly with an additional $\mathcal{O}(m)$ time, as shown by Picard and Queyranne [33]. It is, therefore, natural to redefine the second (s,t) -mincut problem as follows. Design an algorithm that computes an (s,t) -cut of second minimum capacity. For brevity, henceforth we call (s,t) -cut of second minimum capacity by second (s,t) -mincut.

Vazirani and Yannakakis [37] addressed the problem of computing an (s,t) -cut having k^{th} minimum capacity in 1992. For computing any k^{th} minimum capacity (s,t) -cut, they gave an algorithm that uses $\mathcal{O}(n^{2(k-1)})$ maximum (s,t) -flow computations. Conversely, they also argued, using NP-hardness of computing a maximum cut, that exponential dependence on k is unavoidable assuming $P \neq NP$. To arrive at their result, the key problem they addressed is the design of an algorithm that computes a second (s,t) -mincut using $\mathcal{O}(n^2)$ maximum (s,t) -flow computations. They further improve the running time by designing a faster algorithm that computes a second (s,t) -mincut using only $\mathcal{O}(n)$ maximum (s,t) -flow computations. We show that this faster algorithm is incorrect due to a serious error in its analysis. As a result, the existing best-known algorithm for computing a second (s,t) -mincut uses $\mathcal{O}(n^2)$ maximum (s,t) -flow computations [37]. We design an algorithm for the second (s,t) -mincut with a significantly improved running time as shown in the following theorem.

► **Theorem 1 (Second (s,t) -mincut algorithm).** *For any directed graph G on n vertices with integer edge capacities, there is an algorithm that computes a second (s,t) -mincut in G using $\tilde{\mathcal{O}}(\sqrt{n})$ maximum (s,t) -flow computations with high probability.*

The two well-known minimum cuts in a graph are (s,t) -mincut and global mincut. Recently, there has been a growing interest in understanding the difference in the complexity of computing an (s,t) -mincut and computing a global mincut. For undirected weighted graphs, Li and Panigrahi [28] established that the running time of computing a global mincut differs by only poly-logarithmic factors from the running time of computing an (s,t) -mincut. In particular, the authors showed that there is an algorithm that can compute a global mincut using $\tilde{\mathcal{O}}(1)$ maximum (s,t) -flow computations with additional $\tilde{\mathcal{O}}(m)$ time. However, for directed weighted graphs, there is a *large* gap between the running time of computing an (s,t) -mincut and computing a global mincut as follows. Cen et al. [9] designed an algorithm that can compute a global mincut using $\tilde{\mathcal{O}}(\sqrt{n})$ maximum (s,t) -flow computations with additional $\tilde{\mathcal{O}}(m\sqrt{n})$ time. Interestingly, to arrive at our result in Theorem 1, we show that the complexity of computing a global mincut is the same as the complexity of computing a second (s,t) -mincut *modulo* a single maximum (s,t) -flow computation as follows, which might find many other important applications.

► **Theorem 2** (Equivalence between global mincut and second (s,t)-mincut). *For any directed weighted graph G on n vertices and m edges, the following two assertions hold.*

1. *The problem of computing a second (s,t)-mincut is reducible to the problem of computing a global mincut in $\mathcal{O}(MF(m,n))$ time, where $MF(m,n)$ denotes the time complexity for computing a maximum (s,t)-flow in G .*
2. *The problem of computing a global mincut is reducible to the problem of computing a second (s,t)-mincut in $\mathcal{O}(m)$ time.*

► **Remark 3.** Our algorithm for computing a second (s,t)-mincut in Theorem 1 is Monte Carlo and works for graphs with integer edge capacities. However, the equivalence between second (s,t)-mincut and global mincut in Theorem 2 is deterministic and holds even for graphs with real edge capacities.

1.2 Minimum+1 (s,t)-cuts: Efficient Algorithm & Compact Structure

The cuts of capacity minimum+1, known as minimum+1 cuts, have been studied quite extensively in the past [3, 15, 6]. For (un)directed multi-graphs, a minimum+1 (s,t)-cut is a special case of second (s,t)-mincut. We present the following structural result and the first nontrivial algorithm for minimum+1 (s,t)-cuts.

Algorithm. For both minimum+1 global cuts [15] and minimum+1 (s,t)-cuts [3], there exist compact data structures. These data structures have important applications in maintaining minimum+2 edge connected components [15] and designing dual edge sensitivity oracle for (s,t)-mincut [3]. Moreover, there exist efficient algorithms [24, 4, 30] that can compute a global cut of capacity minimum+1. Unfortunately, the existing best-known algorithm for computing an (s,t)-cut of capacity minimum+1 is nothing but the algorithm for computing a second (s,t)-mincut by [37], which uses $\mathcal{O}(n^2)$ maximum (s,t)-flow computations. We present the following result as the first efficient algorithm for computing an (s,t)-cut of capacity minimum+1.

► **Theorem 4** (Minimum+1 (s,t)-cut algorithm). *For any directed multi-graph G on n vertices and m edges, there is an algorithm that, given any maximum (s,t)-flow, computes a minimum+1 (s,t)-cut, if exists in G , in $\mathcal{O}(m)$ time.*

► **Remark 5.** The best-known algorithm for computing an (s,t)-mincut uses one maximum (s,t)-flow computation. It follows from Theorem 4 that the running time of computing a minimum+1 (s,t)-cut differs from the running time of computing an (s,t)-mincut only by additional $\mathcal{O}(m)$ time.

Structure. There are several algorithmic applications on cuts, namely, fault-tolerance [33, 14], dynamic algorithms [22, 23], edge-connectivity augmentation [31, 10] that require an efficient way to distinguish a set of cuts, say minimum cuts or minimum+1 cuts, from the rest of the cuts. A trivial way to accomplish this objective is to store each cut of the required set explicitly. However, this is totally impractical since the set of these cuts is usually quite huge. For example, the number of (s,t)-mincuts are exponential [33], and the number of global mincuts are $\Omega(n^2)$ [14]. This has led the researchers to invent the following concept. A structure H is said to *characterize* a set of cuts \mathcal{C} using a property P if the following condition holds. A cut $C \in \mathcal{C}$ if and only if C satisfies property P in H . It is desirable that H is as compact as possible. Moreover, verifying if a given cut C satisfies P in H has to be as time-efficient as possible.

The design of compact structures for characterizing minimum cuts started with the seminal work of Dinitz, Karzanov, and Lomonosov [14] in 1976. In this work, the well-known cactus graph occupying $\mathcal{O}(n)$ space was invented for storing and characterizing all global mincuts. Several compact structures have been designed subsequently for storing and characterizing cuts of capacity both minimum and near minimum [33, 4, 15, 16, 3]. Quite expectedly, they are playing crucial roles in establishing many important algorithmic results [37, 4, 15, 23, 27, 3]. In particular, compact structures for storing and characterizing all minimum+1 cuts of a graph have been well-studied. For all minimum+1 global cuts in undirected multi-graphs, Dinitz and Nutov [15] constructed an $\mathcal{O}(n)$ space 2-level cactus model that stores and characterizes them. For all minimum+1 (s, t) -cuts in directed multi-graphs, the existing structure that provides a characterization occupies $\mathcal{O}(mn)$ space [3]. Unfortunately, the space occupied by this structure of [3] is significantly inferior to the following widely-known structure designed by Picard and Queyranne [33] for (s, t) -mincut. There is an $\mathcal{O}(m)$ space directed acyclic graph (DAG) that stores and characterizes all (s, t) -mincuts of any graph using 1-*transversal* cuts of this DAG. An (s, t) -cut is said to be 1-transversal if its edge-set³ intersects any path at most once. Interestingly, we are able to achieve the $\mathcal{O}(m)$ bound on space for storing and characterizing all minimum+1 (s, t) -cuts.

