Optimal Sensitivity Oracle for Steiner Mincut

Koustav Bhanja
Department of CSE, IIT Kanpur, India

Abstract

Let $G = (V, E)$ be an undirected weighted graph on $n = |V|$ vertices and $S \subseteq V$ be a Steiner set. Steiner mincut is a well-studied concept, which also provides a generalization to both (s, t)-mincut (when $|S| = 2$) and global mincut (when $|S| = n$). Here, we address the problem of designing a compact data structure that can efficiently report a Steiner mincut and its capacity after the failure of any edge in G; such a data structure is known as a *Sensitivity Oracle* for Steiner mincut.

In the area of minimum cuts, although many Sensitivity Oracles have been designed in unweighted graphs, however, in weighted graphs, Sensitivity Oracles exist only for (s, t)-mincut [Annals of Operations Research 1991, NETWORKS 2019, ICALP 2024], which is just a special case of Steiner mincut. Here, we generalize this result from $|S| = 2$ to any arbitrary set $S \subseteq V$, that is, $2 \leq |S| \leq n$.

We first design an $\mathcal{O}(n^2)$ space Sensitivity Oracle for Steiner mincut by suitably generalizing the approach used for (s, t)-mincuts [Annals of Operations Research 1991, NETWORKS 2019]. However, the main question that arises quite naturally is the following.

Can we design a Sensitivity Oracle for Steiner mincut that breaks the $\mathcal{O}(n^2)$ bound on space?

In this article, we present the following two results that provide an answer to this question.

1. **Sensitivity Oracle:** Assuming the capacity of every edge is known,
   a. there is an $\mathcal{O}(n)$ space data structure that can report the capacity of Steiner mincut in $\mathcal{O}(1)$ time and
   b. there is an $\mathcal{O}(n(n - |S| + 1))$ space data structure that can report a Steiner mincut in $\mathcal{O}(n)$ time

   after the failure of any edge in G.
2. **Lower Bound:** We show that any data structure that, after the failure of any edge in G, can report a Steiner mincut or its capacity must occupy $\Omega(n^2)$ bits of space in the worst case, irrespective of the size of the Steiner set.

The lower bound in (2) shows that the assumption in (1) is essential to break the $\Omega(n^2)$ lower bound on space. Sensitivity Oracle in (1.b) occupies only subquadratic, that is $\mathcal{O}(n^{1+\epsilon})$, space if $|S| = n - n^{\epsilon} + 1$, for every $\epsilon \in [0, 1)$. For $|S| = n - k$ for any constant $k \geq 0$, it occupies only $\mathcal{O}(n)$ space. So, we also present the first Sensitivity Oracle occupying $\mathcal{O}(n)$ space for global mincut. In addition, we are able to match the existing best-known bounds on both space and query time for (s, t)-mincut [Annals of Operations Research 1991, NETWORKS 2019] in undirected graphs.

2012 ACM Subject Classification Theory of computation → Dynamic graph algorithms; Theory of computation → Network flows

Keywords and phrases mincut, (s, t)-mincut, Steiner mincut, fault tolerant structures, data structure, vital edges, vitality, sensitivity oracle

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2024.10

Related Version *Full Version*: `https://arxiv.org/abs/2409.17715`

Funding This research work is partially funded by Research-I Foundation of the Department of CSE, IIT Kanpur, India.

Acknowledgements I am grateful to my doctoral advisor Prof. Surender Baswana for reviewing this article and providing valuable feedback on improving its readability. I also want to thank an anonymous reviewer for his/her insightful comments on this article.

1 Introduction

In the real world, networks (graphs) are often subject to the failure of edges and vertices due to a variety of factors, such as physical damage, interference, or other disruptions. This can lead to changes in the solution to several graph problems. While these failures can happen at any location in the network at any time, they are typically short-lived. Naturally, it requires us to have compact data structures that can efficiently report the solution to the given graph problem (without computing from scratch) once any failure has occurred. Such data structures are known as *Sensitivity Oracles* for several graph problems. There exist elegant Sensitivity Oracles for many fundamental graph problems, such as shortest paths [6, 10], reachability [22, 13], traversals [25, 5], etc.

The minimum cut of a graph is also a fundamental concept of graph theory. Moreover, it has a variety of practical applications in the real world [1]. Designing Sensitivity Oracles for various minimum cuts of a graph has been an emerging field of research for the past few decades [4, 12, 19, 17, 7, 9, 8, 3]. There are two well-known mincuts of a graph. They are global mincut and (s,t)-mincut. Here, we design the first Sensitivity Oracle for global mincut in undirected weighted graphs that can handle the failure of any edge. The concept of Steiner mincut is also well-studied in the area of minimum cuts [18, 16, 19, 8, 14, 21, 23]; moreover, it has global mincut, as well as (s,t)-mincut, as just a special corner case. In this article, as our main result, we present the first *Sensitivity Oracle for Steiner mincut* for handling the failure of any edge in undirected weighted graphs. Interestingly, our result bridges the gap between the two extreme scenarios of Steiner mincut while matching their bounds, namely, (s,t)-mincut [2, 12] and global mincut (designed in this article). In addition, it also provides the first generalization from unweighted graphs [8, 18, 16, 19] to weighted graphs.

Let $G = (V, E)$ be an undirected graph on $n = |V|$ vertices and $m = |E|$ edges with non-negative real values assigned as the capacity to edges. We denote the capacity of an edge e by $w(e)$. Let $S \subseteq V$ be a *Steiner set* of G such that $|S| \geq 2$. A vertex s is called a *Steiner vertex* if $s \in S$; otherwise, s is called a *nonSteiner vertex*.

▶ **Definition 1** (Steiner cut). *A nonempty set $C \subset V$ is said to be a Steiner cut if there is at least one pair of Steiner vertices s, s' such that $s \in C$ and $s' \notin C$.*

For $S = V$, a Steiner cut is a *(global) cut*. Similarly, for $S = \{s, t\}$, a Steiner cut is an *(s,t)-cut*. A cut C is said to *separate* a pair of vertices u, v if $u \in C$ and $v \in \overline{C} = V \setminus C$ or vice versa. An edge $e = (u, v)$ is said to *contribute* to a cut C if C separates endpoints u, v of e. The *capacity of a cut C*, denoted by $c(C)$, is the sum of capacities of all contributing edges of C. A Steiner cut of the least capacity is known as the *Steiner mincut*, denoted by S-mincut. Let λ_S be the capacity of S-mincut. The problem of designing a Sensitivity Oracle for S-mincut for handling the failure of any edge is defined as follows.

▶ **Definition 2** (single edge Sensitivity Oracle for Steiner mincut). *For any graph G, a single edge Sensitivity Oracle for Steiner mincut is a compact data structure that can efficiently report a Steiner mincut and its capacity after the failure of any edge in G.*

For unweighed graphs, there exist single edge Sensitivity Oracles for global mincut [15], (s,t)-mincut [26, 4], and Steiner mincut [18, 16, 8]. Unfortunately, for weighted graphs, in the area of minimum cuts, the only existing results are single edge Sensitivity Oracles for (s,t)-mincut [2, 12, 3]. For undirected weighted graphs, Ausiello et al. [2], exploiting the Ancestor tree data structure of Cheng and Hu [12], designed the first single edge Sensitivity Oracle for (s,t)-mincut. Their Sensitivity Oracle occupies $\mathcal{O}(n^2)$ space. After the failure

of any edge, it can report an (s,t)-mincut C and its capacity in $\mathcal{O}(|C|)$ and $\mathcal{O}(1)$ time, respectively. Recently, Baswana and Bhanja [3] complemented this result by showing that $\Omega(n^2 \log n)$ bits of space is required in the worst case, irrespective of the query time.

For Steiner mincuts, it follows from the above discussion that the existing Sensitivity Oracles are either for undirected unweighted graphs or only for a special case, when $|S| = 2$, in weighted graphs. Therefore, to provide a generalization of these results to any Steiner set, the following is an important question to raise.

Does there exist a single edge Sensitivity Oracle for S-mincut in undirected weighted graphs?

We show that the approach taken by Ausiello et al. [2] can be generalized from $S = \{s,t\}$ to any set $S \subseteq V$. This answers the above-mentioned question in the affirmative and leads to the following result.

