

Vital Edges for (s,t) -Mincut: Efficient Algorithms, Compact Structures, & Optimal Sensitivity Oracles

Surender Baswana   

Department of Computer Science & Engineering, IIT Kanpur, India

Koustav Bhanja   

Department of Computer Science & Engineering, IIT Kanpur, India

Abstract

Let G be a directed weighted graph on n vertices and m edges with designated source and sink vertices s and t . An edge in G is vital if its removal reduces the capacity of (s,t) -mincut. Since the seminal work of Ford and Fulkerson [CJM 1956], a long line of work has been done on computing the *most vital edge* and *all vital edges* of G . However, even after 60 years, the existing results are for either undirected or unweighted graphs. We present the following result for *directed weighted graphs* that also solves an open problem by Ausiello, Franciosa, Lari, and Ribichini [NETWORKS 2019].

- 1. Algorithmic Results:** There is an algorithm that computes all vital edges as well as the most vital edge of G using $\mathcal{O}(n)$ maximum (s,t) -flow computations.

Vital edges play a crucial role in the design of *sensitivity oracle* for (s,t) -mincut – a compact data structure for reporting (s,t) -mincut after insertion/failure of any edge. For directed graphs, the only existing sensitivity oracle is for unweighted graphs by Picard and Queyranne [MPS 1982]. We present the first and optimal sensitivity oracle for *directed weighted graphs* as follows.

- 2. Sensitivity Oracles:**

- (a) There is an optimal $\mathcal{O}(n^2)$ space data structure that can report an (s,t) -mincut C in $\mathcal{O}(|C|)$ time after the failure/insertion of any edge.
- (b) There is an $\mathcal{O}(n)$ space data structure that can report the capacity of (s,t) -mincut after failure or insertion of any edge e in $\mathcal{O}(1)$ time if the capacity of edge e is known.

A *mincut for a vital edge e* is an (s,t) -cut of the least capacity in which edge e is outgoing. For unweighted graphs, in a classical work, Picard and Queyranne [MPS 1982] designed an $\mathcal{O}(m)$ space directed acyclic graph (DAG) that stores and characterizes all mincuts for all vital edges. Conversely, there is a set containing at most $n - 1$ (s,t) -cuts such that at least one mincut for every vital edge belongs to the set. We generalize these results for *directed weighted graphs* as follows.

- 3. Structural & Combinatorial Results:**

- (a) There is a set \mathcal{M} containing at most $n - 1$ (s,t) -cuts such that at least one mincut for every vital edge belongs to the set. This bound is tight as well. We also show that set \mathcal{M} can be computed using $\mathcal{O}(n)$ maximum (s,t) -flow computations.
- (b) We design two compact structures for storing and characterizing all mincuts for all vital edges – (i) an $\mathcal{O}(m)$ space DAG for *partial* and (ii) an $\mathcal{O}(mn)$ space structure for *complete* characterization.

To arrive at our results, we develop new techniques, especially a generalization of *maxflow-mincut* Theorem by Ford and Fulkerson [CJM 1956], which might be of independent interest.


2012 ACM Subject Classification Theory of computation \rightarrow Network flows; Theory of computation \rightarrow Data structures design and analysis; Mathematics of computing \rightarrow Graph algorithms

Keywords and phrases maxflow, vital edges, graph algorithms, structures, st-cuts, sensitivity oracle

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.17

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version:* <https://arxiv.org/abs/2310.12096>

 © Surender Baswana and Koustav Bhanja;
licensed under Creative Commons License CC-BY 4.0
51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 17; pp. 17:1–17:20

 Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



Funding The research work of Surender Baswana was partially supported by Tapas Mishra Memorial Chair at Indian Institute of Technology Kanpur.

Acknowledgements We thank Keerti Choudhary for her valuable feedback on this article.

1 Introduction

For any graph problem, there are edges whose removal affects the solution of the given problem. These edges are called *vital* edges for the problem. There has been extensive research on the vital edges for various fundamental problems – shortest paths/distance [32, 34, 23, 28, 19], minimum spanning trees [15, 26, 33], strongly connected components (SCC) [16]. The concept of vital edge for (s,t) -mincut has existed ever since the seminal work of Ford and Fulkerson [14] on maximum (s,t) -flow. In this article, for directed weighted graphs, we present the following two main results – (1) an efficient *algorithm* for computing all vital edges, and (2) optimal *sensitivity oracles* for (s,t) -mincut. The algorithm in (1) is the first nontrivial algorithm for computing all vital edges in directed weighted graphs, which also answers an open question in [3]. Our optimal sensitivity oracle in (2) is the first sensitivity oracle for directed weighted graphs in the area of minimum cuts. In order to arrive at these results, we present interesting *structural* & optimal *combinatorial* results on mincuts for vital edges. These results provide a generalization of the classical work of Picard and Queyranne [30] and the recent work of Baswana, Bhanja, and Pandey [4].

Let $G = (V, E)$ be a directed graph on $n = |V|$ vertices and $m = |E|$ edges. Each edge $e \in E$ has a capacity, denoted by $w(e)$, which is a positive real number. Let s be a designated source vertex and t be a designated sink vertex in G .

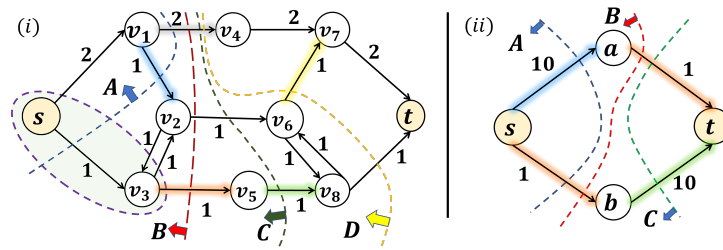
A set $C \subset V$ is said to be a cut if $C \neq \emptyset$. The outgoing edges from C are called *contributing* edges of C . The *capacity* of cut C , denoted by $c(C)$, is defined as the sum of the capacities of all contributing edges of C . A cut C is said to be an (s,t) -cut if $s \in C$ and $t \in \bar{C} = V \setminus C$. An (s,t) -cut C with the least capacity is called an (s,t) -*mincut*. We denote the capacity of (s,t) -mincut by f^* . The (s,t) -mincut of a graph is a fundamental concept in graph theory and is used to design efficient algorithms for numerous real-world problems [1]. Now we formally define the vital edges for (s,t) -mincuts.

► **Definition 1** (vital edge). *An edge $e \in E$ is said to be a vital edge if the removal of e decreases the capacity of (s,t) -mincut in G . E_{vit} denotes the set of all vital edges in G .*

Observe that each edge that contributes to an (s,t) -mincut is definitely a vital edge. However, a vital edge might not necessarily contribute to any (s,t) -mincut (e.g., edge (v_1, v_4) in Figure 1)(i). An edge that is not vital is called a *nonvital* edge. At first glance, it may appear that we may remove all nonvital edges from the graph without affecting the (s,t) -mincut. But it is not true, as stated in the following note.

► **Note 2.** Although the removal of any single nonvital edge does not decrease the capacity of (s,t) -mincut, the removal of a set of nonvital edges might lead to the reduction in the capacity of (s,t) -mincut (e.g., edges (v_2, v_6) and (v_3, v_5) in Figure 1)(i)).

The design of efficient algorithms for various problems [31, 35, 27, 29] related to vital edges started just a few years after the seminal work of Ford and Fulkerson [14]. The *vitality* of an edge e is the reduction in the capacity of (s,t) -mincut after the removal of edge e . Observe that one maximum (s,t) -flow computation is sufficient for computing the vitality of any edge. So, for computing all vital edges and their vitality, there is a trivial algorithm that requires $\mathcal{O}(m)$ maximum (s,t) -flow computations. An edge having the maximum vitality is



■ **Figure 1** (i) Each mincut for vital edge (v_1, v_4) contains a nonvital edge (shown in the same color). (ii) A mincut cover fails to include mincut B for edges (a, t) and (s, b) if property \mathcal{P} is not ensured in the construction. An edge and the mincut for the edge are shown in the same color.

said to be the *most vital edge*. Aneja, Chandrasekaran, and Nair [2] designed an algorithm that performs $\mathcal{O}(n)$ maximum (s, t) -flow computations to compute the most vital edge in an undirected graph. Ausiello, Franciosa, Lari, and Ribichini [3] showed that $\mathcal{O}(n)$ maximum (s, t) -flow computations are sufficient even for computing all vital edges and their vitality in an undirected graph. Unfortunately, even after 60 years, the following question has remained unanswered, which is also posed as an open problem in [3].

► **Question 1.** For directed weighted graph G , does there exist an algorithm that can compute all vital edges along with their vitality using $\mathcal{O}(n)$ maximum (s, t) -flow computations?

We can generalize the notion of (s, t) -mincut to “mincut for an edge” as follows.

► **Definition 3** (mincut for an edge). Let e be a contributing edge of an (s, t) -cut C . C is a mincut for edge e if $c(C) \leq c(C')$ for each (s, t) -cut C' in which edge e appears as a contributing edge.

Not only the study of mincuts for vital edges is important from a graph theoretic perspective but it also plays a crucial role in designing sensitivity oracle for (s, t) -mincuts – a compact data structure that efficiently reports an (s, t) -mincut after the failure/insertion of any edge.