An edge (u, v) is said to *contribute* to a cut C if $u \in C$ and $v \in \overline{C}$. We first establish that for any maximum (s, t) -flow f , there exists a set containing at most $n - 2$ edges, called the *anchor* edges, such that for any minimum+1 (s, t) -cut C , exactly one anchor edge contributes to C . By exploiting this result, we design the following structure for storing and characterizing all minimum+1 (s, t) -cuts.

► **Theorem 6** (Structure for minimum+1 (s, t) -cuts). *Let G be an undirected multi-graph on n vertices and m edges with a maximum (s, t) -flow f . There is an $\mathcal{O}(\min\{m, n\sqrt{\lambda}\})$ space structure, consisting of a directed acyclic graph \mathcal{D} and the set of $n - 2$ anchor edges, that stores and characterizes all (s, t) -mincuts and all minimum+1 (s, t) -cuts of G as follows.*

1. *An (s, t) -cut C is an (s, t) -mincut in G if and only if C is a 1-transversal cut in \mathcal{D} and no anchor edge corresponding to f contributes to C .*
2. *An (s, t) -cut C is a minimum+1 (s, t) -cut in G if and only if C is a 1-transversal cut in \mathcal{D} and exactly one anchor edge corresponding to f contributes.*

For undirected graphs, the best-known structure for storing and characterizing all (s, t) -mincuts is the DAG of Picard and Queyranne [33] that occupies $\mathcal{O}(\min\{m, n\sqrt{\lambda}\})$ space, which is tight as well (refer to [20] and Lemma 12 in [22]). Interestingly, not only our structure in Theorem 6 matches the bound on space with the DAG for (s, t) -mincuts [33] but also it stores and characterizes both (s, t) -mincuts and minimum+1 (s, t) -cuts.

1.3 Dual Edge Sensitivity Oracle: Breaking the Quadratic Barrier

The study of minimum+1 (s, t) -cuts often turns out to be useful in designing elegant *dual edge sensitivity oracles* [3, 15, 6], which is defined as follows.

► **Definition 7** (Dual edge sensitivity oracle). *A dual edge sensitivity oracle for minimum cuts is a compact data structure that can efficiently report a minimum cut in the graph after the insertion or failure of any given pair of query edges.*

Designing sensitivity oracles for various minimum cuts of a graph has been an emerging field of research [33, 15, 16, 3, 2]. For (s, t) -mincut in multi-graphs, the DAG structure of Picard and Queyranne [33], as shown in [3], can be used to design an $\mathcal{O}(n)$ space sensitivity oracle

³ The *edge-set* of a cut C is the set of edges with exactly one endpoint in C .

that can report an (s,t) -mincut in $\mathcal{O}(n)$ time after the failure/insertion of any single edge. It is now interesting to design a sensitivity oracle that can handle multiple failures/insertions of edges. To solve this generic problem, as argued in [3], the natural approach is to first design a dual edge sensitivity oracle for (s,t) -mincut. This approach has also been taken for various other classical graph problems, including distance and connectivity [17], traversals [32], reachability [12, 11]. Moreover, it has been observed that handling two edge failures/insertions is significantly more nontrivial than handling a single one. It also provides important insights that either expose the hardness or help in generalizing the problem for multiple failures. To extend the result of [33] from single to multiple edge failures/insertions, Baswana, Bhanja, and Pandey [3] designed the first dual edge sensitivity oracle for (s,t) -mincut occupying $\mathcal{O}(n^2)$ space in directed multi-graphs. It can report a resulting (s,t) -mincut in $\mathcal{O}(n)$ time.

Note that quadratic space data structures often pose practical challenges as n can be quite *large* for the real world networks/graphs. It thus raises the need to design sensitivity oracles for various fundamental graph problems that occupy subquadratic space [7, 35, 5]. Unfortunately, it has been shown [3, 6] that any dual edge sensitivity oracle for (s,t) -mincut in undirected multi-graphs must occupy $\Omega(n^2)$ bits of space in the worst case, irrespective of the query time. However, for undirected unweighted simple graphs, we break this quadratic barrier on the space of any dual edge sensitivity oracle while achieving nontrivial query time as follows.

► **Theorem 8** (Dual edge sensitivity oracle for (s,t) -mincut). *Let G be an undirected unweighted simple graph on n vertices and m edges. There exists a data structure occupying $\mathcal{O}(\min\{m, n^{1.5}\})$ space that can report an (s,t) -mincut C (including the contributing edges of C) in $\mathcal{O}(\min\{m, n^{1.5}\})$ time after failure/insertion of any given pair of query edges in G .*

For the existing dual edge sensitivity oracles for (s,t) -mincut [3, 6], no nontrivial preprocessing time is known till date. We establish the following almost linear preprocessing time for our dual edge sensitivity oracle stated in Theorem 8.

► **Theorem 9** (Preprocessing time). *For undirected unweighted simple graphs on n vertices and m edges, there is an algorithm that, given any maximum (s,t) -flow, can construct the dual edge sensitivity oracle in Theorem 8 in $\mathcal{O}(m)$ time.*

2 Organization of this article

This article is organized as follows. Basic preliminaries and notations are stated in Section 3. A limitation of an existing algorithm in [37] for computing second (s,t) -mincut is provided in Section 4. In Section 5, we design two algorithms for computing a second (s,t) -mincut in directed weighted graphs. For undirected multi-graphs, in Section 6, we establish close relationships between maximum (s,t) -flow and (s,t) -cuts of capacity beyond (s,t) -mincut, which are used as tools to arrive at the results in the following sections. For undirected multi-graphs, Section 7 contains the design of our linear space structure for storing and characterizing all $(\lambda + 1)$ (s,t) -cuts. In Section 8, we design the subquadratic space dual edge sensitivity oracle for (s,t) -mincut in undirected unweighted simple graphs. Finally, we conclude in Section 9. All the omitted proofs are provided in the full version.

3 Preliminaries

By integrality of maximum (s,t) -flow [19], for integer-weighted graphs, we consider any given maximum (s,t) -flow to be integral. The following notations to be used throughout the article.

- $G \cup A$ (likewise $G \setminus A$): Graph obtained after adding (likewise removing) a set of edges A in G .
- $(\lambda + \Delta)$ (s, t) -cut: An (s, t) -cut of capacity $\lambda + \Delta$ where $\Delta \geq 0$.
- (u, v) -path : A simple directed path from vertex u to vertex v .
- f denotes a maximum (s, t) -flow in graph G .
- $c(C, H)$: Capacity of a cut C in a graph H .
- A cut C *subdivides* a set of vertices X if $C \cap X \neq \emptyset$ and $\overline{C} \cap X \neq \emptyset$.
- A cut C *separates* a pair of vertices u, v if $u \in C$ and $v \in \overline{C}$ or vice-versa.
- The *edge-set* of a cut C , denoted by $E(C)$, is the set of all edges whose endpoints are separated by C .

Let f' be any (s, t) -flow in G .

- $H^{f'}$ denotes the residual graph for any graph H corresponding to an (s, t) -flow f' .
- $f'(e)$ denotes the value of flow f' assigned to an edge e .
- $f'_{out}(C)$ and $f'_{in}(C)$: For any (s, t) -cut C , $f'_{out}(C)$ (likewise $f'_{in}(C)$) is the sum of flow assigned to all edges $e = (u, v) \in E(C)$ with $u \in C, v \in \overline{C}$ (likewise $v \in C, u \in \overline{C}$).

