Minimum+1 (s,t)-cuts and dual edge sensitivity oracle

Surender Baswana
Department of Computer Science & Engineering, IIT Kanpur, Kanpur – 208016, India

Koustav Bhanja
Department of Computer Science & Engineering, IIT Kanpur, Kanpur – 208016, India

Abhyuday Pandey
Department of Computer Science & Engineering, IIT Kanpur, Kanpur – 208016, India

Abstract

Let G be a directed multi-graph on n vertices and m edges with a designated source vertex s and a designated sink vertex t. We study the (s,t)-cuts of capacity minimum+1 and as an important application of them, we give a solution to the dual edge sensitivity for (s,t)-mincuts – reporting the (s,t)-mincut upon failure or addition of any pair of edges.

Picard and Queyranne [Mathematical Programming Studies, 13(1):8-16, 1980] showed that there exists a directed acyclic graph (DAG) that compactly stores all minimum (s,t)-cuts of G. This structure also acts as an oracle for the single edge sensitivity of minimum (s,t)-cut. Dinitz and Nutov [STOC, pages 509-518, 1995] showed that there exists an $\mathcal{O}(n)$ size 2-level cactus model that stores all global cuts of capacity minimum+1. However, for minimum+1 (s,t)-cuts, no such compact structures exist till date. We present the following structural and algorithmic results on minimum+1 (s,t)-cuts.

1. There exists a pair of DAGs of size $\mathcal{O}(m)$ that compactly store all minimum+1 (s,t)-cuts of G. Each minimum+1 (s,t)-cut appears as a (s,t)-cut in one of the 2 DAGs and is 3-transversal – it intersects any path in the DAG at most thrice.
2. There exists an $\mathcal{O}(n^2)$ size data structure that, given a pair of vertices $\{u,v\}$ which are not separated by an (s,t)-mincut, can determine in $\mathcal{O}(1)$ time if there exists a minimum+1 (s,t)-cut, say (A,B), such that $\{s,u\} \in A$ and $\{v,t\} \in B$; the corresponding cut can be reported in $\mathcal{O}(|B|)$ time.
3. There exists an $\mathcal{O}(n^2)$ size data structure that solves the dual edge sensitivity problem for (s,t)-mincuts. It takes $\mathcal{O}(1)$ time to report the value of a resulting (s,t)-mincut (A,B) and $\mathcal{O}(|B|)$ time to report the cut.
4. For the data structure problems addressed in (2) and (3) above, we also provide a matching conditional lower bound. We establish a close relationship among three seemingly unrelated problems – all-pairs directed reachability problem, the dual edge sensitivity problem for (s,t)-mincuts, and 2×2 maximum flow. Assuming the directed reachability hypothesis, this leads to $\tilde{\Omega}(n^2)$ lower bounds on the space for the latter two problems.

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

Keywords and phrases mincut, maxflow, fault tolerant.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.52

1 Introduction

The concept of cuts of a graph is very fundamental in graph theory and has many algorithmic applications as well. There are mainly two types of cuts – global cuts and (s,t)-cuts. A set of edges whose removal disconnects a given undirected graph is called a *global* cut. Let $G = (V,E)$ be a directed multi-graph (E is a multiset) consisting of $n = |V|$ vertices and

$m = |E|$ edges with a designated source vertex s and a designated sink vertex t. An (s,t)-cut of G is defined as follows.

▶ **Definition 1** ((s,t)-cut). *For a subset $C \subset V$ with $s \in C$ and $t \notin C$, the set of outgoing edges of C is called an (s,t)-cut.*

It follows from Definition 1 that for each (s,t)-cut, there exists at least one subset $C \subset V$ that defines it. For the simplicity of exposition and without causing any confusion, henceforth we shall use C to denote the corresponding (s,t)-cut as well. An edge is said to *contribute* to an (s,t)-cut C if it is an outgoing edge of C. The set of edges that have one endpoint in C and another endpoint in $\overline{C}$ is known as the *edge-set* of C, denoted by $E(C)$.