▶ **Theorem 3.** *For any undirected weighted graph G on $n = |V|$ vertices, for every Steiner set S, there exists an $\mathcal{O}(n^2)$ space data structure that, after the failure of any edge in G, can report an S-mincut C and its capacity in $\mathcal{O}(|C|)$ time and $\mathcal{O}(1)$ time respectively.*

The space and query time of the Sensitivity Oracle in Theorem 3 match with the existing optimal results for (s,t)-mincut [12, 2, 3]. The lower bound of $\Omega(n^2 \log n)$ bits of space in [3] is only for $|S| = 2$. To the best of our knowledge, no lower bound is known for any $|S| > 2$. Therefore, the main question that we address in this article arises quite naturally as follows.

▶ **Question 1.** *For undirected weighted graphs, does there exist a single edge Sensitivity Oracle for S-mincut that breaks the quadratic bound on space and still achieves optimal query time if $|S| > 2$?*

1.1 Our Results

A Sensitivity Oracle in a weighted graph addresses queries in a more generic way [3]. Given any edge e and any value Δ satisfying $\Delta \geq 0$, the aim is to efficiently report the solution of a given problem after reducing the capacity of edge e by Δ. In this generic setting, using the well-known Gomory and Hu Tree data structure [20], we design the first single edge Sensitivity Oracle for global mincut in weighted graphs that achieves optimal query time.

▶ **Theorem 4** (Sensitivity Oracle for Global Mincut)**.** *For any undirected weighted graph $G = (V,E)$ on $n = |V|$ vertices, there is an $\mathcal{O}(n)$ space data structure that, given any edge e in G and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report the capacity of global mincut in $\mathcal{O}(1)$ time and a global mincut C in $\mathcal{O}(|C|)$ time after reducing the capacity of edge e by Δ.*

The result in Theorem 4 matches the bounds on both space and query time with the best-known single edge Sensitivity Oracles for global mincut in unweighted graphs [15].

Now, in order to bridge the gap between the two extreme scenarios of Steiner set ($|S| = n$ and $|S| = 2$) while matching their bounds, we present our main result that breaks the $\mathcal{O}(n^2)$ space bound of Theorem 3, and answers Question 1 in the affirmative.

▶ **Theorem 5** (Sensitivity Oracle for Steiner Mincut)**.** *Let $G = (V,E)$ be an undirected weighted graph on $n = |V|$ vertices and $m = |E|$ edges. For any Steiner set S of G,*

1. *there is an $\mathcal{O}(n)$ space rooted tree $\mathcal{T}(G)$ that, given any edge $e \in E$ and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report the capacity of S-mincut in $\mathcal{O}(1)$ time after reducing the capacity of edge e by Δ and*

2. *there is an* $\mathcal{O}(n(n-|S|+1))$ *space data structure* $\mathcal{F}(G)$ *that, given any edge* $e \in E$ *and any value* Δ *satisfying* $0 \leq \Delta \leq w(e)$*, can report an* S*-mincut* C *in* $\mathcal{O}(|C|)$ *time after reducing the capacity of edge* e *by* Δ*.*

For any $\epsilon \in [0,1)$, the space occupied by the single edge Sensitivity Oracle for S-mincut in Theorem 5(2) is subquadratic, that is $\mathcal{O}(n^{1+\epsilon})$, for $|S| = n - n^{\epsilon} + 1$. Moreover, it approaches to $\mathcal{O}(n)$ as $|S|$ tends to n. In particular, for $|S| = n - k$, for any constant $k \geq 0$, it occupies only $\mathcal{O}(n)$ space.

Observe that our results in Theorem 5 interestingly match the bounds on both space and query time for the two extreme scenarios of the Steiner set. On one extreme ($|S| = n$), it occupies $\mathcal{O}(n)$ space for global mincut. On the other extreme ($|S| = 2$), it occupies $\mathcal{O}(n^2)$ space, which match the best-known existing results for (s,t)-mincut [2, 12, 3]. Finally, the time taken by our Sensitivity Oracle to answer any query is also worst-case optimal.

We also provide lower bounds on both space and query time of Sensitivity Oracles for S-mincut. Our first lower bound is for reporting the capacity of S-mincut and our second lower bound is for reporting an S-mincut.

▶ **Theorem 6** (Lower Bound for Reporting Capacity)**.** *Let* D *be any data structure that can report the capacity of Steiner mincut after the failure of any edge for undirected weighted graphs on* n *vertices. Data structure* D *must occupy* $\Omega(n^2 \log n)$ *bits of space in the worst case, irrespective of the query time and the size of the Steiner set.*

For reporting the capacity of S-mincut, Theorem 6 provides a generalization of the existing lower bound on both space and time for (s,t)-mincut by Baswana and Bhanja [3]. However, for reporting an S-mincut, no lower bound on space or query time for single edge Sensitivity Oracle was known till date, even for the two extreme scenarios of Steiner set. So, the following theorem is the first lower bound for reporting an S-mincut after the failure of any edge.

▶ **Theorem 7** (Lower Bound for Reporting Cut)**.** *Let* D *be any data structure that can report a Steiner mincut* C *in* $\mathcal{O}(|C|)$ *time after the failure of any edge for undirected weighted graphs on* n *vertices. Data structure* D *must occupy* $\Omega(n^2)$ *bits of space in the worst case, irrespective of the size of the Steiner set.*

▶ Remark 8. It is assumed in Theorem 5 that the query edge e is present in G and the change in capacity (that is, Δ) provided with the query is at most $w(e)$. So, the lower bounds of $\Omega(n^2)$ bits of space in Theorem 6 and Theorem 7 do not violate the sub-quadratic space data structures in Theorem 5. Moreover, the assumption in Theorem 5 seems practically justified. This is because, as discussed in [3], in the real world, the capacity of an edge reduces only if the edge actually exists in the graph, and furthermore, it can reduce by a value at most the capacity of the edge.

1.2 Related Works

In the seminal works by Dinitz and Vainshtein [18, 16, 19], they designed an $\mathcal{O}(\min\{n\lambda_S, m\})$ space data structure, known as *Connectivity Carcass*, for storing all S-mincuts of an unweighted undirected graph. It can report an S-mincut in $\mathcal{O}(m)$ time and its capacity in $\mathcal{O}(1)$ time. Baswana and Pandey [8], using Connectivity Carcass as the foundation, designed an $\mathcal{O}(n)$ space single edge Sensitivity Oracle for S-mincut in undirected unweighted graphs that also reports an S-mincut in $\mathcal{O}(n)$ time. Their result matches the bounds on both space and time for the existing result on the two extreme scenarios of S-mincut, namely, (s,t)-mincut [26] and global mincut [15]. The result on S-mincut in [8] also acts as the foundation of single edge Sensitivity Oracles for all-pairs mincut [8]

For directed weighted graphs, Baswana and Bhanja [3] presented a single edge Sensitivity Oracle for (s,t)-mincut that matches both space and query time of the undirected weighted graph results [12, 2].
Providing a generalization from the two extreme scenarios of the Steiner set ($|S| = n$ and $|S| = 2$) is also addressed for various problems, namely, computing Steiner mincut [18, 19, 14, 21, 23], Steiner connectivity augmentation and splitting-off [11], construction of a cactus graph for Steiner mincuts [19, 21].

1.3 Organization of the Article

This article is organized as follows. Section 2 contains the basic preliminaries. We first construct an $\mathcal{O}(n^2)$ space single edge Sensitivity Oracle for Steiner mincut in Section 3. In Section 4, we design an $\mathcal{O}(n)$ space single edge Sensitivity Oracle for reporting only the capacity of Steiner mincut. A linear space single edge Sensitivity Oracle for global mincut is designed in Section 5. Our main result on the subquadratic space single edge Sensitivity Oracle for Steiner mincut is developed in Section 6. Finally, we conclude in Section 7. The proofs of the lower bounds are provided in the full version of this article.

2 Preliminaries

In this section, we define a set of basic notations and properties of cuts. Let $G \setminus \{e\}$ denote the graph obtained from G after the removal of edge e. We now define the concept of crossing cuts, introduced by Dinitz, Karzanov, and Lomonosov [15].

▶ **Definition 9** (crossing cuts). *A pair of cuts C and C' in G is said to be crossing if each of the four sets $C \cap C'$, $C \setminus C'$, $C' \setminus C$, and $\overline{C \cup C'}$ is nonempty.*

The following concept of mincut for an edge and vital edges are to be used crucially in the construction of our data structure.

▶ **Definition 10** (Mincut for an edge). *A Steiner cut C is said to be a mincut for an edge e if e contributes to C and $c(C) \leq c(C')$ for every Steiner cut C' in which e contributes.*

▶ **Definition 11** (Vital Edge). *Let e be an edge and C be a mincut for edge e. Edge e is said to be a vital edge if its removal reduces the capacity of Steiner mincut, that is, $c(C) - w(e) < \lambda_S$.*

We now define a *special* mincut for an edge.