The minimum cuts in a graph can be quite large in number – $\Omega(n^2)$ global mincuts [11], $\Omega(2^n)$ (s, t) -mincuts [30]. Interestingly, compact structures have been invented that compactly store and *characterize* various types of minimum cuts and cuts of capacity near minimum [13, 12, 30, 11, 4]. A compact structure G' is said to characterize a set \mathcal{C} of cuts using a property \mathcal{P} if the following holds. A cut C belongs to \mathcal{C} if and only if cut C satisfies property \mathcal{P} in graph G' . Specifically for (s, t) -mincuts in any directed weighted graph G , Picard and Queyranne [30] showed that there exists a directed acyclic graph (DAG), denoted by $\mathcal{D}_{PQ}(G)$, occupying $\mathcal{O}(m)$ space that compactly stores all (s, t) -mincuts. In addition, it provides the following characterization for each (s, t) -mincut.

An (s, t) -cut C is an (s, t) -mincut in G if and only if C is a 1-transversal cut in $\mathcal{D}_{PQ}(G)$. An (s, t) -cut is said to be 1-transversal if its edges intersect any simple path at most *once*; e.g., cut A is 1-transversal but cut $\{s, v_3\}$ is not 1-transversal in Figure 1(i). The 1-transversality property also plays a crucial role in designing sensitivity oracles for (s, t) -mincuts in unweighted graphs [30, 4]. Observe that $\mathcal{D}_{PQ}(G)$ stores all mincuts for all edges that contribute to (s, t) -mincuts. However, edges contributing to (s, t) -mincuts may constitute a small subset of the set of all vital edges. Therefore, $\mathcal{D}_{PQ}(G)$ may fail to preserve all mincuts for all vital edges. This raises the following question.

► **Question 2.** Does there exist a compact graph structure that stores and characterizes all mincuts for all vital edges in directed weighted graph G ?

Constructing the smallest set storing a minimum cut for every edge or every pair of vertices has been addressed extensively [22, 11, 9, 18, 20, 21] since the remarkable work of Gomory and Hu [18]. The smallest set of (s, t) -cuts that has at least one mincut for every edge is called a *mincut cover*. For directed unweighted graphs, Baswana, Bhanja, and Pandey [4], exploiting the DAG structure in [30], showed that there is a mincut cover of cardinality at most $n - 1$ for all vital edges. For undirected weighted graphs, it is shown in [3] that there is a mincut cover of cardinality at most $n - 1$ for all edges, and hence for all vital edges. Unfortunately, for directed weighted graphs, we establish that it is not possible to have a mincut cover of cardinality $o(n^2)$ for all edges (Theorem 37). Therefore, the following question naturally arises.

► **Question 3.** *Does there exist a mincut cover of cardinality $o(n^2)$ for all vital edges in directed weighted graph G ?*

Design of sensitivity oracles have been studied quite extensively for various fundamental problems in both unweighted and weighted graphs – shortest paths/distances [8, 19], reachability [24, 10], strongly connected components [5, 16], all-pairs mincuts [6]. In weighted graphs, the concept of failures/insertions of edges is, interestingly, more generic. Here, the aim is to report the solution to the given problem given that the capacities of a *small* set of edges are decreased/increased by an amount $\Delta > 0$.

For (s, t) -mincuts in directed graphs, the existing sensitivity oracles are only for unweighted graphs and completely based on DAG $\mathcal{D}_{PQ}(G)$ [30], which dates back to 1982. Firstly, there is an $\mathcal{O}(m)$ space sensitivity oracle given in [30]. After the failure/insertion of an edge, the oracle takes $\mathcal{O}(1)$ time for reporting the capacity and $\mathcal{O}(m)$ time for reporting the corresponding (s, t) -mincut. Assuming the edge of the query exists in the graph, an $\mathcal{O}(n)$ space data structure can be designed [4]. This data structure can report the capacity of (s, t) -mincut in $\mathcal{O}(1)$ time and an (s, t) -mincut C in $\mathcal{O}(|C|)$ time after the failure/insertion of an edge. For various problems related to sensitivity analysis, it is also important to efficiently report a compact structure that stores and characterizes *all* (s, t) -mincuts after the failure of any edge, as shown in [30]. For unweighted graphs, DAG \mathcal{D}_{PQ} for the resulting graph can be reported in $\mathcal{O}(m)$ time [30]. All these results crucially exploit the property that, in unweighted graphs, a mincut for a vital edge is also an (s, t) -mincut; hence, every contributing edge is also vital. However, for weighted graphs, this property no longer holds. In fact, nonvital edges may contribute to all mincuts for a vital edge (e.g., all mincuts $\{A, B, C, D\}$ for edge (v_1, v_4) in Figure 1(i)). So, the following question arises.

► **Question 4.** *For directed weighted graph G , does there exist*

1. *an $\mathcal{O}(n)$ space data structure that can report the capacity of (s, t) -mincut in $\mathcal{O}(1)$ time,*
2. *a compact data structure that can report an (s, t) -mincut C in $\mathcal{O}(|C|)$ time, and*
3. *a compact data structure that can report DAG \mathcal{D}_{PQ} in $\mathcal{O}(m)$ time for the resulting graph after increasing/decreasing the capacity of any given edge?*

► **Note 4.** It is a simple exercise to design an $\mathcal{O}(n^2)$ space data structure and an $\mathcal{O}(n)$ space data structure that, after increasing the capacity of any edge, can report an (s, t) -mincut C in $\mathcal{O}(|C|)$ time and its capacity in $\mathcal{O}(1)$ time, respectively. Henceforth, while addressing sensitivity oracles, we focus only on handling the decrease in the capacity of an edge.

1.1 Our contribution and the organization of the results

We provide an affirmative answer to all the four questions raised above. We present basic notations and terminologies in the following section. Thereafter, we present a generalization of *maxflow-mincut* Theorem by Ford and Fulkerson [14] in Section 3. This property, which

might be of independent interest, plays a key role in establishing various results in this article. Mincut cover for all vital edges and sensitivity oracles for (s, t) -mincut are presented in Section 4 and 5 respectively. Algorithm for computing all vital edges is presented in Section 6. Compact structures for storing and characterizing all mincuts for all vital edges are described in Section 7. We present various lower bounds in Section 8.

2 Preliminaries

For any directed weighted graph H with a designated source vertex s and a designated sink vertex t , we define the following notations to be used throughout the article.

- $H \setminus \{e\}$: Graph after the failure of an edge e in H .
- $\text{value}(f, H)$: value of an (s, t) -flow f in graph H . We denote $\text{value}(f, G)$ by f^* .
- $f(e, H)$: value of (s, t) -flow f along edge e in H .
- $f_{in}(C, H)$: For any (s, t) -flow f and an (s, t) -cut C in H , $f_{in}(C, H)$ is the sum of the flow through all edges that leave \bar{C} and enter C .
- $f_{out}(C, H)$: For any (s, t) -flow f and an (s, t) -cut C in H , $f_{out}(C, H)$ is the sum of the flow through all edges that leave C and enter \bar{C} .

Without causing any ambiguity, we use $f_{in}(C)$ and $f_{out}(C)$ to denote $f_{in}(C, G)$ and $f_{out}(C, G)$ respectively.

The following lemma can be proved easily using the conservation of an (s, t) -flow.

► **Lemma 5.** *For any (s, t) -cut C in H , $f_{out}(C, H) - f_{in}(C, H) = \text{value}(f, H)$.*

For a set $U \subseteq V$, $E_{vit}(U)$ denotes the set of vital edges whose both endpoints belong to U . We now introduce a notation that quantitatively captures the *vitality* of an edge.

- $w_{min}(e)$: the reduction in the capacity of (s, t) -mincut in G after the failure of edge e .
- The following observation is immediate from Definition 1.

► **Observation 6.** *For any vital edge e , $w_{min}(e) > 0$.*

► **Definition 7** (A mincut cover for a set of edges). *A set \mathcal{A} consisting of (s, t) -cuts is said to be a mincut cover for a set of edges E' if, for each edge $e \in E'$, at least one mincut for e is present in \mathcal{A} , and $|\mathcal{A}|$ is the smallest.*

The following lemma provides a maximum (s, t) -flow based characterization for a vital edge.

► **Lemma 8.** *An edge e is vital if and only if $f(e) > 0$ in every maximum (s, t) -flow f in G .*

Lemma 8 can be proved easily using the strong duality between maximum (s, t) -flow and (s, t) -mincut [14].

► **Definition 9** (Edge-set of a cut). *A cut C is said to separate a pair of vertices $u, v \in V$ if either $u \in C$ and $v \in \bar{C}$ or $v \in C$ and $u \in \bar{C}$. The edge-set of C is the set of all those edges whose endpoints are separated by C .*

For an undirected graph, the set of contributing edges of a cut is the edge-set of the cut. In a seminal work, Gomory and Hu [17] established the following two results for an undirected graph. (1) There is a set of $n - 1$ cuts such that the cut of the least capacity separating each pair of vertices is present in this set. (2) Each of these $n - 1$ cuts appears as a cut in a spanning tree on the vertex set V . This tree is widely-known as Gomory-Hu tree.

The construction of Gomory-Hu tree crucially exploits that the capacity of each cut in graph is defined as the sum of the capacities of all contributing edges of the cut. Suppose capacity of each cut is defined by any arbitrary real-valued function F . Even for this generic

setting, Cheng and Hu [9] showed that there exists a set consisting of $n - 1$ cuts such that a cut of the least capacity (F -value) separating any pair of vertices is present in the set. Furthermore, Cheng and Hu [9] showed that all these cuts can be stored in a suitably augmented rooted full binary tree, called *ancestor tree* as follows. Every vertex of the given graph is mapped to a unique leaf node in the ancestor tree, and the cut of the least capacity separating any pair of vertices is stored at their lowest common ancestor (LCA) in the tree. The ancestor tree occupies $\mathcal{O}(n^2)$ space and, Cheng and Hu [9] also designed an efficient algorithm to construct the tree – It performs only $\mathcal{O}(n)$ calls to an algorithm that, given any pair of vertices, can report a cut of the least capacity separating them.