A (s,t)-cut (likewise a global cut) consisting of least number of contributing edges is called an (s,t)-mincut (likewise a global mincut).

There has been extensive research in designing efficient algorithms for global mincuts ([15], [18], [19]) as well as (s,t)-mincuts ([10], [16]). In addition, elegant graph structures have been invented that compactly store and characterize these cuts – Cactus graph for all global mincuts given by Dinitz, Karzanov, and Lomonosov [5], and a directed acyclic graph (DAG) given by Picard and Queyranne [23] for all (s,t)-mincuts. These structures can also serve as an efficient data structure for single edge sensitivity problem – to report the (s,t)-mincut (or global mincut) after the insertion/failure of an edge.

It is very natural to ask if there exists any compact structure for cuts of value greater than the value of the minimum cuts. For global minimum+1 cuts, Dinitz and Nutov answered this question in affirmative. In a seminal work [7], they showed that there exists an $\mathcal{O}(n)$ size 2-level cactus model that stores all minimum+1 cuts; they also gave a characterization of these cuts. An incremental maintenance of this structure solves the problem of maintaining minimum+2 edge connected components for any value of minimum cuts under insertion of edges; generalizing the results of Galil and Italiano [12] and Dinitz [6].

However, for minimum+1 (s,t)-cuts, to the best of our knowledge, no such compact structure exists till date. Note that the approach taken by Picard and Queyranne [23] for (s,t)-mincuts does not seem extendable for minimum+1 (s,t)-cuts. This is because their structure is based on the residual network resulting from a maximum (s,t)-flow and thus crucially exploits the equivalence between maximum flow and minimum cut (Ford and Fulkerson [10]); unfortunately, for a given minimum+1 (s,t)-cut, there is no equivalent (s,t)-flow.

While a compact structure for minimum+1 (s,t)-cuts is of significant importance from a graph theoretic perspective, equally important is a compact data structure that can efficiently answer the following fundamental query for any given edge $(u,v) \in E$.

$Q(u,v)$: report a minimum+1 (s,t)-cut C, if exists, such that $u \in C$, $v \in \overline{C}$.

Interestingly, the DAG structure of Picard and Queyranne [23] for (s,t)-mincuts can answer efficiently the question stated above in case of (s,t)-mincuts. For this purpose, it crucially exploits the property that the set of (s,t)-mincuts are closed under intersection and union operations. Unfortunately, this property no longer holds for minimum+1 (s,t)-cuts, thus making it quite nontrivial to design a data structure for answering query Q.

The study of minimum+1 (s,t)-cuts has an immediate application as well. As is evident from the following fact, their study is indispensable for efficiently solving the dual edge sensitivity problem for (s,t)-mincuts.

▶ **Fact 1.1.** *The failure of a given pair of edges will reduce the value of (s,t)-mincut if and only if there is a minimum (s,t)-cut containing any failed edge or a minimum+1 (s,t)-cut containing both the failed edges.*

Sensitivity data structure for a given graph problem is motivated by the fact that graphs in the real world are prone to failures of vertices/edges. These failures are transient in nature. So while the set of failed vertices/edges keep changing with time, at any stage of time, the number of failed vertices/edges remains quite small. Therefore, the aim is to have a compact data structure that can efficiently report the solution of the given problem for a given set of failed vertices/edges. In a similar manner, we would like to efficiently report the solution of the given problem for a given set of newly added vertices/edges. This may help us determine which newly added vertex/edge changes the solution most significantly. In the past, many elegant fault-tolerant structures have been designed for various classical problems, like single-source reachability [17], shortest paths [4], breadth-first search [21], (s,t)-mincuts [23], all-pairs mincuts [1] etc, which can handle the failure of a single vertex/edge. It is certainly interesting and important to handle more than a single failure/addition. In this endeavour, it is quite natural to first design data structures that can handle dual failures (or dual insertions). This either helps, or exposes the difficulty, in solving the problem in its generality. It has turned out that the data structures that handle dual failures are often more complex and require deeper insight into the problem than the data structures that handle only single failure. This is evident at least for the following problems – single-source reachability [3], breadth-first search [20], shortest paths [9] etc. For the case of (s,t)-mincuts, note that the 40 years old data structure of Picard and Queyranne [23] is the only known sensitivity data structure and it can handle only a single edge failure/addition. It occupies $\mathcal{O}(m)$ space and can report the value of the resulting (s,t)-mincut and the corresponding cut in $\mathcal{O}(1)$ and $\mathcal{O}(m)$ time, respectively. No nontrivial data structure exists for the dual-edge sensitivity of (s,t)-mincuts till date.