▶ **Definition 12** (Nearest mincut for an edge). *A mincut C for an edge $e = (x, y) \in E$ with $x \in C$ is said to be a nearest mincut for e if there is no mincut C' for e such that $x \in C'$ and $C' \subset C$. The set of all nearest mincuts for an edge e is denoted by $N(e)$.*

▶ **Lemma 13** (Sub-modularity of Cuts (Problem 48(a,b) in [24])). *For any two sets $A, B \subset V$,*

1. $c(A) + c(B) \geq c(A \cap B) + c(A \cup B)$ *and*
2. $c(A) + c(B) \geq c(A \setminus B) + c(B \setminus A)$.

▶ **Definition 14** (Laminar family of cuts). *A set of cuts $\mathcal{L}$ is said to form a laminar family if, for any pair of cuts $C_1, C_2 \in \mathcal{L}$, exactly one of the three is true – $C_1 \cap C_2$ is an empty set, $C_1 \subseteq C_2$, and $C_2 \subseteq C_1$.*

A rooted tree $\mathcal{T}_{\mathcal{L}}$ representing a laminar family $\mathcal{L}$. For any given laminar family $\mathcal{L}$ of cuts in G, we can construct an $\mathcal{O}(n)$ space rooted tree $\mathcal{T}_{\mathcal{L}}$ that stores every cut belonging to $\mathcal{L}$ as follows. Every vertex x of G is mapped to a unique node in $\mathcal{T}_{\mathcal{L}}$, denoted by $\phi_{\mathcal{L}}(x)$. Every node μ in $\mathcal{T}_{\mathcal{L}}$ represents a unique cut C in $\mathcal{L}$ as follows. Cut C is the set of vertices mapped to the subtree rooted at μ (including μ). For any pair of nodes μ and ν in $\mathcal{T}_{\mathcal{L}}$, μ is a child of ν if and only if the cut represented by μ is a maximal proper subset of the cut represented by ν. The minimal cuts of $\mathcal{L}$ are represented by leaf nodes of $\mathcal{T}_{\mathcal{L}}$. For any vertex x in G, let SubTree(x) denote the set of all vertices mapped to the subtree rooted at $\phi_{\mathcal{L}}(x)$ (including $\phi_{\mathcal{L}}(x)$) in $\mathcal{T}_{\mathcal{L}}$. This leads to the following lemma.

▶ **Lemma 15.** *For any laminar family $\mathcal{L}$ of cuts in G, there exists an $\mathcal{O}(n)$ space rooted tree $\mathcal{T}_{\mathcal{L}}$ such that a cut $C \in \mathcal{L}$ if and only if there exists a node μ (except root node) of $\mathcal{T}_{\mathcal{L}}$ and C is the set of vertices mapped to the subtree rooted at μ (including node μ).*

3 An $\mathcal{O}(n^2)$ Space Sensitivity Oracle for Steiner Mincut

In this section, we first provide the limitations of the previous results in unweighted graphs. Later, we design an $\mathcal{O}(n^2)$ space single edge Sensitivity Oracle for S-mincut.

Limitations of the existing results. For unweighted graphs, the following property is used crucially to design every existing single edge Sensitivity Oracle.
Property P_1: *Failure of an edge e reduces the capacity of S-mincut if and only if edge e contributes to an S-mincut.*
Dinitz and Vainshtein [18, 16, 19] designed the following quotient graph, known as the flesh graph, of G. Flesh graph is obtained by contracting every pair of vertices in G that are not separated by any S-mincut. The construction ensures that every pair of vertices in flesh is separated by an S-mincut of G. Every vertex in G is mapped to a unique vertex in flesh. Therefore, the endpoints of any edge e are mapped to different vertices in flesh if and only if failure of e reduces capacity of S-mincut. Thus, by Property P_1, storing the $\mathcal{O}(n)$ space mapping of vertices of G to the vertices of flesh is sufficient to answer the capacity of S-mincut in $\mathcal{O}(1)$ time after the failure of any edge. In addition, flesh graph can be used to report an S-mincut after the failure of any edge in G in $\mathcal{O}(m)$ time.
Flesh graph is one of the three components of Connectivity Carcass designed by Dinitz and Vainshtein [18, 16, 19]; the other two components are *Skeleton* and *Projection mapping*. Recently, Baswana and Pandey [8], exploiting the properties of all the three components of Connectivity Carcass established an *ordering* among the vertices of flesh graph. By using Property P_1, they showed that this ordering, along with Skeleton and Projection mapping, can be used to design an $\mathcal{O}(n)$ space single edge Sensitivity Oracle for S-mincut in unweighted graphs. This single edge Sensitivity Oracle can report an S-mincut in $\mathcal{O}(n)$ time.

Unfortunately, for weighted graphs, it is easy to observe that multiple edges can exist that do not contribute to any S-mincut but failure of each of them reduces the capacity of S-mincut. Hence, in weighted graphs, Property P_1 no longer holds. So, the existing structures are not suitable for handling the failure of weighted edges. It requires us to explore the structure of mincuts for every edge whose both endpoints belong to the same node of flesh graph. Moreover, the capacity of mincut for these edges can be quite *large* compared to the capacity of S-mincut.

Sensitivity Oracle for Steiner Mincut: $\mathcal{O}(n^2)$ Space

We now give a proof of Theorem 3 by designing an $\mathcal{O}(n^2)$ space single edge Sensitivity Oracle for S-mincut. Let F be any arbitrary real-valued function defined on cuts. Cheng and Hu [12] presented the following result. There is an $\mathcal{O}(n^2)$ space data structure, known as Ancestor tree, that, given any pair of vertices u and v, reports a cut C of the least capacity (F-value) separating u, v in $\mathcal{O}(|C|)$ time and the capacity of C in $\mathcal{O}(1)$ time.

In order to design Ancestor tree for Steiner cuts, similar to (s,t)-mincuts given by Ausiello et al. [2], we define function F for Steiner cuts as follows.

$$\text{For a set } C \subset V,\ F(C) = \begin{cases} c(C), & \text{if } C \text{ is a Steiner cut} \\ \infty, & \text{otherwise.} \end{cases} \tag{1}$$

Let $e = (x, y)$ be any failed edge. Ancestor tree can report a cut C of the least capacity separating x and y in $\mathcal{O}(|C|)$ time and its capacity in $\mathcal{O}(1)$ time. By Equation 1, C is also a Steiner cut separating x and y. Therefore, by Definition 10, C is a mincut for edge e. Hence, the new capacity of S-mincut is either $c(C) - w(e)$ or remains λ_S if $c(C) - w(e) \geq \lambda_S$. By storing the capacities of all edges of G, we can determine whether $c(C) - w(e) < \lambda_S$ in $\mathcal{O}(1)$ time. If the capacity of S-mincut reduces, then we can report C in $\mathcal{O}(|C|)$ time; otherwise report an S-mincut C_m in $\mathcal{O}(|C_m|)$ time. This completes the proof of Theorem 3.

4 A Sensitivity Oracle for Reporting Capacity of Steiner Mincut

In this section, we address the problem of reporting the capacity of S-mincut after reducing the capacity of an edge $e \in E$ by a value Δ satisfying $0 \leq \Delta \leq w(e)$. We denote this query by $\textsc{cap}(e, \Delta)$. Observe that a trivial data structure for answering query CAP occupies $\mathcal{O}(m)$ space if we store the capacity of mincut for each vital edge in G. For $|S| = 2$ in directed weighted graphs, Baswana and Bhanja [3] designed an $\mathcal{O}(n)$ space data structure that implicitly stores all vital edges for (s,t)-mincut and the capacity of their mincuts. We extend their approach from vital edges to any set of edges in undirected weighted graphs in order to establish the following. For any Steiner set S, with $2 \leq |S| \leq n$, there exists an $\mathcal{O}(n)$ space data structure that can answer query CAP in $\mathcal{O}(1)$ time.

Let $\mathcal{E} \subseteq E$ and $V(\mathcal{E})$ denote the smallest set of vertices such that, for each edge $(u, v) \in \mathcal{E}$, both u and v belongs to $V(\mathcal{E})$. We first design an $\mathcal{O}(|V(\mathcal{E})|)$ space rooted full binary tree for answering query CAP for all edges in $\mathcal{E}$. In Section 6, this construction also helps in designing a compact structure for reporting mincuts for a *special* subset of edges. Later in this section, we show an extension to $\mathcal{O}(n)$ space rooted full binary tree for answering query CAP for all edges in E.