Ausiello, Franciosa, Lari, and Ribichini [3] showed that the ancestor tree of Cheng and Hu [9] can be constructed for (s, t) -cuts as well if function F is defined as follows.

$$\text{For a set } C \subset V, F(C) = \begin{cases} c(C), & \text{if } s \in C \text{ and } t \in \bar{C} \\ \infty, & \text{otherwise.} \end{cases} \quad (1)$$

This insight played a crucial role in the computation of all vital edges in an undirected graph using $\mathcal{O}(n)$ maximum (s, t) -flow computations [3].

3 A Generalization of FlowCut Property

Let $E_{min} \subseteq E$ be the set of all edges contributing to any (s, t) -mincut. Ford and Fulkerson [14] established a strong duality between (s, t) -mincut and maximum (s, t) -flow. Exploiting this, the following property provides a maximum (s, t) -flow based characterization for mincut for any edge $e \in E_{min}$.

FLOWCUT: *Let C be an (s, t) -cut and $e \in E_{min}$ is a contributing edge of C . C is a mincut for edge e if and only if for any maximum (s, t) -flow, each contributing edge of C is fully saturated and each incoming edge of C carries no flow.*

For directed weighted graphs, we know that E_{min} maybe only a proper subset of the set of all vital edges. We now present a generalization of FLOWCUT property that provides a maximum (s, t) -flow based characterization for each mincut $C(e)$, $\forall e \in E_{vit}$.

► **Theorem 10 (GENFLOWCUT).** *Let C be an (s, t) -cut and a vital edge $e = (u, v)$ contributes to C in G . C is a mincut for e if and only if there is a maximum (s, t) -flow f such that*

1. $f_{in}(C) = 0$.
2. e carries exactly $w_{min}(e)$ amount of (s, t) -flow and every other contributing edge e' in C is fully saturated, that is, $f(e') = w(e')$.

Proof. Suppose C is a mincut for vital edge e in G . After the removal of edge e , the capacity of (s, t) -cut C is related to f^* as follows.

$$\text{In graph } G \setminus \{e\}, c(C) = f^* - w_{min}(e) \quad (2)$$

Let G' be the graph obtained from G after reducing the capacity of edge e from $w(e)$ to $w_{min}(e)$. Therefore, using Equation 2, the capacity of C in G' is $(f^* - w_{min}(e)) + w_{min}(e) = f^*$. Let $\mathcal{C}_{u,v}$ be the set of all (s, t) -cuts that keep u on the side of s and v on the side of t . The capacity of each cut in $\mathcal{C}_{u,v}$ gets reduced by the same amount due to reduction in the capacity of e . Therefore, C is a mincut for edge e in G' as well. Hence the capacity of every cut belonging to $\mathcal{C}_{u,v}$ in G' is at least f^* . Capacity of any (s, t) -cut in G' , that does not belong to $\mathcal{C}_{u,v}$, is at least f^* since it remains unaffected by the reduction in the capacity of

edge e . These facts imply that f^* is the capacity of (s, t) -mincut in G' . Thus, using the strong duality between maximum (s, t) -flow and (s, t) -mincut, it follows that there exists a maximum (s, t) -flow f in G' such that $\text{value}(f, G') = f^*$. Using Lemma 5 for C in G' , we get the following equality.

$$f_{out}(C, G') - f_{in}(C, G') = f^* \quad (3)$$

It follows from the capacity constraint that $f_{out}(C, G') \leq c(C)$. So, Equation 3 implies that $f_{in}(C, G') \leq c(C) - f^*$. Since $c(C) = f^*$ in G' as shown above, we arrive at the following.

$$f_{in}(C, G') = 0 \quad \text{and} \quad f_{out}(C, G') = f^* \quad (4)$$

This implies that, in G' , each outgoing edge of C is fully saturated and each incoming edge does not carry any amount of flow. f is also a valid maximum (s, t) -flow for graph G because f^* is the value of maximum (s, t) -flow in G and $f((u, v)) \leq w((u, v))$ in G . Therefore, it follows from Equation 4 that in graph G , $f_{in}(C) = 0$, each outgoing edge of C is fully saturated, and edge e carries exactly $w_{min}(e)$ amount of (s, t) -flow.

We now prove the converse part. Suppose there is a maximum (s, t) -flow in G such that each outgoing edge of C , except e , is fully saturated and each incoming edge carries no flow. Therefore, $c(C)$ in G is $f^* - w_{min}(e) + w(e)$. So, in graph $G \setminus \{e\}$, capacity of C is $f^* - w_{min}(e)$. Since e is a vital edge, it follows from definition that $f^* - w_{min}(e)$ is the capacity of (s, t) -mincut in $G \setminus \{e\}$. Therefore, C is an (s, t) -mincut in $G \setminus \{e\}$. Hence, each (s, t) -cut in G that keeps u on the side of s and v on the side of t has a capacity at least $f^* - w_{min}(e) + w(e)$. This implies that C is a mincut for edge e . ◀

► **Remark 11.** Theorem 10 crucially exploits Equation 2 which holds for vital edges only. Note that $w_{min}(e) = 0$ for a nonvital edge e . So, if C is a mincut for e , C might have capacity $> f^*$ even after removing nonvital edge e from G .

4 A Mincut Cover for All Vital Edges

For any subset \mathcal{E} of vital edges, let $V(\mathcal{E})$ denote the smallest set of vertices such that for each edge $(u, v) \in \mathcal{E}$, both u and v belong to $V(\mathcal{E})$. We establish an upper bound on the cardinality of the mincut cover of \mathcal{E} in terms of $|V(\mathcal{E})|$.

Let e be a vital edge, and let C be a mincut for e . In undirected graphs, recall that each edge belonging to the edge-set of C is a contributing edge of C . However, in directed graphs, there may exist edges incoming to C . Any such incoming edge, say e' , is not a contributing edge of C , and certainly, any mincut for edge e' is different from C . Can e' be a vital edge? The following lemma answers this question in negation. It crucially exploits the properties of (s, t) -maxflow across C as stated in GENFLOWCUT Property (Theorem 10).

► **Lemma 12.** *If C is a mincut for a vital edge, each incoming edge of C must be a nonvital edge.*

Proof. Let e' be an incoming edge of C . It follows from Theorem 10(1) that there is a maximum (s, t) -flow f such that $f_{in}(C) = 0$. So e' does not carry any flow in maximum (s, t) -flow f . Hence, it follows from Lemma 8 that e' is not a vital edge in G . ◀

The following lemma, which can be seen as a corollary of Lemma 12, establishes that a mincut for a vital edge partitions all the vital edges into three sets.

► **Lemma 13.** *Let $G = (V, E)$ be a directed weighted graph with a designated source vertex s and a designated sink vertex t . Let \mathcal{E} be a subset of vital edges. For any edge $e \in \mathcal{E}$, let C be a mincut for e . C partitions the entire set \mathcal{E} into three subsets – (i) \mathcal{E}_C : edges that are contributing to C , (ii) \mathcal{E}_L : edges whose both endpoints belong to C , and (iii) \mathcal{E}_R : edges whose both endpoints belong to \overline{C} .*

Let $\mathcal{M}(\mathcal{E})$ denote the mincut cover for a subset \mathcal{E} of vital edges. Suppose Lemma 13 additionally guarantees the following property.

\mathcal{P} : For each vital edge $e' \in \mathcal{E}$ that contributes to C , C is a mincut for e' as well.

Property \mathcal{P} ensures that C is a mincut for all vital edges that belong to \mathcal{E}_C . This implies that $\mathcal{M}(\mathcal{E}_L) \cup \mathcal{M}(\mathcal{E}_R) \cup \{C\}$ is a mincut cover for \mathcal{E} as well. Therefore, the cardinality of mincut cover for set \mathcal{E} can be bounded as follows.

$$|\mathcal{M}(\mathcal{E})| \leq |\mathcal{M}(\mathcal{E}_L)| + |\mathcal{M}(\mathcal{E}_R)| + 1 \quad (5)$$

Let $\mathcal{N}(\mu)$ denote the cardinality of a mincut cover for any set \mathcal{E} of vital edges with $\mu = |V(\mathcal{E})|$. Note that $\mathcal{E} = \emptyset$ implies $\mu = 0$, and $\mathcal{E} \neq \emptyset$ implies $\mu \geq 2$. Equation 5, Lemma 13, and Property \mathcal{P} lead to the following recurrence for $\mathcal{N}(\mu)$.

Base case: $\mathcal{N}(0) = 0$.

For any $\mu \geq 2$, $\mathcal{N}(\mu) \leq 1 + \mathcal{N}(\mu_1) + \mathcal{N}(\mu_2)$, where $\mu_1 = |V(\mathcal{E}_L)|$, $\mu_2 = |V(\mathcal{E}_R)|$ (6)

Note that $\mu_1, \mu_2 < \mu$, and $\mu_1 + \mu_2 \leq \mu$. Using induction on μ , it is a simple exercise to show that $\mathcal{N}(\mu) \leq \mu - 1$ for all $\mu \geq 2$.

Though property \mathcal{P} is crucially used in establishing an upper bound on the mincut cover of a set of vital edges, Lemma 13, in its current form, does not guarantee it. However, we can enforce \mathcal{P} easily as follows. In Lemma 13, the mincut for edge e is of the least capacity among mincuts for all edges in \mathcal{E} .

Using Recurrence 6 for the entire set of vital edges, we get an upper bound of $n - 1$ on the cardinality of the mincut cover for all vital edges in G . This leads to the following theorem that answers Question 3 in the affirmative.