► **Lemma 10** (Conservation of flow). *For any (s, t) -cut C , $f'_{out}(C) - f'_{in}(C) = \text{value}(f')$*

► **Definition 11** (Quotient Graph). *A graph H is said to be a quotient graph of G if H is obtained from G by contracting disjoint subsets of vertices into single nodes.*

► **Definition 12** (Quotient Path). *A path P_q is said to be a quotient path of a path P if P_q is obtained from P by contracting a set of edges in P .*

Residual graph for undirected multi-graphs. Although the residual graph is defined for directed graphs [19], for undirected graphs, we define the residual graph in the following way. Let $e = (x, y)$ be any edge in an undirected multi-graph H with an (s, t) -flow f' . There exist two edges (y, x) (likewise (x, y)) in $H^{f'}$ if e carries flow in the direction x to y (likewise y to x); otherwise, there is a pair of edges (x, y) and (y, x) in $H^{f'}$.

3.1 A DAG structure for storing and characterizing all (s, t) -mincuts

In a seminal work, Picard and Queyranne [33] designed a DAG, denoted by $\mathcal{D}_{PQ}(G)$, that stores all (s, t) -mincuts in G and characterizes them as 1-transversal cuts. We now briefly describe the construction of $\mathcal{D}_{PQ}(G)$.

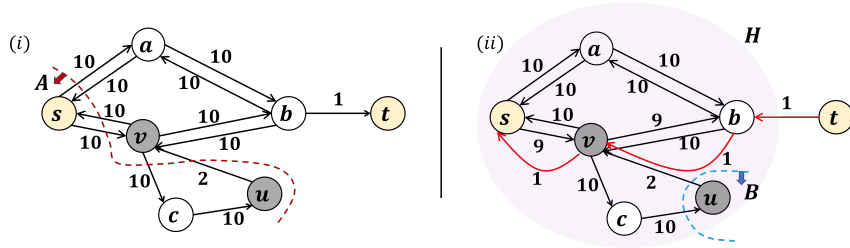
Construction of $\mathcal{D}_{PQ}(G)$. Let G' be the graph obtained by contracting each Strongly Connected Component (SCC) of G^f into a single node. Let \mathbb{T} denote the node containing t and \mathbb{S} denote the node containing s in G' . If G is an undirected graph, $\mathcal{D}_{PQ}(G)$ is the graph G' . For directed graphs, $\mathcal{D}_{PQ}(G)$ is obtained by suitably modifying G' as follows. For each node μ reachable from \mathbb{S} in G' , μ is contracted into node \mathbb{S} . Likewise, each node μ that has a path to \mathbb{T} , is contracted into the node \mathbb{T} . Given any maximum (s, t) -flow f , $\mathcal{D}_{PQ}(G)$ can be obtained in $\mathcal{O}(m)$ time and has the following property. Without causing ambiguity, we refer to (\mathbb{S}, \mathbb{T}) -cut in $\mathcal{D}_{PQ}(G)$ by (s, t) -cut.

► **Theorem 13** ([33]). *Let G be any directed weighted graph on m edges with a designated source vertex s and designated sink vertex t . There is an $\mathcal{O}(m)$ space DAG $\mathcal{D}_{PQ}(G)$ that stores and characterizes each (s, t) -mincut in G as follows. An (s, t) -cut C is an (s, t) -mincut in G if and only if C is a 1-transversal cut in $\mathcal{D}_{PQ}(G)$.*

4 Limitation of the Existing Algorithm for Second (s,t)-mincut

We state here a limitation of the existing algorithm given by Vazirani and Yannakakis [37] to compute a second (s, t) -mincut. The algorithm for computing an (s, t) -cut of k^{th} minimum capacity by [37] uses $\mathcal{O}(n^{2(k-1)})$ maximum (s, t) -flows. So, for $k = 2$, the algorithm uses $\mathcal{O}(n^2)$ maximum (s, t) -flow computations. In the same article [37], they stated the following property.

► **Lemma 14** (Lemma 3.2(1) in [37]). *Let G^f be the residual graph corresponding to a maximum (s, t) -flow f in G . Let H be an SCC in G^f and u, v be a pair of vertices in H . If the capacity of the least capacity cut separating $\{u\}$ and $\{v\}$ is k in H , then the least capacity cut separating $\{s, u\}$ and $\{v, t\}$ has capacity $\lambda + k$ in G .*



■ **Figure 1** A counter example for Lemma 3.2(1) in [37] (i) A graph G . $V \setminus \{t\}$ is the (s, t) -mincut in G with capacity 1 and $c(A)=22$. (ii) H is an SCC in G^f and $c(B, H) = 2$.

Vazirani and Yannakakis [37] used Lemma 14 to design an algorithm that computes second (s, t) -mincut using $\mathcal{O}(n)$ maximum (s, t) -flow computations (the algorithm preceding Theorem 3.3 in [37]). Unfortunately, Lemma 14 (or Lemma 3.2(1) in [37]) is not correct as shown in Figure 1(ii). H is the SCC in G^f containing vertices u and v . The least capacity cut B separating $\{u\}$ and $\{v\}$ has capacity 2 in H . By Lemma 14, the least capacity cut separating $\{s, u\}$ and $\{v, t\}$ must have capacity $2 + 1 = 3$ in G . However, as it can be verified easily using Figure 1(i), A is a least capacity cut separating $\{s, u\}$ and $\{v, t\}$ in G and its capacity is 22. Therefore, the algorithm stated above Theorem 3.3 in [37], fails to correctly output a second (s, t) -mincut.

5 Faster Algorithm for Second (s,t)-mincut

We design two efficient algorithms for computing a second (s, t) -mincut. Our first algorithm computes a second (s, t) -mincut for directed weighted graphs using $\mathcal{O}(n)$ maximum (s, t) -flow computations, which achieves the aim of Vazirani and Yannakakis [37]. This algorithm can be seen as an immediate application of the recently invented covering technique of [3]. We refer to the full version for the details of this algorithm. Our second algorithm takes a totally different approach. This approach is based on a relationship between the second (s, t) -mincuts and the global mincuts in directed weighted graphs. So, as our main result, we design an algorithm for computing a second (s, t) -mincut that uses $\tilde{\mathcal{O}}(\sqrt{n})$ maximum (s, t) -flow computations and works for directed graphs with integer edge capacities. We now provide an overview of this result.

Observe that a global mincut is not necessarily an (s, t) -cut in G . On the other hand, any second (s, t) -mincut can never be a global mincut in G . So apparently there does not seem to be any relationship between the global mincuts and the second (s, t) -mincuts of G .

Let us first work with a special case when graph G has exactly two (s, t) -mincuts – $\{s\}$ and $V \setminus \{t\}$. Suppose there exists a second (s, t) -mincut C in graph G such that C separates all the neighbors of s from all the neighbors of t . It is easy to compute a second (s, t) -mincut in this graph as follows. Compute a maximum (s, t) -flow after contracting all neighbors of s with s and all neighbors of t with t . The challenge arises when every second (s, t) -mincut has at least one contributing edge that is incident on s and/or t . We now show that the residual graph G^f plays a crucial role in overcoming this hurdle. Moreover, G^f turns out to establish the bridge between second (s, t) -mincuts and global mincuts.

In a seminal work in 1956, Ford and Fulkerson [19] established a strong duality between (s, t) -mincut and maximum (s, t) -flow, which is widely known as the *Maxflow-Mincut Theorem*. We begin by stating the following property, which is immediate from Maxflow-Mincut Theorem.