Let $C(e)$ denote a mincut for an edge e. Note that $C(e)$ is a Steiner cut as well (Definition 10). We say that an edge e belongs to a set $U \subset V$ if both endpoints of e belong to U. Suppose $C(e)$ is a mincut for an edge e belonging to $V(\mathcal{E})$ such that, for every other edge $e' \in V(\mathcal{E})$, $c(C(e)) \leq c(C(e'))$. Let e' be an edge from $V(\mathcal{E})$. If e' contributes to $C(e)$, it follows from the selection of edge e that $C(e)$ is a Steiner cut of the least capacity to which e' contributes. Hence, $C(e)$ is a mincut for edge e' as well. This ensures that $C(e)$ partitions the set of all edges belonging to $V(\mathcal{E})$ into three sets – edges of $V(\mathcal{E})$ belonging to $C(e) \cap V(\mathcal{E})$, edges of $V(\mathcal{E})$ belonging to $\overline{C(e)} \cap V(\mathcal{E})$, and edges of $V(\mathcal{E})$ that contribute to $C(e)$. This leads to a recursive procedure (Algorithm 1) for the construction of a tree $\mathcal{T}$. Each internal node μ of tree $\mathcal{T}$ has three fields – (i) μ.cap stores the capacity of mincut for the selected edge at μ, (ii) μ.LEFT points to the left child of μ, and (iii) μ.RIGHT points to the right child of μ. Each vertex $u \in V(\mathcal{E})$ is mapped to a leaf node of $\mathcal{T}$, denoted by $\mathbb{L}(u)$. We invoke Algorithm 1 with $U = V(\mathcal{E})$.

Algorithm 1 Construction of Tree $\mathcal{T}$.

```
1: procedure SteinerTreeConstruction(U)
2:     Create a node ν;
3:     For any set U ⊆ V, let E(U) denote the edges whose both endpoints belong to U;
4:     if there is no edge in E(U) then
5:         for each vertex x ∈ U do 𝕃(x) ← ν;
6:         end for
7:     else
8:         Select an edge e ∈ E(U) such that c(C(e)) ≤ c(C(e')), ∀ edge e' ∈ E(U);
9:         Assign ν.cap ← c(C(e));
10:        ν.left ← SteinerTreeConstruction(U ∩ C(e));
11:        ν.right ← SteinerTreeConstruction(U ∩ C̄(e));
12:    end if
13:    return ν;
14: end procedure
```

Observe that tree $\mathcal{T}$ resulting from Algorithm 1 is a full binary tree. There are $\mathcal{O}(|V(\mathcal{E})|)$ leaf nodes. So, the space occupied by the tree is $\mathcal{O}(|V(\mathcal{E})|)$.

Answering Query cap$(e = (x, y), \Delta)$. Suppose edge e belongs to $\mathcal{E}$. Let μ be the lowest common ancestor (LCA) of $\mathbb{L}(x)$ and $\mathbb{L}(y)$ in $\mathcal{T}$. It follows from the construction of tree $\mathcal{T}$ that field μ.cap at node μ in $\mathcal{T}$ stores the capacity of mincut for edge e. Therefore, if $\mu.\text{cap} - \Delta < \lambda_S$, then we report μ.cap as the new capacity of S-mincut; otherwise, the capacity of S-mincut does not change. It leads to the following lemma.

▶ **Lemma 16.** *Let $G = (V, E)$ be an undirected weighted graph on $n = |V|$ vertices. For any Steiner set $S \subseteq V$ and a set of edges $\mathcal{E} \subseteq E$, there is an $\mathcal{O}(|V(\mathcal{E})|)$ space full binary tree $\mathcal{T}_{\mathcal{E}}$ that, given any edge $e \in \mathcal{E}$ and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report the capacity of S-mincut in $\mathcal{O}(1)$ time after reducing the capacity of edge e by Δ.*

We now answer query CAP(e, Δ) where edge $e \in E$. Observe that edge e in query CAP can be either a vital or a nonvital edge. In order to determine whether an edge is vital or not, we design a full binary tree $\mathcal{T}_E$ by invoking Algorithm 1 with $U = V$ since $\mathcal{E} = E$. Let us denote the tree by $\mathcal{T}(G)$. By Lemma 16, the size of tree $\mathcal{T}(G)$ is $\mathcal{O}(n)$. It is now easy to observe that an edge e is a vital edge in graph G if and only if the capacity of the Steiner mincut in graph $G \setminus \{e\}$ is $\mu.\text{cap} - w(e) < \lambda_S$, where node μ is the LCA$(\mathbb{L}(x), \mathbb{L}(y))$. This leads to the following lemma.

▶ **Lemma 17.** *Let $G = (V, E)$ be an undirected weighted graph on $n = |V|$ vertices. For any Steiner set $S \subseteq V$, there is an $\mathcal{O}(n)$ space full binary tree $\mathcal{T}(G)$ that, given any edge $e \in E$ and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report the capacity of S-mincut in $\mathcal{O}(1)$ time after reducing the capacity of edge e by Δ.*

Lemma 17 completes the proof of Theorem 5(1).

▶ Remark 18. Ancestor tree of Cheng and Hu [12] can also be used to design an $\mathcal{O}(n)$ space data structure for answering query CAP in $\mathcal{O}(1)$ time. However, Ancestor tree alone does not seem sufficient to establish Lemma 16, which is used crucially to achieve the subquadratic space single edge Sensitivity Oracle for S-mincut stated in Theorem 5.

5 An $\mathcal{O}(n)$ Space Sensitivity Oracle for Global Mincut

The well-known (s,t)-mincut is one extreme scenario of S-mincut when $|S| = 2$. In weighted graphs, designing a single edge Sensitivity Oracle for (s,t)-mincut has been addressed quite extensively [2, 12, 3]. Moreover, each of them occupies $\mathcal{O}(n^2)$ space. However, to this day, no nontrivial single edge Sensitivity Oracle exists for global mincut, which is the other extreme scenario of S-mincut when $|S| = n$. We now present the first single edge Sensitivity Oracle for global mincut that occupies only $\mathcal{O}(n)$ space and achieves optimal query time.

Let λ_V be the capacity of global mincut. Given any edge e, we want to determine the capacity of mincut for edge e for Steiner set $S = V$. Observe that, for $S = V$, every cut in the graph is a Steiner cut (or global cut). Exploiting this insight, we can state the following interesting relation between global mincut and all-pairs mincuts (or (u,v)-mincut, for every $u, v \in V$).

▶ **Lemma 19.** *For an edge (u,v), C is a cut of the least capacity that separates u, v if and only if C is a mincut for edge (u,v).*

Gomory and Hu [20] designed the following tree data structure for all-pairs mincuts, which is widely known as Gomory Hu Tree.

▶ **Theorem 20** (Gomory Hu Tree [20])**.** *For any undirected weighted graph $G = (V, E)$ on $n = |V|$ vertices, there is an $\mathcal{O}(n)$ space undirected weighted tree $\mathcal{T}_{GH}$ on vertex set V that satisfies the following property. Let u, v be any pair of vertices in G. A cut of the least capacity separating u, v in $\mathcal{T}_{GH}$ is also a cut of the least capacity separating u, v in G. Moreover, $\mathcal{T}_{GH}$ can report a cut C of the least capacity separating u, v in $\mathcal{O}(|C|)$ time and its capacity in $\mathcal{O}(1)$ time.*

Let $\mathcal{T}_{GH}$ be a Gomory Hu Tree of G. By Theorem 20, for every pair of vertices u, v in G, $\mathcal{T}_{GH}$ stores a cut of the least capacity separating u, v. By Lemma 19, it follows that, for $S = V$, $\mathcal{T}_{GH}$ stores a mincut for every edge in G. Hence, it acts as a single edge Sensitivity Oracle for global mincut and can report a mincut C for any given edge e in $\mathcal{O}(|C|)$ time and its capacity in $\mathcal{O}(1)$ time. Therefore, after reducing $w(e)$ by a value Δ, if $c(C) - \Delta < \lambda_V$, we can report a global mincut and its capacity optimally using $\mathcal{O}(n)$ space. This establishes the results in Theorem 4.

Now, for both extreme scenarios of Steiner mincuts, we have a single edge Sensitivity Oracle. Interestingly, the Sensitivity Oracle for global mincut achieves better than quadratic space. The question that arises is how to generalize these results to any Steiner set.