► **Theorem 14 (Mincut Cover).** *For any directed weighted graph G on n vertices with a designated source vertex s and a designated sink vertex t , there exists a set \mathcal{C}_{min} containing at most $n - 1$ (s, t) -cuts such that, for any vital edge e in G , at least one mincut for edge e is present in set \mathcal{C}_{min} .*

The following note emphasizes the need of property \mathcal{P} in establishing the upper bound derived above.

► **Note 15.** Selecting any arbitrary edge $e \in \mathcal{E}$ might skip some mincuts for vital edges. Refer to Figure 1(ii) on Page 3. The set of vital edges, denoted by \mathcal{E} , contains four vital edges (s, a) , (s, b) , (a, t) , and (b, t) . Suppose we first select vital edge (s, a) and mincut $A = \{s\}$ for edge (s, a) . It follows from Lemma 13 that \mathcal{E} is partitioned into \mathcal{E}_L , \mathcal{E}_R and \mathcal{E}_A . Observe that \mathcal{E}_L does not contain any vital edge, vital edges (a, t) and (b, t) belong to \mathcal{E}_R , and (s, b) belongs to \mathcal{E}_A . We now recurse on set $\mathcal{E} = \mathcal{E}_R$ and select the vital edge (b, t) . Mincut $C = \{s, a, b\}$ partitions \mathcal{E} into \mathcal{E}_L , \mathcal{E}_R , and \mathcal{E}_C . Observe that both \mathcal{E}_L and \mathcal{E}_R do not contain any vital edge, and vital edge (a, t) belongs to \mathcal{E}_C . The process terminates as $V(\mathcal{E}_L)$ and $V(\mathcal{E}_R)$ are empty. We get a pair of (s, t) -cuts A and C as mincut cover for \mathcal{E} . Unfortunately, it does not contain mincut $B = \{s, a\}$ for edges (a, t) and (s, b) .

5 Optimal Sensitivity Oracles for (s,t) -mincut

Let $e = (u, v)$ be an edge in G , and we wish to determine the impact of the failure of e on (s, t) -mincut. We begin with a brief overview of $\mathcal{O}(n)$ space sensitivity oracle [4] for (s, t) -mincuts in unweighted graphs. By construction of $\mathcal{D}_{PQ}(G)$ [30], there is a mapping from the vertices of G to the nodes of $\mathcal{D}_{PQ}(G)$ such that the following assertion holds – u and v are mapped to the same node of $\mathcal{D}_{PQ}(G)$ if and only if there is no (s, t) -mincut in G that separates u and v . In $\mathcal{D}_{PQ}(G)$, each 1-transversal cut is an (s, t) -mincut in G . Exploiting this property, it is shown in [4] that there is an (s, t) -mincut to which edge (u, v) contributes if and only if the node containing u succeeds the node containing v in any topological ordering of $\mathcal{D}_{PQ}(G)$. If u and v are mapped to the same node of $\mathcal{D}_{PQ}(G)$, it follows that the mincut for edge (u, v) has a capacity strictly larger than that of the (s, t) -mincut. Therefore, if G is unweighted, the failure of edge (u, v) does not reduce the capacity of the (s, t) -mincut; so (u, v) is not a vital edge. Thus, the mapping from V to the nodes of $\mathcal{D}_{PQ}(G)$ along with its topological ordering serves as an $\mathcal{O}(n)$ space sensitivity oracle in unweighted graphs.

If G is weighted, it is still possible that though u and v mapped to the same node of $\mathcal{D}_{PQ}(G)$, yet the failure of edge (u, v) reduces (s, t) -mincut. In other words, if G is weighted, there may be many vital edges internal to the nodes of $\mathcal{D}_{PQ}(G)$ (refer to Figure 2). Thus $\mathcal{D}_{PQ}(G)$ fails to serve as sensitivity oracle for a weighted graph G ; hence we need to explore the structure of cuts internal to a node of $\mathcal{D}_{PQ}(G)$.

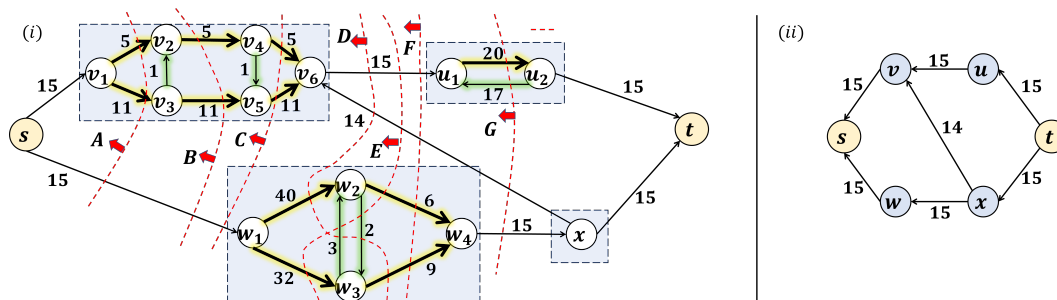


Figure 2 (i) A graph H and (ii) $\mathcal{D}_{PQ}(H)$. Thick edges in (i) represent the vital edges of H that are internal to the nodes of $\mathcal{D}_{PQ}(H)$. A mincut for them is represented by dashed curves.

Let $\text{CAP}(e, \Delta)$ denote the query for reporting the capacity of (s, t) -mincut after reducing the capacity of $e \in E$ by Δ such that $0 \leq \Delta \leq w(e)$. We now address the design of a compact data structure that can efficiently answer query $\text{CAP}(e, \Delta)$ for G .

Let us first consider only the set of vital edges. There may exist $\Omega(n^2)$ vital edges in a graph (refer to Figure 4(ii) on Page 17). Therefore, explicitly storing the capacity of a mincut for each vital edge would occupy $\Omega(n^2)$ space. To design an $\mathcal{O}(n)$ space data structure, we use the following lemma for vital edges. This lemma follows from Lemma 13 and the discussion we used for bounding the size of mincut cover for vital edges.

► **Lemma 16.** *Let $U \subseteq V$ and an edge $e \in E_{\text{vit}}(U)$. If mincut for e , say C , has the least capacity among the mincuts for all vital edges from $E_{\text{vit}}(U)$, then, for any edge $e' \in E_{\text{vit}}(U)$, either C is a mincut for e' or both endpoints of e' belong to $U \cap C$ or $U \cap \bar{C}$.*

Using Lemma 16, we design a divide and conquer based algorithm to build a data structure that compactly stores the capacity of mincut for each vital edge as follows.

17:10 Vital Edges for (s,t)-Mincut: Algorithms, Structures and Sensitivity Oracles

Let e^* be an edge in $E_{vit}(V)$ such that the capacity of mincut for e^* is less than or equal to the capacity of mincut for each vital edge in G . Let C be a mincut for e^* . C serves as the mincut for each vital edge contributing to C . For the mincuts of the remaining vital edges, we recursively process $E_{vit}(V \cap C)$ and $E_{vit}(V \cap \overline{C})$. Refer to Algorithm 1 for the pseudocode of this recursive algorithm.

It can be observed that the data structure resulting from Algorithm 1 is a rooted binary tree. We refer to it as $\mathcal{T}_{vit}(G)$ henceforth. Its leaves are associated with disjoint subsets of the entire vertex set V – for each vertex $v \in V$, $\mathcal{L}(v)$ stores the pointer to the leaf to which v is mapped. Each internal node ν in $\mathcal{T}_{vit}(G)$ has the following three fields.

- $\nu.cap$ is the capacity of mincut for the vital edge selected during the recursive call in which node ν is created.
- $\nu.left$ is the left child of ν and $\nu.right$ is the right child of ν .

■ **Algorithm 1**: TREECONSTRUCTION(V) computes $\mathcal{T}_{vit}(G)$.

```

1: procedure TREECONSTRUCTION( $U$ )
2:   Create a node  $\nu$ ;
3:   if  $E_{vit}(U) = \emptyset$  then
4:     for each  $x \in U$  do  $\mathcal{L}(x) \leftarrow \nu$ ;
5:   end for
6:   else
7:     Let  $C(e)$  denote a mincut for edge  $e$ ;
8:     Select an edge  $e^* \in E_{vit}(U)$  such that  $c(C(e^*)) \leq c(C(e')) \forall e' \in E_{vit}(U)$ ;
9:     Assign  $\nu.cap \leftarrow c(C(e^*))$ ;
10:     $\nu.left \leftarrow$  TREECONSTRUCTION( $U \cap C(e^*)$ );
11:     $\nu.right \leftarrow$  TREECONSTRUCTION( $U \cap \overline{C(e^*)}$ );
12:   end if
13:   return  $\nu$ ;
14: end procedure

```

The following observation is immediate from the construction of $\mathcal{T}_{vit}(G)$.

► **Observation 17.** *Let ν be an internal node in $\mathcal{T}_{vit}(G)$ and C be the (s,t) -cut associated with node ν (chosen in Step 8 of Algorithm 1). For a vertex u , $\mathcal{L}(u)$ belongs to the subtree rooted at $\nu.left$ if and only if $u \in C$.*

Observe that $\mathcal{T}_{vit}(G)$ is a full binary tree with at most n leaves. So the number of internal nodes is at most $n - 1$. Moreover, every internal node of $\mathcal{T}_{vit}(G)$ stores $\mathcal{O}(1)$ information. Hence, we can state the following theorem.