1.1 New results and their overview

Let λ be the value of the (s,t)-mincut in G. Henceforth, we use $(\lambda+k)$ (s,t)-cut to denote a minimum+k (s,t)-cut, $k \in \{0,1\}$. Using just the sub-modularity of (s,t)-cuts (Lemma 9) and the relation between any pair of (s,t)-mincuts, we first present an alternate DAG structure, denoted by $\mathcal{D}_\lambda$, that compactly stores and characterizes all (s,t)-mincuts as follows. An (s,t)-cut in G is a (s,t)-mincut if and only if it appears as a 1-transversal cut in $\mathcal{D}_\lambda$ – the edge-set of an (s,t)-cut intersects any path in $\mathcal{D}_\lambda$ at most once. It can be easily observed that $\mathcal{D}_\lambda$, upon reversal of its edges, is identical to the DAG of Picard and Queyranne [23]. The major advantage of the approach taken for designing this alternate DAG is that it shows a way to design compact structures for $(\lambda+1)$ (s,t)-cuts using only the properties of (s,t)-cuts without exploiting the relation between cuts and flows. We now present an overview of our main results on the $(\lambda+1)$ (s,t)-cuts.

The set of minimum cuts are closed under intersection and union. This property has played a crucial role in designing compact structure as well as characterization of these mincuts – cactus graph for global mincuts by Dinitz, Karzanov, Lomonosov [5], the skeleton structure for Steiner mincuts by Dinitz and Vainshtein [8], and DAG for (s,t)-mincuts described in this paper. However, it turns out that $(\lambda+1)$ (s,t)-cuts are not closed under intersection as well as union. Therefore, in order to design compact structure for $(\lambda+1)(s,t)$-cuts and their characterization, we analyse the relation between a pair of $(\lambda+1)(s,t)$-cuts. On the basis of the capacity of the cuts resulting from the intersection and union, each pair of $(\lambda+1)$ (s,t)-cuts is classified into exactly one of the three types – Type-1, Type-2 and Type-3. While designing compact structure for $(\lambda+1)$ (s,t)-cuts, the presence of pairs of Type-1 cuts poses a challenge in characterizing them. Likewise the presence of pairs of Type-2 cuts poses a challenge in determining the (s,t)-mincut upon the failure/addition of any pair of edges. A

natural idea to conquer these challenges is to partition the set of $(\lambda+1)$ (s,t)-cuts into a *small* number of sets so that there is no pair of cuts from Type-1 (likewise Type-2) in a set. It can be observed that any arbitrary partitioning may produce a *large* number of sets which may defeat our objective of designing compact structures.

We present a technique called *covering* of (s,t)-cuts using which we can build a pair of graphs that cover all (s,t)-cuts of G and each of them has no pair of cuts from Type-1. It also helps in carefully handling pairs of cuts from Type-2. However, the covering technique works only for a graph with at most two (s,t)-mincuts, and hence can not be applied directly. In order to tackle this problem we introduce the concept of $(\lambda+1)$ (s,t)-class as follows.