6 A Sensitivity Oracle for Steiner Mincut: Breaking Quadratic Bound

In this section, we address the problem of reporting an S-mincut after reducing the capacity of any given edge $e \in E$ by any given value Δ satisfying $0 < \Delta \leq w(e)$. We denote this query by $\textsc{cut}(e, \Delta)$. Our objective is to design a data structure that breaks $\mathcal{O}(n^2)$ bound on space for efficiently answering query CUT. A simple data structure can be designed by augmenting tree $\mathcal{T}(G)$ in Theorem 5(1) as follows. For each internal node μ of tree $\mathcal{T}(G)$, Algorithm 1 selects an edge e in Step 7 and stores the capacity of mincut $C(e)$ for edge e in μ.cap. Observe that if we augment node μ with $C(e)$, then it helps in answering query CUT as well. However, the augmented tree occupies $\mathcal{O}(n^2)$ space, which defeats our objective.

For global mincut ($S = V$), observe that Gomory Hu Tree essentially acts as a data structure that stores at least one mincut for every edge quite compactly. To design a more compact data structure for answering query CUT for S-mincut compared to Theorem 3, we take an approach of designing a data structure that can compactly store at least one mincut for every edge.

We begin by a *classification* of all edges of graph G. This classification not only helps in combining the approaches taken for (s,t)-mincut and global mincut but also provides a way to design a compact data structure for efficiently answering query CUT for any Steiner set S.

A Classification of All Edges. An edge e_1 in G belongs to
- Type-1 if both endpoints of e_1 belong to $V \setminus S$.
- Type-2 if both endpoints of e_1 belong to S.
- Type-3 if exactly one endpoint of e_1 belongs to S.

Given any edge e_1, we can classify e_1 into one of the above-mentioned three types in $\mathcal{O}(1)$ time using sets V and S. We can take care of edges from Type-1 by extending an approach for (s,t)-mincut given by Baswana and Bhanja [3]. Similarly, we can handle edges from Type-2 by extending the approach used for global mincut (Theorem 4). However, the **main challenge** arises in designing a data structure for compactly storing a mincut for all edges from Type-3. We now design a compact data structure for efficiently answering query CUT for each type of edges separately.

6.1 An $\mathcal{O}((n-|S|)n)$ Space Data Structure for All Edges from Type-1

We aim to design a data structure for answering query CUT for all edges from Type-1. Each edge from Type-1 has both endpoints in set $V \setminus S$. The number of vertices in $V \setminus S$ is $n-|S|$. Therefore, trivially storing a mincut for every edge would occupy $\mathcal{O}((n-|S|)^2 n)$ space, which is $\mathcal{O}(n^3)$ for $|S| = k$ for any constant $k \geq 2$. Exploiting the fact that the number of distinct endpoints of all edges from Type-1 is at most $n-|S|$, we design an $\mathcal{O}(n(n-|S|))$ space data structure for all edges from Type-1 using Algorithm 1 and Lemma 16 as follows.

Let E_1 be the set of all edges from Type-1. It follows from Lemma 16 that, using Algorithm 1, it is possible to design a rooted full binary tree $\mathcal{T}_{E_1}$ occupying $\mathcal{O}(n-|S|)$ space for answering query CAP(e,Δ) when edge e is from Type-1. We augment each internal node μ of $\mathcal{T}_{E_1}$ with a mincut for the edge selected by Algorithm 1 (in Step 7) while processing node μ. The resulting structure occupies $\mathcal{O}((n-|S|)n)$ space and acts as a data structure for answering query CUT for all edges from Type-1. Hence the following lemma holds.

▶ **Lemma 21** (Sensitivity Oracle for Type-1 Edges). *For any Steiner set $S \subseteq V$, there is an $\mathcal{O}((n-|S|)n)$ space data structure that, given any edge e from Type-1 and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report an S-mincut C in $\mathcal{O}(|C|)$ time after reducing the capacity of edge e by Δ.*

6.2 An $\mathcal{O}(n)$ Space Data Structure for All Edges from Type-2

In this section, we design a data structure for answering query CUT for all edges from Type-2. For each edge from Type-2, both endpoints are Steiner vertices. So, the number of distinct endpoints of edges from Type-2 can be at most $|S|$. Trivially, storing a mincut for every edge from Type-2 would occupy $\mathcal{O}(|S|^2 n)$ space, which is $\mathcal{O}(n^3)$ if $|S| = \mathcal{O}(n)$. In a similar way as designing $\mathcal{T}_{E_1}$ for all edges from Type-1 (Lemma 21), by using Lemma 16 and Algorithm 1, it is possible to design an $\mathcal{O}(|S|n)$ space data structure for answering query CUT for all edges

from Type-2. Unfortunately, it defeats our objective because, for $|S| = n$ or global mincuts, it occupies $\mathcal{O}(n^2)$ space. Interestingly, by exploiting the fact that both the endpoints of every edge from Type-2 are Steiner vertices, we are able to show that a Gomory Hu Tree of graph G is sufficient for answering query cut for all edges from Type-2.

▶ **Lemma 22.** *Let $\mathcal{T}_{GH}$ be a Gomory Hu Tree of G. Then, for any edge $e = (u, v)$ from Type-2, a cut of the least capacity separating u and v in $\mathcal{T}_{GH}$ is a mincut for edge e in G.*

Proof. Let $\mathcal{T}_{GH}$ be a Gomory Hu Tree of G. By Theorem 20, $\mathcal{T}_{GH}$ stores a cut of the least capacity separating every pair of vertices in G. Let (u, v) be any edge from Type-2. Suppose C is a cut of the least capacity separating u and v in $\mathcal{T}_{GH}$. So, edge (u, v) is contributing to C in G. By definition of Type-2 edges, both u and v are Steiner vertices. Therefore, C is a Steiner cut of G in which edge (u, v) is contributing. It follows from Theorem 20 that C is also a cut of the least capacity in G that separates u and v. So, C is also a Steiner cut of the least capacity in which edge (u, v) is contributing in G. Hence, C is a mincut for (u, v). ◀

Let $e = (u, v)$ be any edge from Type-2 and $\mathcal{T}_{GH}$ be a Gomory Hu Tree of G. By Theorem 20, $\mathcal{T}_{GH}$ can report in $\mathcal{O}(|C|)$ time a cut C of the least capacity in $\mathcal{T}_{GH}$ that separates u and v. By Lemma 22, C is a mincut for edge e in G. This establishes the following lemma.

▶ **Lemma 23** (Sensitivity Oracle for Type-2 Edges)**.** *For any Steiner set $S \subseteq V$, there is an $\mathcal{O}(n)$ space data structure that, given any edge e from Type-2 and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report an S-mincut C in $\mathcal{O}(|C|)$ time after reducing the capacity of edge e by Δ.*

For global mincut or $S = V$, both endpoints of every edge are Steiner vertices. Therefore, Theorem 4 can also be seen as a corollary of Lemma 23.

6.3 An $\mathcal{O}((n - |S|)n)$ Space Data Structure for All Edges from Type-3

The objective is to design a data structure for answering query cut for all edges from Type-3. Observe that the size of the smallest set of vertices that contains all the endpoints of all edges from Type-3 can be $\Omega(n)$. Therefore, using Lemma 16 and Algorithm 1, we can have an $\mathcal{O}(n^2)$ space data structure, which is no way better than the trivial data structure for answering query cut (Theorem 3). Now, each edge from Type-3 has exactly one nonSteiner endpoint. So, unlike edges from Type-2, Lemma 22 no longer holds for edges from Type-3. This shows the limitations of the approaches taken so far in designing a data structure for answering query cut. Trivially storing a mincut for every edge from Type-3 requires $\mathcal{O}((n - |S|)n|S|)$ space. For $|S| = \frac{n}{k}$, any constant $k \geq 2$, it occupies $\mathcal{O}(n^3)$ space. Interestingly, we present a data structure occupying only $\mathcal{O}((n - |S|)n)$ space for answering query cut for all edges from Type-3.

For any edge $e_1 = (x, u)$ from Type-3 with $x \in S$, without loss of generality, we assume that any mincut C for edge e_1 contains the Steiner vertex x, otherwise consider $\overline{C}$. Note that the set of global mincuts and (s, t)-mincuts are closed under both intersection and union. This property was crucially exploited in designing a compact structure for storing them [15, 26]. To design a compact structure for storing a mincut for every edge from Type-3, we also explore the relation between a pair of mincuts for edges from Type-3. Let A and B be mincuts for edges e_1 and e_2 from Type-3, respectively. Unfortunately, it turns out that if A crosses B, then it is quite possible that neither $A \cap B$ nor $A \cup B$ is a mincut for e_1 or e_2 even if both are Steiner cuts (refer to Figure 1(i)). This shows that mincuts for edges from Type-3 are not closed under intersection or union. To overcome this hurdle, we first present a partitioning of the set of edges from Type-3 based on the nonSteiner vertices as follows.