► **Theorem 18.** *Let $G = (V, E)$ be a directed weighted graph on $n = |V|$ vertices with designated source vertex s and designated sink vertex t . There is a rooted full binary tree occupying $\mathcal{O}(n)$ space that stores the capacity of mincut for each vital edge.*

Let $e = (x, y)$ be a query edge. If e is a vital edge, it follows from the construction of tree $\mathcal{T}_{vit}(G)$ in Theorem 18 that the LCA of $\mathcal{L}(x)$ and $\mathcal{L}(y)$ stores the capacity of mincut for (x, y) . So, using the efficient data structure for LCA [7], it takes $\mathcal{O}(1)$ time to answer $CAP(e, \Delta)$ for any vital edge e . However, what if the query edge is a nonvital edge? Although there is no impact on (s, t) -mincut capacity due to the reduction in the capacity of a nonvital edge, tree $\mathcal{T}_{vit}(G)$ stores no information explicitly about nonvital edges. Therefore, in order to answer $CAP(e, \Delta)$ for any edge, it seems a classification of all edges as vital or nonvital is required. Unfortunately, any such explicit classification would require $\Omega(m)$ space, which is also trivial for answering query CAP as discussed above.

Interestingly, $\mathcal{T}_{vit}(G)$ can answer $\text{CAP}(e, \Delta)$ for any edge e without any such classification. This is because, as shown in the following lemma, it is already capable of classifying an edge to be vital or nonvital.

► **Lemma 19.** *Let $e = (x, y)$ be any edge in G such that $\mathcal{L}(x) \neq \mathcal{L}(y)$. Let ν be the LCA of $\mathcal{L}(x)$ and $\mathcal{L}(y)$ in $\mathcal{T}_{vit}(G)$, and let C be the (s, t) -cut associated with node ν in $\mathcal{T}_{vit}(G)$. Edge (x, y) is a nonvital edge if and only if exactly one of the following two conditions is satisfied.*

1. $\mathcal{L}(x) \in \nu.right$ and $\mathcal{L}(y) \in \nu.left$.
2. $\mathcal{L}(x) \in \nu.left$, $\mathcal{L}(y) \in \nu.right$, and $c(C) - w(e) \geq f^*$.

Proof. Suppose edge (x, y) is a nonvital edge. Observe that $\mathcal{L}(x)$ and $\mathcal{L}(y)$ belong to the subtree rooted at ν in $\mathcal{T}_{vit}(G)$ because ν is the LCA of $\mathcal{L}(x)$ and $\mathcal{L}(y)$. It follows from the construction of $\mathcal{T}_{vit}(G)$ (Algorithm 1) that edge (x, y) either contributes to C or is an incoming edge to C . Suppose (x, y) is a contributing edge to C . So using Observation 17, $\mathcal{L}(x) \in \nu.left$ and $\mathcal{L}(y) \in \nu.right$. If $c(C)$ falls below f^* upon removal of edge (x, y) , (x, y) would be a vital edge by Definition 1. Hence, $c(C) - w(e) \geq f^*$. So condition (2) is satisfied. Suppose (x, y) is an incoming edge to C , that is, $x \notin C$ and $y \in C$. It follows from Observation 17 that $\mathcal{L}(x)$ must belong to the right subtree and $\mathcal{L}(y)$ must belong to the left subtree of ν . So condition (1) is satisfied.

We now prove the converse part. Suppose condition (1) is satisfied, that is, $\mathcal{L}(x) \in \nu.right$ and $\mathcal{L}(y) \in \nu.left$. Observation 17 implies that $x \in \bar{C}$ and $y \in C$. Hence, edge (x, y) is an incoming edge of C . It follows from the construction of $\mathcal{T}_{vit}(G)$ that C is a mincut for some vital edge. So, by Theorem 10(1), there exists a maximum (s, t) -flow f such that $f_{in}(C) = 0$. Hence $f(x, y) = 0$. So it follows from Lemma 8 that (x, y) is a nonvital edge. Suppose condition (2) is satisfied. Hence (x, y) is an outgoing edge of C . Assume to the contrary that e is a vital edge. It follows from the construction of $\mathcal{T}_{vit}(G)$ that C is a mincut for e . Since e is a vital edge, by Definition 1, $c(C) - w(e) < f^*$, a contradiction. ◀

For any edge $e \in E$ and $0 \leq \Delta \leq w(e)$, Algorithm 2 verifies conditions of Lemma 19 to answer query $\text{CAP}(e, \Delta)$ in $\mathcal{O}(1)$ time using $\mathcal{T}_{vit}(G)$. So we can state the following Theorem.

■ **Algorithm 2** Reporting the capacity of (s, t) -mincut after decreasing the capacity of an edge.

```

1: procedure CAP( $(x, y), \Delta$ )
2:   if  $\mathcal{L}(x) == \mathcal{L}(y)$  then
3:     return  $f^*$ 
4:   else
5:      $\nu \leftarrow$  LCA of  $\mathcal{L}(x)$  and  $\mathcal{L}(y)$  in  $\mathcal{T}_{vit}(G)$ .
6:   end if
7:    $NewCapacity \leftarrow \nu.cap - \Delta$ 
8:   if  $(\mathcal{L}(x) \in \nu.left \wedge \mathcal{L}(y) \in \nu.right) \wedge NewCapacity < f^*$  then
9:     return  $NewCapacity$ .
10:  else
11:    return  $f^*$ .
12:  end if
13: end procedure

```

► **Theorem 20** (Sensitivity Oracle for Reporting Capacity). *Let G be a directed weighted graph on n vertices with a designated source vertex s and a designated sink vertex t . There is an $\mathcal{O}(n)$ space data structure that, given any edge $e \in E$ and a value Δ satisfying $0 \leq \Delta \leq w(e)$, can report in $\mathcal{O}(1)$ time the capacity of (s, t) -mincut after reducing the capacity of e by Δ .*

Reporting an (s, t) -mincut. An internal node, say ν , of $\mathcal{T}_{vit}(G)$ stores only the capacity of the mincut for a vital edge. We augment ν with a mincut for the edge that was picked by Algorithm 1 in Step 8. The size of the resulting data structure is $\mathcal{O}(n^2)$. Observe that the condition $\Delta \leq w(e)$ can be verified for any edge e in $\mathcal{O}(1)$ time if we store the capacity of all edges of G . This will require only additional $\mathcal{O}(m) = \mathcal{O}(n^2)$ space. So we can state the following theorem.

► **Theorem 21 (Sensitivity Oracle).** *Let G be a directed weighted graph on n vertices with a designated source vertex s and a designated sink vertex t . There is an $\mathcal{O}(n^2)$ space data structure that, given any edge e and any value $\Delta \geq 0$, can report*

1. *the capacity of (s, t) -mincut in $\mathcal{O}(1)$ time, and*
2. *an (s, t) -mincut C in $\mathcal{O}(|C|)$ time for the resulting graph*
3. *the DAG \mathcal{D}_{PQ} occupying $\mathcal{O}(m)$ space in $\mathcal{O}(m)$ time (proof is in the full version) after reducing the capacity of the edge e by Δ .*

► **Remark 22.** It is a simple exercise to show that our data structure in Theorem 21(1) and (2) can be extended to handle a vertex failure using the standard technique of vertex splitting.

Lower Bound. We complement the result in Theorem 21 by a matching lower bound, which holds even for undirected graphs as follows (refer to Section 8 for the proof).

► **Theorem 23.** *Any data structure for reporting the capacity of (s, t) -mincut after the failure of an edge in an (un)directed graph on n vertices with positive edge capacities must require $\Omega(n^2 \log n)$ bits of space in the worst case, irrespective of the query time.*

► **Remark 24.** The $\mathcal{O}(n)$ space upper bound in Theorem 20 does not violate the $\Omega(n^2)$ space lower bound in Theorem 23. This is because Theorem 20 assumes that the query edge e belongs to E and the reduction in capacity of e is at most $w(e)$. However, this assumption seems practically justified since, in real world, the capacity of an edge can decrease only if the edge *actually* exists, and furthermore, it can decrease by an amount at most the capacity of the edge.

Labeling Scheme. Let f be any function defined on the vertex/edge set of the graph. A labeling scheme of f assigns small labels to each vertex/edge of the graph in such a way that, for any given subset A of edges/vertices, $f(A)$ can be computed only by using the labels of vertices/edges in A . A labeling scheme of $\mathcal{O}(\log^2 n + \log n \log W)$ bits can be designed for reporting capacity of (s, t) -mincut after decreasing capacity of any edge in G using Theorem 20 and a labeling scheme for LCA query [25]. Here W is the maximum capacity of any edge in G . The proof is in the full version.

6 An Algorithm for Computing All Vital Edges

The existing algorithms [2, 3] that compute the most vital edge or all vital edges in undirected graphs take *cut-based* approaches as follows. For computing the most vital edge in an undirected graph, Aneja, Chandrasekaran, and Nair [2] establish that there exists an (s, t) -cut such that the most vital edge is the maximum capacity edge among all the edges that contribute to the cut. Ausiello, Franciosa, Lari, and Ribichini [3] define function F on cuts as in Equation 1 in order to use the ancestor tree of Cheng and Hu [9]. To compute all vital edges efficiently in a directed weighted graph, we take a *flow-based* approach that analyzes flow along a set of edges in a maximum (s, t) -flow. We first classify all vital edges into two types based on the maximum flow that an edge can carry in any maximum (s, t) -flow.

A classification of vital edges. In graph G , there may exist multiple (s, t) -flows attaining value f^* . The flow along an edge may be different in these multiple maximum (s, t) -flows. A vital edge is a *tight* edge if it carries flow equal to its capacity in at least one maximum (s, t) -flow; otherwise, it is a *loose* edge.

Our algorithm for computing all vital edges makes use of the classification of vital edges, and employs two results that were derived for solving two seemingly *unrelated* problems.