▶ **Definition 2.** *A $(\lambda+1)$ (s,t)-class is a maximal set of vertices $A \subset V$ such that any pair of vertices from A are not separated by any (s,t)-mincut.*

It can be observed that the set of vertices of G that are assigned to a vertex of $\mathcal{D}_\lambda$ corresponds to a $(\lambda+1)$ (s,t)-class. We first form a partition of the set of all $(\lambda+1)$ (s,t)-cuts (excluding a set of *degenerate* cuts) with respect to the $(\lambda+1)$ (s,t)-classes. For each $(\lambda+1)$ (s,t)-class $\mathcal{W}$, we construct a new graph $G(\mathcal{W})$ that preserves all $(\lambda+1)$ (s,t)-cut of G that subdivides $\mathcal{W}$ and has at most two (s,t)-mincuts. We show that it is sufficient to work with $G(\mathcal{W})$ for each $\mathcal{W}$ to design compact structure for $(\lambda+1)$ (s,t)-cuts as well as dual edge sensitivity data structure for (s,t)-mincuts.

1. A 2-level DAG structure for $(\lambda+1)$ (s,t)-cuts: Along similar lines of $\mathcal{D}_\lambda$, we build a compact graph $\mathcal{D}_{\lambda+1}$ that stores all $(\lambda+1)$ (s,t)-cuts of $G(\mathcal{W})$. In order to establish the 1-transversality property of (s,t)-mincuts in $\mathcal{D}_\lambda$, the acyclicity of $\mathcal{D}_\lambda$ played a crucial role. Therefore, at first sight, we would expect $\mathcal{D}_{\lambda+1}$ to be acyclic and a $(\lambda+1)$ (s,t)-cut C in $G(\mathcal{W})$ to be a ℓ-transversal cut in $\mathcal{D}_{\lambda+1}$ – edge-set of C intersects any path in $\mathcal{D}_{\lambda+1}$ at most ℓ times, for some constant ℓ. We find that $\mathcal{D}_{\lambda+1}$ is not necessarily acyclic and, surprisingly enough, a $(\lambda+1)$ (s,t)-cut may appear in $\mathcal{D}_{\lambda+1}$ as an $\Omega(n)$-transversal cut. The root cause of non-acyclicity is the presence of pairs of cuts from Type-1. In order to tackle pairs of cuts from Type-1, we exploit the fact that $G(\mathcal{W})$ has at most two (s,t)-mincuts. We use the covering technique to construct a pair of graphs $G(\mathcal{W})^I$ and $G(\mathcal{W})^U$ that are Type-1 free. Now applying the same approach that was applied to obtain $\mathcal{D}_\lambda$, we construct a pair of DAGs – $\mathcal{D}_{\lambda+1}$ for graph $G(\mathcal{W})^I$ and $\mathcal{D}_{\lambda+1}$ for graph $G(\mathcal{W})^U$. This pair of DAGs is capable of characterizing each $(\lambda+1)$ (s,t)-cut in G that subdivides $\mathcal{W}$ as follows. A $(\lambda+1)$ (s,t)-cut in $G(\mathcal{W})$ appears as a *3-transversal* cut in exactly one of the two DAGs. In this way we obtain an $\mathcal{O}(m)$ size structure for compactly storing and characterizing all $(\lambda+1)$ (s,t)-cuts and it consists of two levels – (i) DAG $\mathcal{D}_\lambda$ and (ii) a pair of DAGs for each $(\lambda+1)$ (s,t)-class associated with a vertex of $\mathcal{D}_\lambda$.

Now we attempt to answer query $Q(u,v)$ using our 2-level DAG structure. Note that each 1-transversal cut in $\mathcal{D}_\lambda$ is also a (s,t)-mincut in G. Hence for the set of (s,t)-mincuts the query can be answered efficiently using a topological ordering of $\mathcal{D}_\lambda$. However, a 3-transversal cut in the pair of DAGs needs not be a $(\lambda+1)$ (s,t)-cut because of the existence of *certain* pairs of cuts from Type-2.