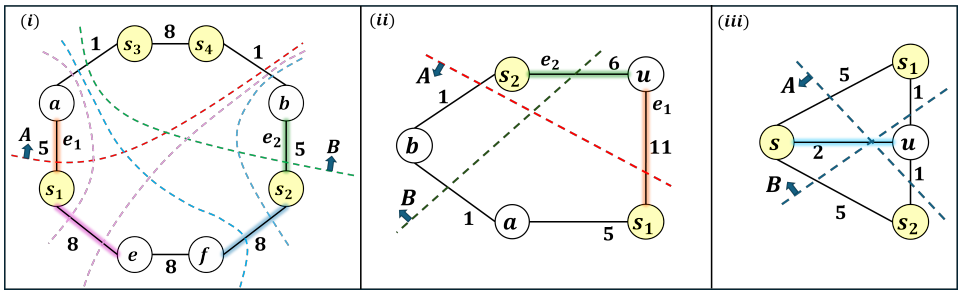

Figure 1 Yellow vertices are Steiner vertices. A mincut for an edge is represented by the same color. (i) Mincuts A, B are for edges $e_1 = (s_1, a)$ and $e_2 = (s_2, b)$. Observe that $A \cap B$ and $A \cup B$ are Steiner cuts but not mincuts for edges e_1, e_2. Moreover, cuts $A \setminus B$ and $B \setminus A$, in which edges e_1 and e_2 are contributing, are not even Steiner cuts. (ii) Edges e_1 and e_2 are vital and from Type-3(u). Mincuts A and B for edges e_1 and e_2 are crossing, but $A \cap B$ and $A \cup B$ are not Steiner cuts. (iii) Edge (s, u) is from Type-3 and $N((s, u)) = \{A, B\}$.

Let $V' \subseteq V \setminus S$ be the smallest set of nonSteiner vertices such that every edge from Type-3 has endpoint in V'. Let u be any vertex from V'. Let Type-3(u) be the set that contains all edges from Type-3 having u as one of the two endpoints. We aim to design an $\mathcal{O}(n)$ space data structure that can report a mincut for each edge from Type-3(u). This is because storing an $\mathcal{O}(n)$ space data structure for every nonSteiner vertex of G would lead to an $\mathcal{O}((n - |S|)n)$ space data structure.

In order to design an $\mathcal{O}(n)$ space data structure for edges from Type-3(u), we consider the set of nearest mincuts (Definition 12) for edges from Type-3(u). The following lemma provides a strong reason behind the use of nearest mincuts for edges from Type-3(u).

▶ **Lemma 24** (Disjoint Property). *Let $C \in N(e_1)$ and $C' \in N(e_2)$ such that $e_1 = (x, u)$, $e_2 = (x', u)$ are edges from Type-3(u). Then, $x' \notin C$ and $x \notin C'$ if and only if $C \cap C' = \emptyset$.*

Proof. Suppose $x' \notin C$ and $x \notin C'$. Assume to the contrary that $C \cap C' \neq \emptyset$. Observe that $x \in C \setminus C'$ and $x' \in C' \setminus C$. As a result, $C \setminus C'$ as well as $C' \setminus C$ is a Steiner cut. It is given that C is the nearest mincut for edge (x, u) and $x \in C \setminus C'$. This implies that $c(C \setminus C') > c(C)$. It follows from sub-modularity of cuts (Lemma 13(2)) that $c(C' \setminus C) < c(C')$. Therefore, we get a Steiner cut $C' \setminus C$ of capacity strictly less than $c(C')$ and edge (x', u) is a contributing edge of Steiner cut $C' \setminus C$, a contradiction.

The proof of the converse part is immediate. ◀

Let $e_1 = (x, u)$ and $e_2 = (x', u)$ be any pair of edges from Type-3(u). Let C be a nearest mincut for e_1 and C' be a nearest mincut for e_2. Lemma 24 essentially states that if e_2 contributes to $C' \setminus C$ and e_1 contributes to $C \setminus C'$, then C must not cross C'. Now, the problem arises when one of the two edges e_1, e_2 is contributing to a nearest mincut for the other edge. Firstly, there might exist multiple nearest mincuts for an edge (refer to Figure 1(iii)). It seems quite possible that an edge, say e_2, is contributing to one nearest mincut for e_1 and is not contributing to another nearest mincut for e_1. Secondly, the union of a pair of nearest mincuts for an edge e_1 from Type-3(u) might not even be a Steiner cut if they cross (refer to Figure 1(iii)). Hence, the union of them can have a capacity strictly less than the capacity of mincut for e. So, it seems that the nearest mincuts for edges from Type-3(u) appear quite *arbitrarily*. It might not be possible to have an $\mathcal{O}(n)$ space structure for storing them. Interestingly, we are able to circumvent all the above challenges as follows.

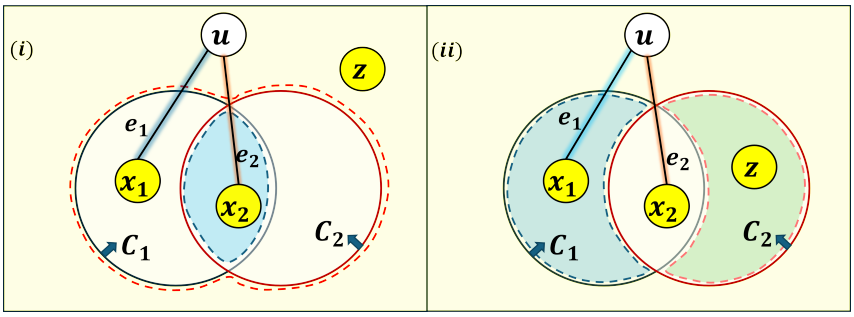

Figure 2 Illustration of the proof of Lemma 25. (i) There is a Steiner vertex z in $\overline{C_1 \cup C_2}$. The red dashed cut shows cut $C_1 \cup C_2$ and the blue dashed cut (blue region) shows cut $C_1 \cap C_2$. (ii) There is a Steiner vertex z in $C_2 \setminus C_1$. Blue dashed cut (blue region) is $C_1 \setminus C_2$. Similarly, red dashed cut (light green region) is $C_2 \setminus C_1$.

Observe that we are interested in only those edges from Type-3(u) whose failure reduces S-mincut. They are the set of all vital edges that belong to Type-3(u), denoted by VitType-3(u). By exploiting *vitality* of edges from VitType-3(u), we establish the following crucial result for any pair of crossing mincuts for edges from VitType-3(u). Interestingly, this result holds even if the union of a pair of mincuts for a pair of edges from VitType-3(u) is not always a Steiner cut (refer to Figure 1(ii)).

▶ **Lemma 25** (Property of Intersection). *Let C_1 and C_2 be mincuts for edges $e_1 = (x_1, u)$ and $e_2 = (x_2, u)$ from VitType-3(u) respectively. Steiner vertex x_2 is present in C_1 if and only if $C_1 \cap C_2$ is a mincut for edge e_2.*

Proof. Suppose Steiner vertex x_2 is present in C_1. Since u is a common endpoint of both edges e_1, e_2, we have $u \notin C_1 \cup C_2$. C_1 is a mincut for edge e_1, so, $x_1 \in C_1$. Now, there are two possibilities – either (1) $x_1 \notin C_2$ or (2) $x_1 \in C_2$. We now establish each case separately.

Case 1. Suppose $x_1 \notin C_2$, in other words, $x_1 \in C_1 \setminus C_2$. It implies that $C_1 \setminus C_2$ is nonempty. Observe that C_2 is not a subset of C_1; otherwise, $C_1 \cap C_2 = C_2$ is a mincut for e_2 and the lemma holds. So, $C_2 \setminus C_1$ is also nonempty. Let $c(C_1) = \lambda_1$ and $c(C_2) = \lambda_2$. Since $x_2 \in C_1 \cap C_2$, edge e_2 is contributing to C_1. Therefore, $c(C_2) \leq c(C_1)$; otherwise, C_2 is not a mincut for edge e_2. Since C_1 is a Steiner cut, there must be a Steiner vertex z such that $z \notin C_1$. Based on the position of z with respect to cut C_2, observe that z appears either (1.1) in $\overline{C_1 \cup C_2}$ or (1.2) in $C_2 \setminus C_1$ (refer to Figure 2).