► **Fact 15.** *For any graph G with maximum (s, t) -flow f' , every (s, t) -mincut in G is an (s, t) -cut of capacity zero in $G^{f'}$.*

It follows from Fact 15 that there is a bijective mapping between the set of all (s, t) -mincuts in G and the set of global mincuts containing s and not t in G^f . However, Fact 15 does not reveal any information on how a second (s, t) -mincut in G appears in residual graph G^f . To explore the structure of second (s, t) -mincuts in G^f , using only the conservation of flow and the construction of the residual graph, we provide a generalization of Fact 15 as follows.

► **Theorem 16** (Maxflow (Min+ Δ)-cut Theorem). *Let $G = (V, E)$ be a directed weighted graph with a designated source vertex s and a designated sink vertex t . Let f be any maximum (s, t) -flow in G and G^f be the corresponding residual graph. Let C be an (s, t) -cut in G . The capacity of C in G is $(\lambda + \Delta)$ if and only if the capacity of C in G^f is Δ , where $\Delta \geq 0$.*

Proof. Let $E_{out}(A, H)$ (likewise $E_{in}(A, H)$) denote the set of outgoing edges (likewise the set of incoming edges) of a cut A in a graph H . For any edge e in G , let $r(e)$ be the residual capacity, that is, $r(e) = w(e) - f(e)$. For any edge e in graph G^f , let $w'(e)$ denote the capacity of edge e . By construction of G^f , for each edge $e = (u, v) \in E_{out}(C, G)$ with $r(e) \neq 0$, there exists a forward edge $(u, v) \in E_{out}(C, G^f)$ with $w'(u, v) = r(e)$. Similarly, for each edge $e = (u, v) \in E_{in}(C, G)$ with $f(e) \neq 0$, there exists a backward edge $(v, u) \in E_{out}(C, G^f)$ with $w'(v, u) = f(e)$. This provides us with the following equality.

$$\begin{aligned}
 \sum_{e' \in E_{out}(C, G^f)} w'(e') &= \sum_{e \in E_{out}(C, G)} r(e) + \sum_{e \in E_{in}(C, G)} f(e) \\
 &= \sum_{e \in E_{out}(C, G)} (w(e) - f(e)) + \sum_{e \in E_{in}(C, G)} f(e) \\
 &= \sum_{e \in E_{out}(C, G)} w(e) - \sum_{e \in E_{out}(C, G)} f(e) + \sum_{e \in E_{in}(C, G)} f(e) \\
 &= c(C) - (f_{out}(C) - f_{in}(C)) = c(C) - \lambda \quad \text{Using Lemma 10} \quad (1)
 \end{aligned}$$

Equation 1 completes the proof. ◀

It follows from Theorem 16 that every second (s, t) -mincut in G is a second (s, t) -mincut in G^f . Let the capacity of second (s, t) -mincut in G be $\lambda + \Delta_2$, where $\Delta_2 > 0$. Recall that our aim is to establish a relationship between second (s, t) -mincuts and global mincuts using G^f . So, by Theorem 16, we need to focus on global cuts of capacity exactly Δ_2 in G^f . However, observe that a global cut of capacity Δ_2 can never be a global mincut in G^f since $\Delta_2 > 0$. Moreover, there may exist *many* global cuts of capacity Δ_2 that cannot be a second (s, t) -mincut in G .

It is observed that a directed graph has global mincut capacity strictly greater than zero if it is an SCC. It turns out that there is exactly one nontrivial SCC, say H , in the residual graph G^f since G has exactly two trivial (s, t) -mincuts. Let us consider any global mincut A in H . Observe that A is not a second (s, t) -mincut in G^f since $s, t \in \bar{A}$. In fact, the capacity of any global cut A in H might be strictly less than the capacity of A in G^f . This is because of the existence of edges that are incident on s and/or t , which may contribute to A in G^f . Interestingly, exploiting the structure of G^f , the properties of SCC H , and Fact 15, we establish the following bijective mapping between the set of all second (s, t) -mincuts in G^f and the set of all global mincuts in H .

► **Lemma 17.** *Let C_1 be a global mincut in H and C_2 be a second (s, t) -mincut in G^f . Then,*

1. $c(C_2, G^f) = c(C_1, H)$,
2. $C_1 \cup \{s\}$ is a second (s, t) -mincut in G^f and $C_2 \setminus \{s\}$ is a global mincut in H .

Proof. Let us consider global mincut C_1 in graph H , and let $c(C_1, H) = \lambda_1$. Since H is an SCC, it is easy to show that $\lambda_1 > 0$. It follows from the construction of G^f that $C_1 \cup \{s\}$ is an (s, t) -cut in G^f . Observe that the edge-set of $C_1 \cup \{s\}$ in G^f and the edge-set of C_1 in H differ only on the set of edges, say E' , that are incident on s and t in G^f . Since f is maximum (s, t) -flow, by Fact 15, s has outdegree zero; likewise, t has indegree zero in G^f . So, every edge in E' is an incoming edge to (s, t) -cut $C_1 \cup \{s\}$ in G^f . This implies that the contributing edges of (s, t) -cut $C_1 \cup \{s\}$ in G^f are the same as the contributing edges of cut C_1 in H . Hence, we arrive at the following equation.

$$c(C_1 \cup \{s\}, G^f) = c(C_1, H) = \lambda_1 \quad (2)$$

It is given that C_2 is a second (s, t) -mincut in G^f . Let $c(C_2, G^f) = \lambda_2$, where $\lambda_2 > 0$. It follows from Fact 15 that $\{s\}$ and $V \setminus \{t\}$ are the only (s, t) -mincuts in G^f . So, any (s, t) -cut in G^f except $\{s\}$ and $V \setminus \{t\}$ has capacity at least λ_2 . Since C_1 is a cut in H , $C_1 \cup \{s\}$ is neither $\{s\}$ nor $V \setminus \{t\}$. Moreover, since $C_1 \cup \{s\}$ is an (s, t) -cut in G^f , we get $c(C_2, G^f) \leq c(C_1 \cup \{s\}, G^f)$. So, it follows from Equation 2 that $c(C_2, G^f) \leq \lambda_1$. Therefore, we arrive at the following inequality.

$$\lambda_2 \leq \lambda_1 \quad (3)$$

Observe that second (s, t) -mincut C_2 must separate a pair of vertices in H since $\{s\}$ and $V \setminus \{t\}$ are (s, t) -mincuts in G^f . Thus, the cut $C_2 \setminus \{s\}$ appears as a global cut in graph H . Thus, we can arrive at the following equation using similar arguments as in the case of Equation 2.

$$c(C_2 \setminus \{s\}, H) = c(C_2, G^f) = \lambda_2 \quad (4)$$

Since C_1 is a global mincut in H and $C_2 \setminus \{s\}$ is a global cut in H , $c(C_1, H) \leq c(C_2 \setminus \{s\}, H)$. Therefore, by Equation 4, we get the following inequality.

$$\lambda_1 \leq \lambda_2 \quad (5)$$

It follows from Equations 3 and 5 that $\lambda_2 = \lambda_1$. Evidently, $c(C_2, G^f) = c(C_1, H)$ and it completes the proof of (1). Now, by Equation 2, $c(C_1 \cup \{s\}, G^f) = \lambda_2$ and by Equation 4, $c(C_2 \setminus \{s\}, H) = \lambda_1$. Therefore, (s, t) -cut $C_1 \cup \{s\}$ is a second (s, t) -mincut in G^f and $C_2 \setminus \{s\}$ is a global mincut in H . This completes the proof of (2). ◀

The following lemma is immediate from Lemma 17 and Theorem 16.