2. Data structure for $(\lambda+1)$ (s,t)-cuts: For each vertex u, there exists a unique (s,t)-mincut C, called *nearest (s,t)-mincut*, that keeps u on the side of s such that $C \subseteq C'$ for every other (s,t)-mincut C' that keeps u on the side of s. We can use this nearest (s,t)-mincut of u to determine if there exists a (s,t)-mincut C such that $u \in C$ and $v \in \overline{C}$ for any pair of vertices $\{u,v\}$. Unfortunately, there are multiple nearest $(\lambda+1)$ (s,t)-cuts that keep u on the side of s, and hence this approach fails.

Let $\mathcal{W}$ be the $(\lambda+1)$ (s,t)-class to which u belongs. Let C and C' be any pair of nearest $(\lambda+1)$ (s,t)-cuts of u. We show that C and C' do not *cross* in $\mathcal{W}$, that is, $\overline{C \cup C'} \cap \mathcal{W} = \emptyset$. Using this crucial insight, we are able to design an $\mathcal{O}(n^2)$ size data structure summarized in the following theorem.

▶ **Theorem 3.** *Let G be a directed multi-graph on n vertices and m edges with a designated source vertex s and a designated sink vertex t. There exists a data structure occupying $\mathcal{O}(n^2)$ space that can determine in $\mathcal{O}(k)$ time whether there exists a $(\lambda+1)$ (s,t)-cut C such that $u \in C$ and $v_1, \ldots, v_k \in \overline{C}$ for any given vertices $u, v_1, \ldots, v_k$ belonging to a $(\lambda+1)$ (s,t)-class. It can also report C in $\mathcal{O}(|\overline{C}|)$ time.*

3. An oracle for dual edge sensitivity for (s,t)-mincuts: We show that there is a data structure $\{\mathcal{F}, \mathcal{I}\}$ occupying $\mathcal{O}(n^2)$ space which is capable of answering dual edge sensitivity query for (s,t)-mincuts in $\mathcal{O}(1)$ time. The data structure $\mathcal{F}$ for handling dual edge failure query is obtained as follows.

When both failed edges are not belonging to the same $(\lambda+1)$ (s,t)-class, data structures for (s,t)-mincuts are sufficient to answer the query. The main challenge arises when endpoints of both failed edges are belonging to the same $(\lambda+1)$ (s,t)-class $\mathcal{W}$. Notice that the data structure for $(\lambda+1)$ (s,t)-cut from Theorem 3 can determine whether there is a $(\lambda+1)$ (s,t)-cut in which a single failed edge is contributing, but cannot answer if both edges are contributing to a single $(\lambda+1)$ (s,t)-cut. Suppose there is a $(\lambda+1)$ (s,t)-cut C^* of G in which both failed edges, say (x,y) and (x',y'), are contributing. Then the necessary condition is that y' must not belong to the nearest $(\lambda+1)$ (s,t)-cut from x to y, and y must not belong to the nearest $(\lambda+1)$ (s,t)-cut from x' to y'. Therefore, if necessary condition holds then both edges are contributing to the union of the two nearest $(\lambda+1)$ (s,t)-cuts. However, the union needs not necessarily be a $(\lambda+1)$ (s,t)-cut because of the existence of certain pairs of $(\lambda+1)$ (s,t)-cuts from Type-2.

We employ the covering technique to tackle pairs of cuts from Type-2. Unfortunately, covering does not necessarily eliminate all Type-2 pairs of cuts like the way it does in case of Type-1 pairs. In order to tackle the problem arising due to the prevailing Type-2 pairs, we exploit the insight into the structure of $G(\mathcal{W})^I$ and $G(\mathcal{W})^U$. The end result is that, in order to determine whether any given pair of edges are contributing to a single $(\lambda+1)$ (s,t)-cut, we only have to perform a couple of nearest $(\lambda+1)$ (s,t)-cuts queries on $G(\mathcal{W})^I$ and $G(\mathcal{W})^U$.