To compute all the loose edges, we crucially exploit the following result on maximum (s, t) -flow, which has been used for solving *mincost maximum (s, t) -flow problem*.

► **Lemma 25** (Theorem 11.1 and Theorem 11.2 in [1]). *For any directed weighted graph G , there exists a maximum (s, t) -flow $f^\#$ such that the edges that carry nonzero flow but not fully saturated are at most $n - 1$.*

For any maximum (s, t) -flow, by definition, the set of all loose edges is a subset of all edges that carry nonzero flow but are not fully saturated. So, by Lemma 25, the number of loose edges can be at most $n - 1$. Moreover, there is an algorithm that, given any maximum (s, t) -flow, can compute $f^\#$ of Lemma 25 in $\mathcal{O}(mn)$ time [1]. As a result, all the loose edges of G can be computed using $\mathcal{O}(n)$ maximum (s, t) -flow computations.

Observe that the efficient computation of the set of loose edges crucially exploits the fact that there can be at most $n - 1$ loose edges. However, there can be $\Omega(n^2)$ tight edges in a graph (refer to Figure 4(ii) on page 17). Hence, to compute all the vital edges, the main challenge arises in computing all the tight edges of G . We now state a property of tight edges that plays a crucial role in their efficient computation.

A property satisfied by tight edges

Let E_{min} be the set of all edges that contribute to (s, t) -mincuts. Set E_{min} is a subset of all the tight edges since all edges in E_{min} are fully saturated in every maximum (s, t) -flow. Exploiting the strong duality between maximum (s, t) -flow & (s, t) -mincut and Lemma 5, it can be observed that each edge $(u, v) \in E_{min}$ satisfies the following property. If C is any (s, t) -cut such that $v \in C$ and $u \in \bar{C}$, then $c(C)$ is strictly greater than f^* . The following lemma shows that this property is satisfied by each tight edge as well.

► **Lemma 26.** *Let $e = (u, v)$ be a tight edge. Let C be a mincut for edge e and $C_{v,u}$ be an (s, t) -cut of the least capacity that keeps v on the side of s and u on the side of t . Then, $c(C) < c(C_{v,u})$.*

Proof. Edge (u, v) is a tight edge, so by definition, there is a maximum (s, t) -flow f such that $f(e)$ is equal to $w(e)$. This would imply that $f_{in}(C_{v,u})$ is at least $w(e)$ since edge (u, v) is an incoming edge for cut $C_{v,u}$. It follows from Lemma 5 that $f_{out}(C_{v,u}) - f_{in}(C_{v,u}) = f^*$. Therefore, $f_{out}(C_{v,u}) - w(e) \geq f^*$. Since the capacity of an (s, t) -cut is an upper bound on the value of any outgoing flow of the (s, t) -cut, therefore, $f_{out}(C_{v,u}) \leq c(C_{v,u})$. Hence we arrive at the following inequality.

$$c(C_{v,u}) \geq f^* + w(e) \tag{7}$$

It follows from Theorem 10(2) that $w_{min}(e)$ is the minimum amount of flow that must pass through edge (u, v) to get a maximum flow of value f^* . Let f' be the value of maximum (s, t) -flow in graph $G \setminus \{e\}$. Thus we arrive at the following equality.

$$f^* = f' + w_{min}(e) \tag{8}$$

17:14 Vital Edges for (s,t)-Mincut: Algorithms, Structures and Sensitivity Oracles

We now add $(w(e) - w_{min}(e))$ on both sides of Equation 8 and get the following equation.

$$f^* + w(e) - w_{min}(e) = f' + w(e) \quad (9)$$

It follows from Theorem 10(2) that f' is also equal to the sum of capacities of all edges contributing to C except edge (u, v) . Therefore, $f' + w(e)$ is equal to the capacity of C in G . So it follows from Equation 9 that $f^* + w(e) - w_{min}(e) = c(C)$. Since (u, v) is a vital edge, therefore, $w_{min}(e) > 0$ using Observation 6. Hence,

$$f^* + w(e) > c(C) \quad (10)$$

It follows from Inequality 7 and Inequality 10 that $c(C_{v,u}) > c(C)$. ◀

For any tight edge (u, v) , let C be an (s, t) -cut of the least capacity that separates vertex u and vertex v . It follows from Lemma 26 that C is also a mincut for edge (u, v) . So we can state the following theorem.

► **Theorem 27.** *Let G be a directed weighted graph with a designated source vertex s and a designated sink vertex t . For any tight edge $e = (u, v)$ in G , a mincut for edge e is identical to an (s, t) -cut of the least capacity that separates vertex u and vertex v .*

Algorithm for computing tight edges

For the given directed weighted graph G , we define function F as in Equation 1 and then compute the Ancestor tree of Cheng and Hu [9] for (s, t) -cuts. For each pair of vertices, $\mathcal{T}_{(s,t)}$ stores an (s, t) -cut of the least capacity separating them at their LCA. This tree, denoted by $\mathcal{T}_{(s,t)}$, can be built using $\mathcal{O}(n)$ maximum (s, t) -flow computations [9]. We augment $\mathcal{T}_{(s,t)}$ with a data structure for LCA queries [7].

To compute all tight edges, we process $\mathcal{T}_{(s,t)}$ and the edges of G as follows. Let (u, v) be an edge in G . We perform LCA query on $\mathcal{T}_{(s,t)}$ for u and v to get the (s, t) -cut, say C , of the least capacity that separates u and v . We determine whether C satisfies the following two conditions.

1. Edge (u, v) contributes to C .
2. $c(C) - w((u, v)) < f^*$

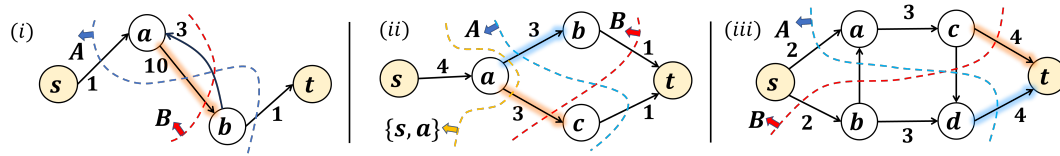
If both conditions are satisfied, by Definition 1, (u, v) is a vital edge. Observe that there may be vital edges that do not satisfy Condition (1); refer to Figure 3(i). However, it follows from Theorem 27 that each tight edge does satisfy these two conditions. After processing all edges of G , let S be the resulting set of vital edges that satisfy both these conditions. We eliminate from S all the loose edges to get all the tight edges.

► **Remark 28.** In an undirected graph, observe that a mincut for an edge (u, v) is always an (s, t) -cut of the least capacity that separates u and v . Therefore, computing all vital edges in an undirected graph amounts to just verifying condition (2) for each edge as shown by Ausiello, Lari, Franciosa, and Ribichini [3].

Thus we can state the following theorem.

► **Theorem 29.** *For any directed weighted graph G on n vertices with a designated source vertex s and a designated sink vertex t , there is an algorithm that computes all tight edges of G using $\mathcal{O}(n)$ maximum (s, t) -flow computations.*

Theorem 29 and the discussion on computing the loose edges lead to Theorem 30.



■ **Figure 3** (i) A is the (s, t) -cut of least capacity that separates a and b . Edge (a, b) is a vital edge, and it is incoming to A . (ii) $\mathcal{D}_{vit}(G) = G$. $\{s, a\}$ is a 1-transversal cut, but not a mincut for any vital edge. (iii) mincuts A and B for vital edges (c, t) and (d, t) , respectively, are not closed under union. Moreover, edge (b, a) (likewise edge (c, d)) contributes to A (likewise to B) and is incoming to B (likewise to A).

► **Theorem 30** (Computing All Vital Edges). *For any directed weighted graph on n vertices with a designated source vertex s and designated sink vertex t , there is an algorithm that computes all vital edges and their vitality using $\mathcal{O}(n)$ maximum (s, t) -flow computations.*

► **Note 31.** We present the following applications of our algorithm in Theorem 30 (refer to the full version).

1. Observe that data structure $\mathcal{T}_{(s,t)}$ also stores a mincut for every tight edge. Therefore, an (s, t) -mincut in graph $G \setminus \{e\}$ for every tight edge e can be computed using $\mathcal{O}(n)$ maximum (s, t) -flow computations. Moreover, since loose edges are at most $n - 1$, therefore, using $\mathcal{O}(n)$ maximum (s, t) -flow computations, we can compute an (s, t) -mincut in graph $G \setminus \{e\}$ for every vital edge e of G .
2. Given a mincut for all vital edges, using Algorithm 1, the process of constructing $\mathcal{T}_{vit}(G)$ and the data structure in Theorem 21(2) requires $\mathcal{O}(m \log n + n^2)$ time. As a byproduct, data structure in Theorem 21(2) provides a mincut cover for all vital edges.
3. As stated in (2), we can construct $\mathcal{T}_{vit}(G)$ using $\mathcal{O}(n)$ maximum (s, t) -flow computations. Given any $k \in [m]$, using $\mathcal{T}_{vit}(G)$ and a maxheap data structure, the k most vital edges can be reported in $\mathcal{O}(m + k \log k)$ time.

► **Note 32.** Given any edge e and its capacity $w(e)$, the data structure in Theorem 20 can be used to report the vitality of e in $\mathcal{O}(1)$ time by assigning $\Delta = w(e)$.