► **Lemma 18.** *Suppose G has exactly two (s, t) -mincuts – $\{s\}$ and $V \setminus \{t\}$. There is an algorithm that, given any maximum (s, t) -flow in G , can compute a second (s, t) -mincut in G using one global mincut computation.*

It turns out that Lemma 18 does not immediately hold for graphs with one or more than two (s, t) -mincuts. For graphs with exactly one (s, t) -mincut, we suitably modify the SCC H in G^f . Finally, to extend our result for graphs having more than two (s, t) -mincuts, we partition the graph into a set of *roughly* disjoint subgraphs, each having at most two (s, t) -mincuts. This is obtained by applying a graph decomposition technique using DAG $\mathcal{D}_{PQ}(G)$ (Theorem 13), which was originally introduced in [3] for (s, t) -cuts of capacity $(\lambda + 1)$ (Section 5.4 in [3]). This leads to Theorem 2(1). The proof of Theorem 2(2) is straightforward using standard techniques.

Cen et al. [9] recently designed an efficient algorithm for computing a global mincut in directed graphs with integer edge capacities. Their algorithm uses $\tilde{O}(\sqrt{n})$ maximum (s, t) -flow computations to compute a global mincut. This result of Cen et al. [9], along with Theorem 2(1), immediately leads to Theorem 1.

6 Characterization of Minimum+1 (s, t) -cuts using Maximum flow

The widely known Maxflow-Mincut Theorem [19] provides a characterization of all (s, t) -mincuts using a maximum (s, t) -flow. However, it seems that any extension of this result might not exist for characterizing (s, t) -cuts of capacity $\lambda + 1$. This is because no equivalent (s, t) -flow is known corresponding to a $(\lambda + 1)$ (s, t) -cut, as stated in [3]. Interestingly, we show that, in fact, there exist close relationships between maximum (s, t) -flow and $(\lambda + 1)$ (s, t) -cuts as follows.

For undirected multi-graphs, we first establish the following property for $(\lambda + k)$ (s, t) -cuts, where $k \geq 0$ is an integer, based on a maximum (s, t) -flow.

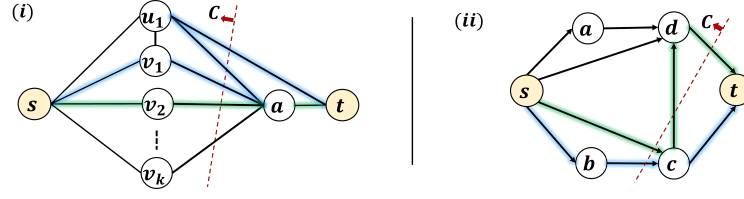
► **Lemma 19.** *Let C be a $(\lambda + k)$ (s, t) -cut in G , where $k \geq 0$ is an integer. Let f be any maximum (s, t) -flow in G and let $\mathcal{E} \subseteq E(C)$ be the set of edges such that $f(e) = 0$ for every edge $e \in \mathcal{E}$. Then, (1) $|\mathcal{E}| \leq k$ and (2) $|\mathcal{E}|$ is odd if and only if k is odd.*

Proof. Suppose C is a $(\lambda + k)$ (s, t) -cut in G . It follows from conservation of flow (Lemma 10), $f_{out}(C) \geq \lambda$, since f is a maximum (s, t) -flow and $f_{in}(C) \geq 0$. Let $f_{out}(C) = \lambda + j$, for any integer $j \geq 0$. Again by Lemma 10, $f_{in}(C) = j$. Therefore, $f_{out}(C) + f_{in}(C) = \lambda + 2j$. It follows that there are exactly $\lambda + 2j$ edges belonging to $E(C)$ that are carrying flow. Let $\mathcal{E} \subseteq E(C)$ be the set of remaining edges such that, for every edge $e \in \mathcal{E}$, $f(e) = 0$. In undirected graphs, every edge belonging to the edge-set of a cut is a contributing edge of the cut. Therefore, using $f_{out}(C) + f_{in}(C) = \lambda + 2j$, we arrive at the following equality.

$$|\mathcal{E}| = c(C) - (f_{out}(C) + f_{in}(C)) = \lambda + k - (\lambda + 2j) = k - 2j \quad (6)$$

It follows from Equation 6 that $|\mathcal{E}| \leq k$. In addition, since $2j$ is always an even number, this implies that $|\mathcal{E}|$ is odd if k is odd; otherwise $|\mathcal{E}|$ is even. ◀

By crucially exploiting Lemma 19, we now establish an interesting *flow-based characterization* of all the $(\lambda + 1)$ (s, t) -cuts, which is of independent interest.



■ **Figure 2** A colored edge represents that the edge carries flow and $\lambda = 2$. (i) Example showing that Theorem 20 cannot be generalized for $k \geq 2$ (ii) Cut C is a $(\lambda + 1)$ (s, t) -cut with $k = 1$ where each edge belonging to $E(C)$ carries flow. So, $|\mathcal{E}|$ is even although k is odd.

► **Theorem 20 (Maxflow (Min+1)-cut Theorem).** *Let G be an undirected multi-graph with a designated source s and a designated sink t . Let f be any maximum (s, t) -flow in G . Then, an (s, t) -cut C in G is a $(\lambda + 1)$ (s, t) -cut if and only if there exists exactly one edge $e \in E(C)$ such that*

1. $f(e) = 0$ and
2. for every edge $e' \in E(C) \setminus \{e\}$, $f(e') = 1$ and e' carries flow in the direction C to \bar{C} .

Proof. Suppose C is a $(\lambda + 1)$ (s, t) -cut. Let $\mathcal{E} \subseteq E(C)$ be the set of edges such that $f(e) = 0$ for each edge $e \in \mathcal{E}$. By assigning the value of $k = 1$ in Lemma 19, we have $|\mathcal{E}| \leq 1$ and $|\mathcal{E}|$ is odd. Since $|\mathcal{E}|$ is odd, therefore, $|\mathcal{E}| = 1$. Since f is a maximum (s, t) -flow, by Lemma 10, $f_{out}(C) \geq \lambda$. Moreover, we have $|E(C)| = \lambda + 1$. Therefore, for every edge $e' \in E(C) \setminus \{e\}$, $f(e') = 1$ and e' carries flow in the direction C to \bar{C} .

We now prove the converse part. Suppose C is an (s, t) -cut satisfying properties (1) and (2). It follows that there is no edge $e' \in E(C)$ such that $f(e') = 1$ and carries flow in the direction \bar{C} to C . So, $f_{in}(C) = 0$. By using Lemma 10, we have $f_{out}(C) = \lambda$. So, in $E(C)$, there are exactly λ edges that are carrying flow and exactly one edge that is carrying no flow. Therefore, $|E(C)| = \lambda + 1$. ◀

We are thankful to an anonymous reviewer for pointing out that the forward direction of Theorem 20 can also be established using the concept of edge-disjoint paths instead of using conservation of flow.

► **Remark 21.** An immediate generalization of Theorem 20 would be the following. An (s, t) -cut C is a $(\lambda + k)$ (s, t) -cut if and only if there are exactly k edges in $E(C)$ that carry zero flow and every other edge carries flow in the direction C to \bar{C} . However, Figure 2(ii) shows that for $\lambda = 2$, (s, t) -cut C has capacity $k + 2$ but there are $k - 2$ edges in $E(C)$ carrying zero flow. Hence, this generalization of Theorem 20 is not possible for any $k \geq 2$.

7 Compact Structure for All Minimum+1 (s,t)-cuts

We address the following problem for undirected multi-graphs in this section.