Interestingly $\mathcal{F}$ can also determine in $\mathcal{O}(kl)$ time whether there exists a $(\lambda+1)$ (s,t)-cut C such that $u_1, \ldots, u_k \in C$ and $v_1, \ldots, v_l \in \overline{C}$ for any given vertices $u_1, \ldots, u_k, v_1, \ldots, v_l$ belonging to a $(\lambda+1)$ (s,t)-class.

4. Lower bound for various mincut data structures: We establish a close relationship between two seemingly unrelated problems – all-pairs directed reachability problem and dual edge sensitivity problem for (s,t)-mincuts. The classical problem of all-pairs directed reachability is defined as follows – Given a directed graph G on n vertices and m edges, preprocess it to form a data structure which can efficiently report if any given vertex v is reachable from another given vertex u. The problem becomes interesting when the underlying graph is sparse, that is, $m = o(n^2)$. It is natural to ask whether there is any data structure for the all-pairs directed reachability that takes $o(n^2)$ space and $o(m)$ query time ? Goldstein et al. [14], following the seminal work of Patrascu [22], stated a conjecture that concisely conveys the belief.

▶ **Conjecture 4** (Directed Reachability Hypothesis [14]). *Any data structure for the all-pairs reachability in a directed graph must either use $\tilde{\Omega}(n^2)$ space or $\tilde{\Omega}(m)$ query time. ($\tilde{\Omega}(.)$ denotes complexity upto polylogarithmic factors)*

We provide an $\mathcal{O}(m)$ time reduction from any instance of the all-pairs directed reachability problem for a given graph on n vertices and m edges to an instance of the dual edge sensitivity data structure for (s,t)-mincuts on a graph with $O(n)$ vertices and $\mathcal{O}(m)$ edges. Thus, assuming Conjecture 4 holds, our $\mathcal{O}(n^2)$ size data structure for dual edge sensitivity for (s,t)-mincuts is indeed the optimal size data structure for achieving $\mathcal{O}(1)$ query time. As a byproduct of this reduction, we establish conditional lower bounds on 2×2 flow tree. In particular, we show that if conjecture 4 holds, any data structure that can report the value of $(\{s,u\},\{v,t\})$-mincut for s,u,v and t being any four vertices of the graph must either use $\tilde{\Omega}(n^2)$ space or $\tilde{\Omega}(m)$ query time. An interesting implication of this result is a matching conditional lower bound for the data structure described in Theorem 3.

1.2 Related work

Benczur [2] gave an $\mathcal{O}(n^2)$ space geometric representation for all the global cuts of value within $\frac{6}{5}$ times the global minimum cut value. As an important application, this structure leads to the improvement in the time complexity of *splitting* algorithms [11].

Vazirani and Yannakakis [24] addressed the following fundamental question about (s,t)-cuts. – Is there a polynomial time algorithm to compute an (s,t)-cut of second minimum weight? They answered this question in the affirmative by showing that there is an algorithm which can compute a k^{th} minimum weight (s,t)-cut using only $\mathcal{O}(n^{2(k-1)})$ maximum flow computations. This algorithm gives an implicit structure for all (s,t)-cuts – a binary tree with ℓ leaves, that stores all (s,t)-cuts of G and the number of (s,t)-cuts is ℓ.

Another related work is on characterizing (s,t)-cuts using the polyhedron corresponding to the dual of linear programming (LP) formulation on maximum (s,t)-flow. Vazirani and Garg [13] showed that not all (s,t)-cuts can be characterized as the vertices of this polyhedron. Moreover, they modify the dual of the LP formulation by adding a polynomial number of constraints such that the corresponding polyhedron of the resulting LP formulation has all (s,t)-cuts as its vertices.