Case 1.1. Suppose $z \in \overline{C_1 \cup C_2}$ (refer to Figure 2(i)). Observe that $C_1 \cup C_2$ is a Steiner cut in which edge e_1 is contributing. So, the capacity of $C_1 \cup C_2$ has to be at least λ_1. By sub-modularity of cuts (Lemma 13(1)), $c(C_1 \cap C_2) + c(C_1 \cup C_2) \leq \lambda_1 + \lambda_2$. It follows that $c(C_1 \cap C_2) \leq \lambda_2$. Since $C_1 \cap C_2$ is a Steiner cut in which edge e_2 is contributing, therefore, the capacity of $C_1 \cap C_2$ is exactly λ_2. Hence $C_1 \cap C_2$ is a mincut for edge e_2.

Case 1.2. We show that this case does not arise. Assume to the contrary that $z \in C_2 \setminus C_1$ (refer to Figure 2(ii)). Here, we crucially exploit the fact that edge e_2 is a vital edge from Type-3(u). Let us now consider graph $G \setminus \{e_2\}$. Since, in graph G, edge e_2 is a vital edge and $c(C_2) \leq c(C_1)$, therefore, in graph $G \setminus \{e_2\}$, the capacity of S-mincut is $\lambda_2 - w(e_2)$ and C_2 is an S-mincut. In G, edge e_2 is also a contributing edge of C_1. Therefore, the capacity of C_1 in $G \setminus \{e_2\}$ is $\lambda_1 - w(e_2)$. Without causing any ambiguity, let us denote the capacity of any cut A in $G \setminus \{e_2\}$ by $c(A)$. By sub-modularity of cuts (Lemma 13(2)), in graph $G \setminus \{e_2\}$, we

have $c(C_1 \setminus C_2) + c(C_2 \setminus C_1) \leq \lambda_1 + \lambda_2 - 2w(e_2)$. Recall that $x_1 \in C_1 \setminus C_2$ and $z \in C_2 \setminus C_1$. So, both $C_1 \setminus C_2$ and $C_2 \setminus C_1$ are Steiner cuts in graph $G \setminus \{e_2\}$. Therefore, the capacity of $C_2 \setminus C_1$ is at least $\lambda_2 - w(e_2)$. It follows that the capacity of $C_1 \setminus C_2$ in $G \setminus \{e_2\}$ is at most $\lambda_1 - w(e_2)$. We now obtain graph G by adding edge e_2 to graph $G \setminus \{e_2\}$. Observe that edge e_2 does not contribute to Steiner cut $C_1 \setminus C_2$. Therefore, the capacity of cut $C_1 \setminus C_2$ remains the same in graph G, which is at most $\lambda_1 - w(e_2)$. Since e_2 is a vital edge, so, $w(e_2) > 0$. This implies that we have $\lambda_1 - w(e_2) < \lambda_1$. Therefore, for cut $C_1 \setminus C_2$ in G, $C_1 \setminus C_2$ is a Steiner cut and has a capacity that is strictly less than λ_1. Moreover, edge e_1 is contributing to $C_1 \setminus C_2$. So, C_1 is not a mincut for edge e_1, a contradiction.

Case 2. In this case, we have $x_1 \in C_2$. Since C_1 and C_2 both are Steiner cuts, observe that either (2.1) there is at least one Steiner vertex z in $\overline{C_1 \cup C_2}$ or (2.2) there exists a pair of Steiner vertices z_1, z_2 such that $z_1 \in C_1 \setminus C_2$ and $z_2 \in C_2 \setminus C_1$. The proof of case (2.1) is along similar lines to the proof of case (1.1). So, let us consider case (2.2). Edges e_1 and e_2 are contributing to both C_1 and C_2. It implies that $c(C_1) = c(C_2)$. Let $c(C_1)$ (or $c(C_2)$) be λ. Let us consider graph $G \setminus \{e_2\}$. Since e_2 is a vital edge, the capacity of S-mincut is $\lambda - w(e_2)$. The capacity of cuts C_1 and C_2 in $G \setminus \{e_2\}$ is $\lambda - w(e_2)$ since e_2 contributes to both of them. Without causing any ambiguity, let us denote the capacity of any cut A in $G \setminus \{e_2\}$ by $c(A)$. By sub-modularity of cuts (Lemma 13(2)), $c(C_1 \setminus C_2) + c(C_2 \setminus C_1) \leq 2\lambda - 2w(e_2)$. Now, it is given that $z_1 \in C_1 \setminus C_2$ and $z_2 \in C_2 \setminus C_1$. Therefore, in $G \setminus \{e_2\}$, $C_1 \setminus C_2$ and $C_2 \setminus C_1$ are Steiner cuts. It follows that $c(C_1 \setminus C_2)$, as well as $c(C_2 \setminus C_1)$, is exactly $\lambda - w(e_2)$. Let us obtain graph G from $G \setminus \{e_2\}$. Observe that e_2 contributes neither to $C_1 \setminus C_2$ nor to $C_2 \setminus C_1$. Therefore, the capacity of cuts $C_1 \setminus C_2$ and $C_2 \setminus C_1$ remains the same in G, which is $\lambda - w(e_2)$. Since e_2 is vital, $w(e_2) > 0$. So, $\lambda - w(e_2) < \lambda_S < \lambda$, where λ_S is the capacity of S-mincut in G. Hence, we have a Steiner cut of capacity strictly smaller than S-mincut, a contradiction.

We now prove the converse part. Suppose $C_1 \cap C_2$ is a mincut for edge e_2. Since x_2 is a Steiner vertex, by Definition 10, x_2 is in $C_1 \cap C_2$. Since $C_1 \cap C_2 \subseteq C_1$, x_2 is in C_1. This completes the proof. ◀

For any pair of nonSteiner vertices $a, b \in V'$, it turns out that Lemma 25 does not necessarily hold as follows. As shown in Figure 1(i), s_2 is present in nearest mincut A of edge (s_1, a) but nearest mincut B for edge (s_2, b) crosses A.

Recall that our objective is to design an $\mathcal{O}(n)$ space structure for storing a mincut for every edge from VitType-3(u). By Lemma 24, the set of nearest mincuts for all edges from VitType-3(u) satisfies DISJOINT property. By exploiting Lemma 25, we now establish two interesting properties satisfied by the nearest mincuts for all edges from VitType-3(u) – UNIQUENESS PROPERTY (Lemma 26) and SUBSET PROPERTY (Lemma 27). These properties help in designing an $\mathcal{O}(n)$ space data structure for storing them. We first establish the UNIQUENESS PROPERTY in the following lemma.

▶ **Lemma 26** (UNIQUENESS PROPERTY)**.** *For any edge $e = (x, u)$ from VitType-3(u), the nearest mincut for edge e is unique.*

Proof. Suppose C_1 and C_2 are a pair of distinct nearest mincuts for edge e. It follows from Lemma 25 that $C = C_1 \cap C_2$ is a mincut for edge e. So, C is a proper subset of both C_1 and C_2, which contradicts that C_1 and C_2 are nearest mincuts for edge e. ◀

Although the nearest mincut for each edge from VitType-3(u) is unique (Lemma 26), the UNIQUENESS PROPERTY alone can only guarantee a data structure occupying $\mathcal{O}(n|S|)$ space for all edges from VitType-3(u). To achieve a better space, we now explore the relation

between the nearest mincuts for a pair of edges from VitType-3(u). Since nearest mincut for an edge from VitType-3(u) is unique (Lemma 26), without causing any ambiguity, we consider $N(e)$ to denote the unique nearest mincut for an edge e from VitType-3(u).

Let $e_1 = (x_1, u)$ and $e_2 = (x_2, u)$ be a pair of edges from VitType-3(u). If neither $x_1 \in N(e_2)$ nor $x_2 \in N(e_1)$, then, by Lemma 24, $N(e_1)$ is disjoint from $N(e_2)$. The other cases are when $x_1 \in N(e_2)$ or $x_2 \in N(e_1)$. We exploit Lemma 25 to establish the following property. This property states that $N(e_1)$ is either identical to $N(e_2)$ or one of $\{N(e_1), N(e_2)\}$ contains the other.

▶ **Lemma 27** (Subset Property)**.** *Let (x, u) and (x', u) be a pair of edges from VitType-3(u). Then, $x' \in N((x, u))$ if and only if $N((x', u)) \subseteq N((x, u))$.*

Proof. Let $C = N((x, u))$ and $C' = N((x', u))$. Let us assume to the contrary that $C' \nsubseteq C$. It is given that $x' \in C$. Therefore, by Lemma 25, $C \cap C'$ is also a mincut for edge (x', u). This contradicts that C' is a nearest mincut for edge (x', u).