► **Problem 22.** *Design a compact structure for storing and characterizing all $(\lambda + 1)$ (s, t) -cuts.*

The problem of designing a compact structure for storing and characterizing all (s, t) -mincuts immediately reduces to Problem 22 by modifying the given graph as follows. Add a dummy source s' (likewise a dummy sink t') with $\lambda - 1$ edges from s' to s (likewise t to t'). Interestingly, for directed multi-graphs, Baswana, Bhanja, and Pandey [3] provide a solution to Problem 22 by essentially reducing it to the problem of designing a compact structure for storing and characterizing all (s, t) -mincuts. However, their structure occupies $\mathcal{O}(mn)$ space. For undirected multi-graphs, we present a structure for storing and characterizing all $(\lambda + 1)$ (s, t) -cuts that occupies $\mathcal{O}(\min\{m, n\sqrt{\lambda}\})$ space as follows.

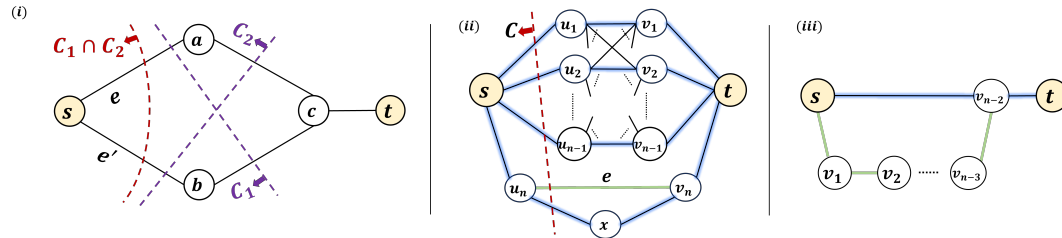
We construct a graph G' such that every $(\lambda + 1)$ (s, t) -cut of G appears as an (s, t) -mincut in G' . This will help us store and characterize all $(\lambda + 1)$ (s, t) -cuts of G using a structure that stores and characterizes all (s, t) -mincuts in G' . To achieve this objective, we pursue the following simple idea – remove one edge from every $(\lambda + 1)$ (s, t) -cut. A naive implementation of this idea might not work as follows. There may be a pair of edges e, e' contributing to the cut $C_1 \cap C_2$ defined by the intersection of two $(\lambda + 1)$ (s, t) -cuts C_1, C_2 . Even if none of the edges e, e' contribute to both C_1 and C_2 , their removal will reduce the (s, t) -mincut capacity if $c(C_1 \cap C_2)$ is λ or $\lambda + 1$ (refer to Figure 3(i)). In order to materialize the idea, we exploit Theorem 20, which motivates us to define a set of edges with respect to $(\lambda + 1)$ (s, t) -cuts in the following way.

► **Definition 23** (Anchor edge). *For any given maximum (s, t) -flow f , an edge e is said to be an anchor edge if e contributes to a $(\lambda + 1)$ (s, t) -cut and $f(e) = 0$.*

By Maxflow-Mincut Theorem, the removal of a set of edges carrying no flow in f does not reduce the capacity of (s, t) -mincut. Moreover, by Theorem 20, for every $(\lambda + 1)$ (s, t) -cut C in G , exactly one anchor edge contributes to C . Hence, every $(\lambda + 1)$ (s, t) -cut, as well as (s, t) -mincut in G , appears as an (s, t) -mincut of capacity λ in $G \setminus \mathcal{A}$. It is also easy to observe that there may exist (s, t) -cuts with capacity more than $\lambda + 1$ appearing as (s, t) -mincuts in $G \setminus \mathcal{A}$. However, using anchor edges \mathcal{A} , we can distinguish the $(\lambda + 1)$ (s, t) -cuts as follows.

► **Lemma 24.** *An (s, t) -cut C is a $(\lambda + 1)$ (s, t) -cut in G if and only if C is an (s, t) -mincut in $G \setminus \mathcal{A}$ and exactly one edge from \mathcal{A} contributes to C .*

By Lemma 24, our compact structure consists of set of edges \mathcal{A} and a structure that stores and characterizes all (s, t) -mincuts in $G \setminus \mathcal{A}$. It follows that the space occupied by our structure exceeds that of any structure for storing and characterizing all (s, t) -mincuts only by the size of \mathcal{A} . Therefore, we need to show that the set \mathcal{A} is *small*.



■ **Figure 3** A blue edge carries flow and a green edge is an anchor edge. (i) $C_1 \cap C_2$ has capacity less than that of (s, t) -mincut in $G \setminus \{e, e'\}$. (ii) A graph with a given maximum (s, t) -flow satisfying $|\text{NoFlow}| = \Omega(n^2)$ but exactly one anchor edge e . (iii) Graph with exactly $(n - 2)$ anchor edges.

7.1 Bounding the Cardinality of The Set of Anchor Edges

We now provide a tight bound on the cardinality of the set of all anchor edges \mathcal{A} . We present an efficient algorithm that, exploiting the properties of anchor edges, computes a set of edges \mathcal{F} such that the following two properties hold.

1. The set of all anchor edges is a subset of \mathcal{F} .
2. The number of edges belonging to \mathcal{F} is at most $n - 2$.

By Definition 23, every anchor edge belongs to the set, say NoFlow , of all edges carrying zero flow in f . However, there exist graphs H such that, for any maximum (s, t) -flow in H , the cardinality of set NoFlow can be $\Omega(n^2)$, yet the number of anchor edges is only $\mathcal{O}(1)$ (refer to Figure 3(ii)). So, NoFlow provides a *loose* upper bound on the number of anchor

edges. Naturally, the question arises whether it is possible to eliminate a large number of edges from NOFLOW while still keeping the set of all anchor edges intact. We answer this question in the affirmative by exploiting the following lemma.

► **Lemma 25.** *Let \mathbb{H} be a cycle formed using a subset of edges in NOFLOW. Then, no edge in \mathbb{H} can be an anchor edge.*

Proof. Consider any edge $e \in \mathbb{H}$ and C be any (s, t) -cut in G such that $e \in E(C)$. Since C is a cut, C must intersect cycle \mathbb{H} at least twice. Hence, C contains at least two edges that carry no flow. Therefore, it follows from Theorem 20 that C cannot be a $(\lambda + 1)$ (s, t) -cut. Hence, by Definition 23, e cannot be an anchor edge. ◀

We now use Lemma 25 to construct a spanning forest $G_{\mathcal{F}}$ to provide an upper bound on the cardinality of set \mathcal{A} .

Construction of Spanning Forest $G_{\mathcal{F}} = (V, \mathcal{F})$. The vertex set of $G_{\mathcal{F}}$ is the same as G . Initially, there is no edge in $G_{\mathcal{F}}$. We construct graph $G_{\mathcal{F}}$ incrementally by executing the following step for each edge e in NOFLOW. If a cycle is formed in $G_{\mathcal{F}} \cup \{e\}$, then by Lemma 25, e cannot be an anchor edge. Hence, we ignore edge e . Otherwise, if there is no cycle in $G_{\mathcal{F}} \cup \{e\}$, then insert edge e to $G_{\mathcal{F}}$.

It follows from the construction of $G_{\mathcal{F}}$ that $\mathcal{A} \subseteq \mathcal{F}$. Moreover, $|\mathcal{F}|$ is at most $n - 1$ since $G_{\mathcal{F}}$ is a spanning forest. We now show a tight bound on $|\mathcal{A}|$, as well as $|\mathcal{F}|$, in the following lemma.