1.3 Organisation of the paper

Section 2 contains the basic preliminaries and the construction of a quotient graph for a set of $(\lambda + k)$ (s,t)-cuts. The technique of covering (s,t)-cuts is explained in Section 3. Section 4 contains the construction and properties of the alternate DAG structure for (s,t)-mincuts. The structure for $(\lambda + 1)$ (s,t)-cuts and their characterization is discussed in Section 5. The data structure for reporting $(\lambda + 1)$ (s,t)-cuts is constructed in Section 6. Section 7 contains the Oracle for dual-edge sensitivity of (s,t)-mincuts. Finally in Section 8 we give the lower bounds.

2 Preliminaries

For any $X, Y \subseteq V$, the capacity of (X,Y) (denote by $c(X,Y)$) is the number of edges leaving X and entering Y; for brevity we use $c(X)$ to denote $c(X,\overline{X})$ where $\overline{X} = (V \setminus X)$. We say that an (s,t)-cut C *subdivides* $X \subseteq V$ if $C \cap X$ and $\overline{C} \cap X$ are non empty.

▶ **Definition 5** (Nearest $(\lambda+k)$ (s,t)-cut of a vertex)**.** *The set $A \subset V$ with $s \in A$ and $t \in \overline{A}$ is said to be the nearest $(\lambda+k)$ (s,t)-cut of a vertex u if $u \in A$ and there is no $(\lambda+k)$ (s,t)-cut $B \subset V$ such that $u \in B$ and $B \subset A$. The set of all nearest $(\lambda+k)$ (s,t)-cuts of u is denoted by $N_k(u)$.*

▶ **Definition 6** (Nearest $(\lambda+k)$ (s,t)-cut for a pair of vertices)**.** *Let A be a nearest $(\lambda+k)$ (s,t)-cut of a vertex u. For each vertex $v \in \overline{A}$, A is said to be a nearest $(\lambda+k)$ (s,t)-cut from vertex $\{u\}$ to a vertex $\{v\}$. The set of all nearest $(\lambda+k)$ (s,t)-cuts from u to v is denoted by $N_k(u,v)$.*

When $|N_1(u)| = 1$ (likewise $|N_1(u,v)| = 1$), without causing any ambiguity we use $N_1(u)$ (likewise $N_1(u,v)$) to denote the corresponding cut as well.

▶ **Definition 7** (ℓ-transversal cut)**.** *A (s,t)-cut in a directed graph H is said to be ℓ-transversal, $\ell \geq 1$, if any path in H intersects with the edge-set of the (s,t)-cut at most ℓ times.*

▶ **Definition 8** (transpose of a graph)**.** *Let H be a directed multi-graph with designated source vertex s and designated sink vertex t. The transpose of graph H (denoted by $(H)^{\mathcal{T}}$) is obtained by reversing the orientation of each edge of H with role of s and t swapped.*

▶ **Lemma 9** (Submodularity of cuts)**.** *For any $A, B \subseteq V$, $c(A)+c(B) \geq c(A \cap B)+c(A \cup B)$.*

▶ **Lemma 10.** *Let A and B be any two (s,t)-mincuts in G that are crossing, that is, $A \setminus B$ as well as $B \setminus A$ are non-empty. There is no edge of G between $A \setminus B$ and $B \setminus A$. (proof in Appendix A)*

2.1 Quotient graph for a family of (s,t)-cuts

Let $\mathcal{C}$ be a set of $(\lambda+k)$ (s,t)-cuts in graph $G=(V,E)$ where $k \in \{0,1\}$. We define the following binary relation on the vertex set of G.