7 Compact Structures for all Mincuts for all Vital Edges

The set containing all (s, t) -mincuts satisfies two important properties – (i) closed under both intersection and union and (ii) an edge contributing to an (s, t) -mincut can never be an incoming edge to another (s, t) -mincut. These two properties are exploited crucially in the design of DAG structures for compactly storing and characterizing all (s, t) -mincuts [30, 4] using 1-transversal cuts. Unfortunately, none of these properties holds for the set of all mincuts for all vital edges (refer to Figure 3(iii)). This makes the problem of designing compact structures for mincuts for all vital edges challenging. We present two structures for storing and characterizing all mincuts for all vital edges – one is a single DAG that provides a *partial* characterization and the other consists of a set of $\mathcal{O}(n)$ DAGs that provides a complete characterization.

(a) An $\mathcal{O}(m)$ Space DAG for Partial Characterization. We first show that the approaches taken in [30, 4] are not sufficient for designing a compact structure for all mincuts for all vital edges. In particular, the resulting graph $Q'(G)$, obtained from G after applying the techniques from [30, 4], is still not acyclic. Moreover, a mincut for a vital edge can have

unbounded transversality (refer to the full version) in $Q'(G)$. It turns out that the source of this problem is the set of all edges that are contributing to a mincut for a vital edge as well as incoming to a mincut for another vital edge. The set of such edges is denoted by Γ -edges. Exploiting GenFlowCut property crucially, we show that all edges in Γ -edges are nonvital. Thereafter, counter intuitive to Note 2, we show that the graph obtained after the removal of all Γ -edges from G still preserves all (s, t) -mincuts of G , and provides a partial characterization for all mincuts for all vital edges as follows.

► **Theorem 33** (Partial Characterization). *For a directed weighted graph G on m edges with a designated source vertex s and a designated sink vertex t , there is an $\mathcal{O}(m)$ space directed acyclic graph $\mathcal{D}_{vit}(G)$ that preserves the capacity of (s, t) -mincut, and for each vital edge e in G , (1) Every mincut for edge e in G is a 1-transversal cut in $\mathcal{D}_{vit}(G)$ and (2) A mincut C for e in G appears as a relevant cut in $\mathcal{D}_{vit}(G)$, that is, $c(C) - w(e) < f^*$ in $\mathcal{D}_{vit}(G)$.*

► **Note 34.** Existing DAG structures for (s, t) -mincuts [30, 4] turn out to be just a special case of DAG $\mathcal{D}_{vit}(G)$ in Theorem 33. Moreover, $\mathcal{D}_{vit}(G)$ additionally guarantees the property (2) in Theorem 33, which does not hold in the existing DAGs [30, 4] (refer to full version).

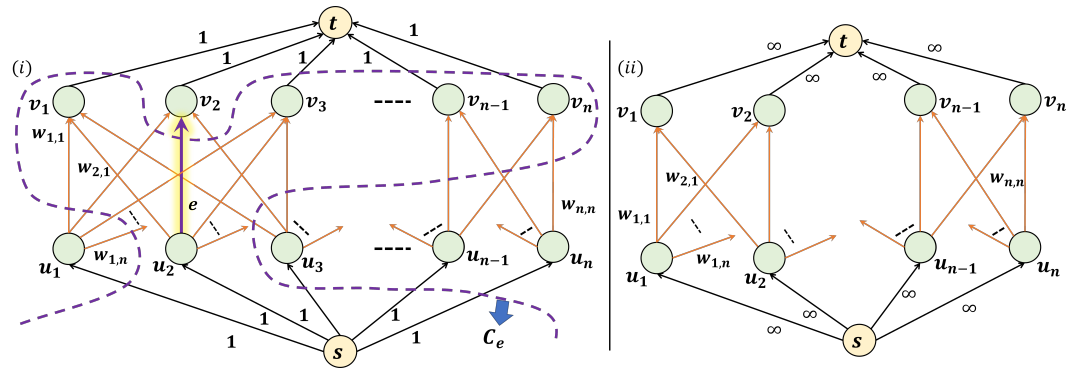
(b) An $\mathcal{O}(mn)$ Space Structure for Complete Characterization. Although each mincut for every vital edge is a 1-transversal cut in $\mathcal{D}_{vit}(G)$, it is not necessary that each 1-transversal cut in $\mathcal{D}_{vit}(G)$ is a mincut for a vital edge (refer to Figure 3(ii)). Hence, $\mathcal{D}_{vit}(G)$ could provide only a partial characterization. Let $e = (u, v)$ be any vital edge, and H_e denote the graph obtained by adding two infinite weight edges, (s, u) and (v, t) , in graph G . Observe that all mincuts for (u, v) are compactly stored in the DAG for (s, t) -mincuts [30] built on graph H_e . We denote this DAG by $\mathcal{D}_{PQ}(H_e)$. It can be seen that keeping $\mathcal{D}_{PQ}(H_e)$ for each vital edge e serves as a compact structure for storing all mincuts for all vital edges and characterizing them in terms of 1-transversal cuts. However, this structure may take $\mathcal{O}(m^2)$ space in the worst case. We present an $\mathcal{O}(mn)$ space structure $\mathcal{S}_{vit}(G)$. This structure consists of $\mathcal{O}(n)$ DAGs for storing and characterizing all mincuts for all vital edges in terms of 1-transversal cuts. To build the structure $\mathcal{S}_{vit}(G)$, the main challenge turns out to be designing a compact structure for storing and characterizing all mincuts for all tight edges. It follows from Theorem 27 that any tight edge $e = (u, v)$ contributes to the (s, t) -cut of the least capacity separating u and v . Therefore, a solution to the following problem would suffice to overcome the challenge.

► **Problem 1.** *Given a directed weighted graph G with a designated source s and a designated sink t , build a compact structure that, for each pair of vertices a and b in G , stores and characterizes all the (s, t) -cuts of the least capacity that separate a and b in G .*

To solve Problem 1, we address the problem of designing a compact structure for storing and characterizing all Steiner (s, t) -mincuts for a given Steiner set, which is of independent interest.

Compact Structure for all Steiner (s, t) -mincuts. Let $S \subseteq V$ be a Steiner set. An (s, t) -cut C is a *Steiner (s, t) -cut* if $C \cap S$ and $\overline{C} \cap S$ are nonempty. A Steiner (s, t) -cut with the least capacity is a *Steiner (s, t) -mincut*. The main hurdle in designing a compact structure for Steiner (s, t) -mincuts is that, unlike (s, t) -mincuts, they are not closed under intersection/union. We overcome this hurdle by generalizing the *covering* technique from [4] to directed weighted graphs and obtain a structure occupying $\mathcal{O}(|S|m)$ space.

Finally, we design a hierarchy tree, which is a generalization of the hierarchy tree of [25], and augment it with the structure for Steiner (s, t) -mincuts. It leads to the following result.



■ **Figure 4** (i) Graph \mathcal{G} . The dashed curve represents the mincut for edge (u_2, v_2) . (ii) Graph $G(M)$. Each edge (u_i, v_j) , $1 \leq i, j \leq n$, is a vital as well as tight edge.

► **Theorem 35 (Complete Characterization).** *For any directed weighted graph G on n vertices and m edges with a designated source vertex s and a designated sink vertex t , there is an $\mathcal{O}(mn)$ space structure $\mathcal{S}_{vit}(G)$ consisting of $\mathcal{O}(n)$ DAGs such that for any vital edge e ,*

1. *There exists at most two DAGs, $\mathcal{D}^s(e)$ and $\mathcal{D}^t(e)$, that store all mincuts for edge e .*
2. *An (s, t) -cut C in G is a mincut for edge e if and only if C is a 1-transversal cut in either $\mathcal{D}^s(e)$ or $\mathcal{D}^t(e)$ and e is a contributing edge of C .*

► **Note 36.** Problem 1 is addressed in [4] for unweighted graphs and only for those pair of vertices for which the capacity of (s, t) -cut of the least capacity separating them is minimum+1. An $\mathcal{O}(mn)$ space structure is designed in [4] for this special case of Problem 1. Hence, Theorem 35 provides a generalization of the result in [4] for weighted graphs while matching the space bound. An important application of structure $\mathcal{S}_{vit}(G)$ is Theorem 21(3) (refer to the full version).

8 Lower Bounds

In this section, we first provide a lower bound on the worst case size of mincut cover for all edges. Later, we give a lower bound on the space for any data structure that can report the capacity of (s, t) -mincut after the failure of any edge in both undirected and directed graphs.

8.1 Mincut Cover for All Edges

In this section, we establish the following theorem.

► **Theorem 37.** *There exists a directed weighted graph $G = (V, E)$ on $n = |V|$ vertices such that the number of mincuts for all edges in G is $\Omega(n^2)$.*

In order to prove Theorem 37, we construct a graph \mathcal{G} on $2n + 2$ vertices with a set of $\Omega(n^2)$ edges E' that has the following property. For each pair of edges $e, e' \in E'$, the capacity of mincut for edge e is different from the capacity of mincut for edge e' (refer to Figure 4(i)).

Construction of \mathcal{G} . The vertex set consists of two disjoint sets $A = \{u_1, u_2, \dots, u_n\}$ and $B = \{v_1, v_2, \dots, v_n\}$ of n vertices each, along with a source vertex s and a sink vertex t . The edges are defined as follows. There is an edge with unit capacity from vertex s to each vertex in A . Likewise, there is a unit capacity edge from each vertex in B to sink t . For each $u \in A$ and $v \in B$, there is an edge (u, v) . Let E' be the set of all edges from A to B . Each edge in E' is assigned a unique capacity from $[n^2, 2n^2]$. It can be observed that $w(e_1) \neq w(e_2)$ for every pair of edges $e_1, e_2 \in E'$.