► **Lemma 26.** *Set \mathcal{A} , as well as set \mathcal{F} , contains at most $(n - 2)$ edges.*

Proof. Let us assume to the contrary that \mathcal{F} contains exactly $n - 1$ edges. It follows that $G_{\mathcal{F}}$ is a spanning tree. Hence, \mathcal{F} contains at least one edge from the edge-set of every (s, t) -cut in G . It implies that there exists an (s, t) -mincut C in G such that \mathcal{F} contains at least one edge from the edge-set of C . By Maxflow-Mincut Theorem, for every edge $e \in E(C)$, $f(e) = 1$. Thus, \mathcal{F} contains an edge e such that $f(e) = 1$, which is a contradiction since $\mathcal{F} \subseteq \text{NOFLOW}$. Since $\mathcal{A} \subseteq \mathcal{F}$, it follows that the cardinality of \mathcal{A} is at most $n - 2$. ◀

We show that there also exist graphs where \mathcal{A} contains exactly $n - 2$ edges (refer to Figure 3(iii)). Therefore, the bound on the set \mathcal{A} given in Lemma 26 is tight.

We also show that set \mathcal{A} can be obtained in $\mathcal{O}(m)$ time. Finally, using the best-known structure \mathcal{D}_{PQ} for storing and characterizing all (s, t) -mincuts [33], we show that $\mathcal{D}_{PQ}(G \setminus \mathcal{A})$ occupies $\mathcal{O}(\min\{m, n\sqrt{\lambda}\})$ space, which leads to Theorem 6.

8 Dual Edge Sensitivity Oracle: Breaking the Quadratic Barrier

In this section, for undirected unweighted simple graphs, we design a subquadratic space data structure that can efficiently answer the query: report an (s, t) -mincut after the failure/insertion of any pair of edges. We assume G to be an undirected unweighted simple graph in this section. We provide an overview for handling the failure of edges in G . Note that handling the insertion of a pair of edges is along a similar line and is provided in the full version. Henceforth, let $e_1 = (x_1, y_1)$ and $e_2 = (x_2, y_2)$ be the two failed edges in G .

There is a folklore result that the residual graph G^f acts as an $\mathcal{O}(m)$ space dual edge sensitivity oracle for (s, t) -mincut that achieves $\mathcal{O}(m)$ query time. The query algorithm is derived from the *augmenting path* based algorithm for computing maximum (s, t) -flow by Ford and Fulkerson [19] (briefly explained as a warm-up below). However, it is known that

the residual graph occupies quadratic space if $m = \Omega(n^2)$. To design a subquadratic space dual edge sensitivity oracle for undirected unweighted simple graphs, instead of the residual graph G^f , we work with our compact structure, consisting of \mathcal{D} and the set \mathcal{A} of anchor edges, from Theorem 6. Recall that \mathcal{D} is the graph $\mathcal{D}_{PQ}(G \setminus \mathcal{A})$, which is just a quotient graph of G^f . Hence, $\mathcal{D} \cup \mathcal{A}$ may fail to preserve the complete information of every path in G^f . However, we show that $\mathcal{D} \cup \mathcal{A}$ is still sufficient to answer dual edge failure queries using the same algorithm used for the folklore result using residual graph G^f .

Warm-up with residual graph. Observe that if none of the failed edges e_1, e_2 carry flow in maximum (s, t) -flow f , then the capacity of (s, t) -mincut remains unchanged in $G \setminus \{e_1, e_2\}$. Henceforth, we assume that edge e_1 always carries flow in the direction from x_1 to y_1 . It follows from the construction of residual graph that there must exist a (t, s) -path P in G^f containing edge (y_1, x_1) . We first reduce the flow in G using P in G^f , and then remove the edges (x_1, y_1) and (y_1, x_1) from G^f . Finally, using the concept of *augmenting paths* [19] in residual graph, we can verify in $\mathcal{O}(m)$ time whether the value of maximum (s, t) -flow becomes $\lambda - 1$ or remains λ in $G \setminus \{e_1\}$. In the residual graph corresponding to the obtained maximum (s, t) -flow in $G \setminus \{e_1\}$, repeat the same procedure for edge e_2 to verify whether failure of edge e_2 reduces the capacity of (s, t) -mincut in $G \setminus \{e_1\}$. This helps in reporting an (s, t) -mincut in the graph $G \setminus \{e_1, e_2\}$ in $\mathcal{O}(m)$ time.

Our solution using $\mathcal{D} \cup \mathcal{A}$. We now use the structure $\mathcal{D} \cup \mathcal{A}$ from Theorem 6 as a subquadratic space dual edge sensitivity oracle. $\mathcal{D}_{PQ}(G)$ (Theorem 13) can be used to design a single edge sensitivity oracle that occupies $\mathcal{O}(n)$ space [33, 3]. As observed by Baswana, Bhanja, and Pandey [3], $\mathcal{D}_{PQ}(G)$ can also handle dual edge failures for the special case when both failed edges contribute to only (s, t) -mincuts, using its reachability information. However, for handling any dual edge failures, the main difficulty arises when endpoints of both edges are mapped to the same node in $\mathcal{D}_{PQ}(G)$ [3]. Our structure $\mathcal{D} \cup \mathcal{A}$ addresses this difficulty seamlessly by first ensuring the following condition. The capacity of (s, t) -mincut changes only if the endpoints of at least one failed edge appear in different nodes of \mathcal{D} . This is achieved using the following property of \mathcal{D} .

► **Lemma 27.** *Any (s, t) -cut separating a pair of vertices mapped to the same node in \mathcal{D} must have capacity at least $\lambda + 2$.*

Henceforth, we assume without loss of generality that endpoints of edge e_1 appear in different nodes in \mathcal{D} . Let us first handle the failure of edge e_1 . It turns out that $\mathcal{D} \cup \mathcal{A}$ can easily report an (s, t) -mincut in $G \setminus \{e_1\}$. This exploits the following mapping of paths between G^f and $\mathcal{D} \cup \mathcal{A}$.

► **Lemma 28.** *Let u, v be any pair of vertices mapped to different nodes μ and ν in \mathcal{D} . There exists an (u, v) -path P in G^f if and only if there exists a (μ, ν) -path P_q in $\mathcal{D} \cup \mathcal{A}$. Moreover, path P_q is a quotient path of P .*

We establish Lemma 28 by using the fact that graph $\mathcal{D} \cup \mathcal{A}$ is a quotient graph of G^f by construction. Let D_1 be the graph obtained from $\mathcal{D} \cup \mathcal{A}$ after applying the query algorithm (described above) on $\mathcal{D} \cup \mathcal{A}$ for the failure of edge e_1 . By following the mapping of paths in Lemma 28, let f_1 be the corresponding maximum (s, t) -flow in $G \setminus \{e_1\}$ and G^{f_1} denote the corresponding residual graph. On a high level, our technical contribution lies in showing that even after handling failure of e_1 , D_1 still preserves enough information about augmenting paths in G^{f_1} that facilitates the handling of the failure of edge e_2 in $G \setminus \{e_1\}$. We now provide the overview.

To handle the failure of edge e_2 , the obtained graph D_1 must satisfy the following property, which actually holds between graphs $\mathcal{D} \cup \mathcal{A}$ and G^f (refer to Lemma 27).

► **Lemma 29.** *D_1 is a quotient graph of G^{f_1} and for every path P in D_1 , there is a path P_1 in G^{f_1} such that P is a quotient path of P_1 .*

In order to establish Lemma 29, the challenging case appears when the (s, t) -mincut remains unchanged after the failure of e_1 . In this case, let us first observe the change in the residual graph after reducing the flow that was passing through edge e_1 . In the resulting residual graph, the query algorithm finds a path P from s to t , and flips the direction of every edge belonging to it. A similar update is executed by the query algorithm using a path P_1 in graph $\mathcal{D} \cup \mathcal{A}$, where P_1 is a quotient path of P . This could lead to the following scenario for the resulting graph D_1 . There is a path P_{uv} in D_1 , but there is no path P_{uv}^r in G^{f_1} such that P_{uv} is a quotient path of P_{uv}^r . In other words, Lemma 29 may fail to hold. So, we might report an incorrect (s, t) -mincut in $G \setminus \{e_1, e_2\}$. Interestingly, exploiting the SCC structure of G^{f_1} , the structure of \mathcal{D} , and Lemma 27, we ensure that such a scenario never occurs. This allows us to handle the failure of e_2 using D_1 exactly in the same way as handling the failure of e_1 using $\mathcal{D} \cup \mathcal{A}$. This leads to Theorem 8.