Since $N((x', u)) \subseteq N((x, u))$ and (x', u) is a contributing edge of $N((x', u))$ with $x' \in N((x', u))$, therefore, x' also belong to $N((x, u))$. This completes the proof. ◀

Let (x_1, u) and (x_2, u) be edges from VitType-3(u), where $x_1, x_2 \in S$. Let C_1 and C_2 be nearest mincuts for edges (x_1, u) and (x_2, u), respectively. It follows from Lemma 27 and Lemma 24 that there are three possibilities for C_1 and C_2 – C_1 is the same as C_2, one of C_1 and C_2 is a proper subset of the other, and C_1 is disjoint from C_2. Therefore, for any vertex $u \in V'$, the set containing the nearest mincuts for every edge from VitType-3(u) forms a Laminar family $\mathcal{L}(u)$ (Definition 14) on set V. This inference, along with Lemma 15, leads to the following result.

▶ **Lemma 28.** *There is an $\mathcal{O}(n)$ space tree $\mathcal{T}_{\mathcal{L}(u)}$ that satisfies the following property. For each edge (x, u) from VitType-3(u) with $x \in S$, SubTree(x) of tree $\mathcal{T}_{\mathcal{L}(u)}$ is the nearest mincut for edge (x, u).*

Data Structure $\mathcal{F}_3$ for all vital edges from Type-3. For each nonSteiner vertex $u \in V'$, we construct a tree $\mathcal{T}_{\mathcal{L}(u)}$ based on the laminar family $\mathcal{L}(u)$ consisting of the nearest mincuts for all edges from VitType-3(u). Since V' contains nonSteiner vertices of G only, there can be at most $n - |S|$ vertices in V'. Therefore, by Lemma 28, the overall space occupied by the data structure is $\mathcal{O}(n(n - |S|))$.

Reporting a mincut for a vital edge from Type-3 using $\mathcal{F}_3$. Given any vital edge $e = (x, u)$ from Type-3, where $x \in S$ and $u \in V \setminus S$, by following Lemma 28, we report the set of vertices stored in SubTree(x) of tree $\mathcal{T}_{\mathcal{L}(u)}$ as the nearest mincut for edge (x, u).

Note that given any edge e from Type-3 and any value Δ satisfying $0 \leq \Delta \leq w(e)$, by using the data structure of Lemma 17, we can determine in $\mathcal{O}(1)$ time whether the capacity of S-mincut reduces after reducing $w(e)$ by Δ. This leads to the following lemma for answering query CUT for all edges from Type-3.

▶ **Lemma 29** (Sensitivity Oracle for Type-3 Edges)**.** *For any Steiner set $S \subseteq V$, there is an $\mathcal{O}((n - |S|)n)$ space data structure that, given any edge e from Type-3 and any value Δ satisfying $0 \leq \Delta \leq w(e)$, can report an S-mincut C in $\mathcal{O}(|C|)$ time after reducing the capacity of edge e by Δ.*

Algorithm 2 Answering Query CUT.

```
1:  procedure CUT(e = (x, y), Δ)
2:      Let C be a Steiner mincut of G;
3:      Assign MINCUT ← C;
4:      TYPE ← the type of edge e determined using the endpoints {x, y};
5:      if TYPE == 1 then
6:          Assign MINCUT ← a mincut for edge e using data structure of Lemma 21;
7:      else if TYPE == 2 then
8:          Assign MINCUT ← a mincut for edge e using data structure of Lemma 23;
9:      else if TYPE == 3 then
10:         Verify using the data structure in Lemma 17 whether e is vital;
11:         if e is a vital edge then
12:             Assign MINCUT ← a mincut for edge e using data structure of Lemma 29;
13:         else
14:             do nothing;
15:         end if
16:     end if
17:     return MINCUT;
18: end procedure
```

Lemma 21, Lemma 23, and Lemma 29 complete the proof of Theorem 5(2).

The pseudo-code for answering query CUT is provided in Algorithm 2. Algorithm 2 is invoked with the failed edge e and the change in capacity Δ of edge e satisfying $0 \leq \Delta \leq w(e)$. In Step 10 of Algorithm 2, the change in capacity (Δ) is required to determine if edge e is a vital edge. Otherwise, Algorithm 2 fails to report the valid S-mincut after reducing the capacity of edge e.

7 Conclusion

We have designed the first Sensitivity Oracle for Steiner mincuts in weighted graphs. It also includes the first Sensitivity Oracle for global mincut in weighted graphs. Interestingly, our Sensitivity Oracle occupies space subquadratic in n when $|S|$ approaches n and also achieves optimal query time. On the other hand, it matches the bounds on both space and query time with the existing best-known results for (s,t)-mincut [12, 2].

Our quadratic space single edge Sensitivity Oracle does not assume that the capacity of the failed edge is known. We have also complemented this result with matching lower bounds. Now, it would be great to see whether there is any single edge Sensitivity Oracle for Steiner mincut that occupies only $\mathcal{O}(n)$ space assuming the capacity of the failed edge is known.

Finally, our obtained structure that breaks the quadratic bound is quite simple as it is a forest of $\mathcal{O}(n - |S|)$ trees. We strongly believe that our techniques and structures will be quite useful for addressing several problems in the future, including the problem of designing a Sensitivity Oracle for S-mincut that can handle failure of multiple edges.

References

1 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications.* Prentice Hall, 1993.

2 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.

3 Surender Baswana and Koustav Bhanja. Vital edges for (s, t)-mincut: Efficient algorithms, compact structures, & optimal sensitivity oracles. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 17:1–17:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.ICALP.2024.17.

4 Surender Baswana, Koustav Bhanja, and Abhyuday Pandey. Minimum+1 (s, t)-cuts and dual-edge sensitivity oracle. *ACM Trans. Algorithms*, 19(4):38:1–38:41, 2023. doi:10.1145/3623271.

5 Surender Baswana, Shreejit Ray Chaudhury, Keerti Choudhary, and Shahbaz Khan. Dynamic dfs in undirected graphs: Breaking the o(m) barrier. *SIAM Journal on Computing*, 48(4):1335–1363, 2019. doi:10.1137/17M114306X.

6 Surender Baswana, Keerti Choudhary, Moazzam Hussain, and Liam Roditty. Approximate single-source fault tolerant shortest path. *ACM Transactions on Algorithms (TALG)*, 16(4):1–22, 2020. doi:10.1145/3397532.

7 Surender Baswana, Shiv Gupta, and Till Knollmann. Mincut sensitivity data structures for the insertion of an edge. *Algorithmica*, 84(9):2702–2734, 2022. doi:10.1007/S00453-022-00978-0.

8 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. doi:10.1137/1.9781611977073.27.

9 Koustav Bhanja. Minimum+ 1 steiner cuts and dual edge sensitivity oracle: Bridging the gap between global cut and (s, t)-cut. *arXiv preprint*, 2024. doi:10.48550/arXiv.2406.15129.

10 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. doi:10.1145/3564246.3585251.

11 Ruoxu Cen, William He, Jason Li, and Debmalya Panigrahi. Steiner connectivity augmentation and splitting-off in poly-logarithmic maximum flows. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2449–2488. SIAM, 2023. doi:10.1137/1.9781611977554.CH95.

12 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.

13 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.

14 Richard Cole and Ramesh Hariharan. A fast algorithm for computing steiner edge connectivity. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 167–176, 2003. doi:10.1145/780542.780568.

15 Efim A Dinitz, Alexander V Karzanov, and Michael V Lomonosov. On the structure of the system of minimum edge cuts in a graph. *Issledovaniya po Diskretnoi Optimizatsii*, pages 290–306, 1976.

16 Ye Dinitz and Alek Vainshtein. Locally orientable graphs, cell structures, and a new algorithm for the incremental maintenance of connectivity carcasses. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 302–311, 1995.

17 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.

18 Yefim Dinitz and Alek Vainshtein. The connectivity carcass of a vertex subset in a graph and its incremental maintenance. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 716–725, 1994. doi:10.1145/195058.195442.

19 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.

20 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.

21 Zhongtian He, Shang-En Huang, and Thatchaphol Saranurak. Cactus representations in polylogarithmic max-flow via maximal isolating mincuts. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1465–1502. SIAM, 2024. doi:10.1137/1.9781611977912.60.

22 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. doi:10.1137/1.9781611976465.163.

23 Stephen Jue and Philip N. Klein. A near-linear time minimum steiner cut algorithm for planar graphs. *CoRR*, abs/1912.11103, 2019. arXiv:1912.11103.

24 L. Lovász. *Combinatorial problems and exercises*. North-Holland Publishing Co., Amsterdam-New York, 1979.

25 Merav Parter. Dual failure resilient BFS structure. In Chryssis Georgiou and Paul G. Spirakis, editors, *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.

26 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.