► **Lemma 38.** *For every pair of edges e_1 and e_2 from E' , the capacity of mincut for e_1 is different from the capacity of mincut for e_2 .*

Proof. Let $e = (u, v)$ be an edge from E' . Let us define an (s, t) -cut C_e in which edge e contributes. C_e keeps each vertex $x \in A \setminus \{u\}$ on the side of t . Similarly, C_e keeps each vertex $x \in B \setminus \{v\}$ on the side of s . Therefore, the contributing edges of (s, t) -cut C_e are edge e , the edges from s to each vertex $x \in A \setminus \{u\}$, and the edges from each vertex $x \in B \setminus \{v\}$ to t . Hence the capacity of C_e is $w(e) + (n-1) + (n-1) = w(e) + 2n - 2$. Any other (s, t) -cut in which edge e contributes must have at least one more contributing edge e' from E' . Since $w(e')$ is at least n^2 , therefore, C_e is the mincut for edge e . For any other edge e'' from E' , in a similar way, the capacity of the mincut for e'' is $w(e'') + 2n - 2$, which is different from the capacity of C_e since $w(e) \neq w(e'')$. This completes the proof. ◀

Since $|E'| = \Omega(n^2)$, it follows from Lemma 38 that graph \mathcal{G} has $\Omega(n^2)$ different capacities of mincuts for edges in E' . This completes the proof of Theorem 37.

8.2 Fault-tolerant (s,t)-mincut

In this section, we provide a lower bound on the space for any data structure that can report the capacity of (s, t) -mincut after the failure of any edge. Let M be a $n \times n$ matrix where for any $i, j \in [n]$, $M[i, j]$ stores an integer in the range $[1, n^c]$ for some constant $c > 0$. Given any instance of the matrix M , we construct the following undirected graph $G(M)$ (refer to Figure 4(ii)).

Description of $G(M)$. The vertex set is defined as follows. There is a source vertex s and a sink vertex t . For all the n rows of matrix M , there is a set R of n vertices $\{u_1, u_2, \dots, u_n\}$. Similarly, For all the n columns of matrix M , there is a set C of n vertices $\{v_1, v_2, \dots, v_n\}$. For each $i, j \in [n]$, there is an edge e_{ij} of capacity $w_{i,j} = M[i, j]$ between vertex u_i and v_j . For each $i \in [n]$, there is an edge of infinite capacity between source s and vertex u_i and there is an edge of infinite capacity between vertex v_i and sink t . Let λ be the capacity of (s, t) -mincut in $G(M)$.

We now establish a relation between the graph $G(M)$ and matrix M in the following lemma.

► **Lemma 39.** *For any $i, j \in [n]$, the capacity of (s, t) -mincut in $G(M)$ after the failure of edge (u_i, v_j) is $\lambda - M[i, j]$.*

Proof. In graph $G(M)$, $C = \{s\} \cup R$ is the only (s, t) -cut of finite capacity. Therefore, C is the (s, t) -mincut in graph $G(M)$. Observe that only edges (u_i, v_j) for any $0 < i, j \leq n$ contribute to C . Hence $\lambda = \sum_{i,j \in [n]} w((u_i, v_j))$. Therefore, upon failure of edge $e_{ij} = (u_i, v_j)$, the reduction in the capacity of (s, t) -mincut is the same as $w(e_{ij})$, which is $M[i, j]$ as follows from the construction of $G(M)$ described above. ◀

Let $F(G(M))$ be any data structure that can report the (s, t) -mincut after the failure of an edge in the graph $G(M)$. We use the data structure $F(G(M))$ and Lemma 39 to report $M[i, j]$ for any $0 < i, j \leq n$ as follows.

return $(\lambda - \lambda')$ where λ' is the value returned by $F(G(M))$ after failure of edge (u_i, v_j) .

Note that there are $2^{n^2 c \log n}$ different instances of the matrix M . It follows from Lemma 39 that for any pair of distinct instances of matrix M , the encoding of the corresponding data structures must differ. Therefore, there is an instance of matrix M for which the data structure $F(G(M))$ must occupy $\Omega(n^2 \log n)$ bits of space.

For directed graphs, a lower bound of $\Omega(n^2 \log n)$ bits of space can be achieved by orienting each undirected edge of $G(M)$ as follows. For each $i, j \in [n]$, orient edge (u_i, v_j) from vertex u_i to vertex v_j . For each $i \in [n]$, orient edge (s, u_i) from source s to vertex u_i . Similarly, for each $j \in [n]$, orient edge (v_j, t) from vertex v_j to sink t . The rest of the proof is the same as the case of undirected graphs. This completes the proof of Theorem 23.

By exploiting Theorem 37, the following data structure lower bound can also be established along similar lines to the proof of Theorem 23.

► **Theorem 40.** *Any data structure that, given a pair of vertices $\{u, v\}$, can report the capacity of an (s, t) -cut C of the least capacity such that $u \in C$ and $v \in \overline{C}$ in a directed weighted graph on n vertices must occupy $\Omega(n^2 \log n)$ bits of space.*

References

- 1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.
- 2 Yash P. Aneja, R. Chandrasekaran, and Kunhiraman Nair. Maximizing residual flow under an arc destruction. *Networks*, 38(4):194–198, 2001. doi:10.1002/net.10001.
- 3 Giorgio Ausiello, Paolo Giulio Franciosa, Isabella Lari, and Andrea Ribichini. Max flow vitality in general and st-planar graphs. *Networks*, 74(1):70–78, 2019. doi:10.1002/net.21878.
- 4 Surender Baswana, Koustav Bhanja, and Abhyuday Pandey. Minimum+1 (s, t) -cuts and dual-edge sensitivity oracle. *ACM Trans. Algorithms*, 19(4), October 2023. doi:10.1145/3623271.
- 5 Surender Baswana, Keerti Choudhary, and Liam Roditty. An efficient strongly connected components algorithm in the fault tolerant model. *Algorithmica*, 81:967–985, 2019.
- 6 Surender Baswana and Abhyuday Pandey. Sensitivity oracles for all-pairs mincuts. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 581–609. SIAM, 2022.
- 7 Michael A Bender, Martin Farach-Colton, Giridhar Pemmasani, Steven Skiena, and Pavel Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.
- 8 Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, Simon Krogmann, and Martin Schirneck. Approximate distance sensitivity oracles in subquadratic space. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1396–1409, 2023.
- 9 Chung-Kuan Cheng and T. C. Hu. Ancestor tree for arbitrary multi-terminal cut functions. *Ann. Oper. Res.*, 33(3):199–213, 1991. doi:10.1007/BF02115755.
- 10 Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2016.
- 11 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.
- 12 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.
- 13 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 Journal on Computing*, 30(3):753–808, 2000.
- 14 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.
- 15 Greg N Frederickson and Roberto Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2):244–266, 1999.

- 16 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.
- 17 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961. URL: <http://www.jstor.org/stable/2098881>.
- 18 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.
- 19 Fabrizio Grandoni and Virginia Vassilevska Williams. Faster replacement paths and distance sensitivity oracles. *ACM Trans. Algorithms*, 16(1):15:1–15:25, 2020. doi:10.1145/3365835.
- 20 Frieda Granot and Refael Hassin. Multi-terminal maximum flows in node-capacitated networks. *Discret. Appl. Math.*, 13(2-3):157–163, 1986. doi:10.1016/0166-218X(86)90079-X.
- 21 Dan Gusfield and Dalit Naor. Efficient algorithms for generalized cut-trees. *Networks*, 21(5):505–520, 1991.
- 22 Refael Hassin and Asaf Levin. Flow trees for vertex-capacitated networks. *Discrete applied mathematics*, 155(4):572–578, 2007.
- 23 John Hershberger and Subhash Suri. Erratum to “Vickrey pricing and shortest paths: What is an edge worth?”. In *Annual Symposium on Foundation of Computer Science*, volume 43, pages 809–810. IEEE Computer Society Press, 2002.
- 24 Giuseppe F Italiano, Adam Karczmarz, and Nikos Parotsidis. Planar reachability under single vertex or edge failures. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2739–2758. SIAM, 2021.
- 25 Michal Katz, Nir A. Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004. doi:10.1137/S0097539703433912.
- 26 Kao-Chêng Lin and Maw-Sheng Chern. The most vital edges in the minimum spanning tree problem. *Inf. Process. Lett.*, 45(1):25–31, 1993. doi:10.1016/0020-0190(93)90247-7.
- 27 Stephen H Lubore, HD Ratliff, and GT Sicilia. Determining the most vital link in a flow network. *Naval Research Logistics Quarterly*, 18(4):497–502, 1971.
- 28 Enrico Nardelli, Guido Proietti, and Peter Widmayer. A faster computation of the most vital edge of a shortest path. *Information Processing Letters*, 79(2):81–85, 2001.
- 29 Cynthia A Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785, 1993.
- 30 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.
- 31 H Donald Ratliff, G Thomas Sicilia, and SH Lubore. Finding the n most vital links in flow networks. *Management Science*, 21(5):531–539, 1975.
- 32 Liam Roditty and Uri Zwick. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Transactions on Algorithms (TALG)*, 8(4):1–11, 2012.
- 33 Fu-Shang P Tsen, Ting-Yi Sung, Men-Yang Lin, Lih-Hsing Hsu, and Wendy Myrvold. Finding the most vital edges with respect to the number of spanning trees. *IEEE Transactions on Reliability*, 43(4):600–603, 1994.
- 34 Virginia Vassilevska Williams, Eyob Woldeghebriel, and Yinzhan Xu. Algorithms and lower bounds for replacement paths under multiple edge failure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 907–918. IEEE, 2022.
- 35 Richard D Wollmer. *Some methods for determining the most vital link in a railway network*. Rand Corporation, 1963.