9 Conclusion

In this article, we have mainly solved the following two problems, achieving significant improvements over the existing bounds.

Firstly, we have addressed the problem of designing an efficient algorithm for computing a second (s, t) -mincut in directed integer-weighted graphs. Our algorithm achieves a running time that is $\tilde{\Omega}(n\sqrt{n})$ times faster than the existing best-known algorithm [37]. Most importantly, to arrive at the solution, a close relationship between global mincut and second (s, t) -mincut is exposed for the first time here. We believe that this connection between global cuts and (s, t) -cuts (via residual graph) should be quite useful for addressing future research problems, including the problem of designing an algorithm for computing an (s, t) -cut of k^{th} minimum capacity, for any $k > 2$, that breaks the existing bound [37].

Finally, we have addressed the problem of designing a dual edge sensitivity oracle for (s, t) -mincut in undirected unweighted simple graphs. This problem is shown to be closely related to the study of (s, t) -cuts of capacity minimum+1, which is a special case of the second (s, t) -mincut in unweighted graphs. Our dual edge sensitivity oracle breaks the recently established quadratic lower bound on space in multi-graphs [6] by a factor of $\Omega(\sqrt{n})$. The foundation of this result lies in our newly designed structure for storing and characterizing all minimum+1 (s, t) -cuts, which also improves the existing best-known structure [3] by $\Omega(n)$ factor. Now, it is interesting to see whether the query time of our dual edge sensitivity oracle can be improved and whether our approach can help us in designing a sensitivity oracle for handling more than two edge insertions/failures.

References

- 1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.
- 2 Surender Baswana and Koustav Bhanja. Vital edges for (s, t) -mincut: Efficient algorithms, compact structures, & optimal sensitivity oracles. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.17.

- 3 Surender Baswana, Koustav Bhanja, and Abhyuday Pandey. Minimum+ 1 (s, t)-cuts and dual-edge sensitivity oracle. *ACM Transactions on Algorithms*, 19(4):1–41, 2023. doi:10.1145/3623271.
- 4 András A. Benczúr. A representation of cuts within $6/5$ times the edge connectivity with applications. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 92–102. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492466.
- 5 Koustav Bhanja. Optimal sensitivity oracle for steiner mincut. In *35th International Symposium on Algorithms and Computation (ISAAC 2024)*, pages 10:1–10:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ISAAC.2024.10.
- 6 Koustav Bhanja. Minimum+1 steiner cut and dual edge sensitivity oracle: Bridging gap between global and (s, t)-cut. In *52nd International Colloquium on Automata, Languages, and Programming, ICALP 2025, July 8-11, 2025, Aarhus, Denmark*, volume 334 of *LIPICs*, pages 27:1–27:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPICS.ICALP.2025.27.
- 7 Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, Simon Krogmann, and Martin Schirneck. Approximate distance sensitivity oracles in subquadratic space. *TheoretCS*, 3, 2024. doi:10.46298/THEORETICS.24.15.
- 8 PM Camerini, L Fratta, and F Maffioli. The k shortest spanning trees of a graph. *Int. Rep*, pages 73–10, 1974.
- 9 Ruoxu Cen, Jason Li, Danupon Nanongkai, Debmalya Panigrahi, Thatchaphol Saranurak, and Kent Quanrud. Minimum cuts in directed graphs via partial sparsification. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1147–1158. IEEE, 2021. doi:10.1109/FOCS52979.2021.00113.
- 10 Ruoxu Cen, Jason Li, and Debmalya Panigrahi. Augmenting edge connectivity via isolating cuts. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3237–3252. SIAM, 2022. doi:10.1137/1.9781611977073.127.
- 11 Diptarka Chakraborty, Kushagra Chatterjee, and Keerti Choudhary. Pairwise reachability oracles and preservers under failures. In *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPICs*, pages 35:1–35:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.ICALP.2022.35.
- 12 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPICs*, pages 130:1–130:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.ICALP.2016.130.
- 13 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
- 14 Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts of a graph. *Studies in discrete optimization*, pages 290–306, 1976.
- 15 Yefim Dinitz and Zeev Nutov. A 2-level cactus model for the system of minimum and minimum+ 1 edge-cuts in a graph and its incremental maintenance. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 509–518, 1995. doi:10.1145/225058.225268.
- 16 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.
- 17 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 506–515. SIAM, 2009. doi:10.1137/1.9781611973068.56.

- 18 David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1998. doi:10.1137/S0097539795290477.
- 19 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.
- 20 Zvi Galil and Xiangdong Yu. Short length versions of menger’s theorem. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC ’95, pages 499–508, New York, NY, USA, 1995. Association for Computing Machinery. doi:10.1145/225058.225267.
- 21 Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.
- 22 Gramoz Goranci and Monika Henzinger. Efficient data structures for incremental exact and approximate maximum flow. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10–14, 2023, Paderborn, Germany*, volume 261 of *LIPICs*, pages 69:1–69:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ICALP.2023.69.
- 23 Gramoz Goranci, Monika Henzinger, and Mikkel Thorup. Incremental exact min-cut in polylogarithmic amortized update time. *ACM Trans. Algorithms*, 14(2):17:1–17:21, 2018. doi:10.1145/3174803.
- 24 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’93, pages 21–30, USA, 1993. Society for Industrial and Applied Mathematics. URL: <http://dl.acm.org/citation.cfm?id=313559.313605>.
- 25 David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996. doi:10.1145/234533.234534.
- 26 N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding k minimum spanning trees. *SIAM Journal on Computing*, 10(2):247–255, 1981. doi:10.1137/0210017.
- 27 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019. doi:10.1145/3274663.
- 28 Jason Li and Debmalaya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16–19, 2020*, pages 85–92. IEEE, 2020. doi:10.1109/FOCS46700.2020.00017.
- 29 Ernst W Mayr and C Greg Plaxton. On the spanning trees of weighted graphs. *Combinatorica*, 12(4):433–447, 1992. doi:10.1007/BF01305236.
- 30 Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.
- 31 Dalit Naor, Dan Gusfield, and Charles Martel. A fast algorithm for optimally increasing the edge connectivity. *SIAM Journal on Computing*, 26(4):1139–1165, 1997. doi:10.1137/S0097539792234226.
- 32 Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 481–490. ACM, 2015. doi:10.1145/2767386.2767408.
- 33 Jean-Claude Picard and Maurice Queyranne. On the structure of all minimum cuts in a network and applications. In *Rayward-Smith V.J. (eds) Combinatorial Optimization II. Mathematical Programming Studies*, 13(1):8–16, 1980. doi:10.1007/BFb0120902.
- 34 Liam Roditty. On the k shortest simple paths problem in weighted directed graphs. *SIAM Journal on Computing*, 39(6):2363–2376, 2010. doi:10.1137/080730950.
- 35 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.
- 36 Jan Van Den Brand, Li Chen, Richard Peng, Rasmus Kyng, Yang P Liu, 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. IEEE, 2023. doi:10.1109/FOCS57990.2023.00037.

- 37 Vijay V Vazirani and Mihalis Yannakakis. Suboptimal cuts: Their enumeration, weight and number. In *International Colloquium on Automata, Languages, and Programming*, pages 366–377. Springer, 1992. doi:10.1007/3-540-55719-9_88.
- 38 Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.