▶ **Definition 11** (Relation $R_{\lambda+k}$)**.** *Any two vertices $\{x,y\} \in V$ are said to be related by $R_{\lambda+k}$ if and only if x and y are not separated by any $(\lambda+k)$ (s,t)-cut from $\mathcal{C}$.*

It is a simple exercise to show that $R_{\lambda+k}$ defines an equivalence relation on the vertex set. We call each equivalence class defined by $R_{\lambda+k}$ as a $(\lambda+k+1)$ (s,t)-class. It can be observed that any (s,t)-cut that subdivides a $(\lambda+k+1)$ (s,t)-class has capacity strictly larger than $\lambda+k$. The following lemma (proof in Appendix B) states how a $(\lambda+k)$ (s,t)-class is related to a $(\lambda+k)$ (s,t)-cut.

▶ **Lemma 12.** *A $(\lambda+k)$ (s,t)-cut can subdivide at most one $(\lambda+k)$ (s,t)-class.*

Since $R_{\lambda+k}$ is an equivalence relation, the $(\lambda+k+1)$ (s,t)-classes form a partition of the vertex set of G into disjoint subsets. Let $G_{\lambda+k}$ be the quotient graph of G obtained by contracting each of these subsets into single vertices. We call each vertex of the quotient graph as nodes. The node of $G_{\lambda+k}$ containing source s is denoted by S and the node containing sink t is denoted by T. We call an (S,T)-cut of $G_{\lambda+k}$ as (s,t)-cut without causing any ambiguity. The following theorem is immediate from the construction.

▶ **Theorem 13.** *For a directed multi-graph G and a set of $(\lambda+k)$ (s,t)-cuts $\mathcal{C}$ for any $k \in \{0,1\}$, there exists a quotient graph $G_{\lambda+k}$ of G such that $C \in \mathcal{C}$ is an (s,t)-cut in G if and only if C is a $(\lambda+k)$ (s,t)-cut in $G_{\lambda+k}$.*

An edge of $G_{\lambda+k}$ is classified as normal or *inverted* using the following definition.

▶ **Definition 14** (inverted edge for $(\lambda+k)$ (s,t)-cut)**.** *An edge (x,y) of $G_{\lambda+k}$ is said to be an inverted edge if there exists no $(\lambda+k)$ (s,t)-cut $C \in \mathcal{C}$ such that $x \in C$ and $y \in \overline{C}$.*

▶ **Fact 2.1.** *For a directed multi-graph H and a set of $(\lambda+k)$ (s,t)-cuts $\mathcal{C}$, let $H_{\lambda+k}$ be the quotient graph of H. Given a pair of nodes μ, ν in graph $H_{\lambda+k}$, the contraction of all vertices which are mapped to μ or ν into a single vertex in H eliminates precisely all those $(\lambda+k)$ (s,t)-cuts from $\mathcal{C}$ that separate μ and ν.*

Consider two sets of $(\lambda+k)$ (s,t)-cuts $\mathcal{C}$ and $\mathcal{C}'$ such that $\mathcal{C} \subset \mathcal{C}'$. Let $G_{\lambda+k}$ and $G'_{\lambda+k}$ be the quotient graphs for $\mathcal{C}$ and $\mathcal{C}'$ respectively. It follows from construction of quotient graphs that there exists equivalence classes of $G_{\lambda+k}$ which are further split into multiple equivalence classes of $G'_{\lambda+k}$. Therefore, the following lemma is immediate.

▶ **Lemma 15.** *Given a directed multi-graph G, let $G_{\lambda+k}$ and $G'_{\lambda+k}$ be a pair of quotient graphs formed on the sets of $(\lambda+k)$ (s,t)-cuts $\mathcal{C}$ and $\mathcal{C}'$ respectively. If $\mathcal{C} \subset \mathcal{C}'$ then $G_{\lambda+k}$ is a quotient graph of $G'_{\lambda+k}$.*

3 A covering of all (s,t)-cuts of a special graph

We first formalize the notion of *covering* the (s,t)-cuts of a graph using the following definition.

▶ **Definition 16** (Covering all (s,t)-cuts)**.** *Given a directed multi-graph H with a designated source vertex s and a designated sink vertex t, let $F = \{H^1, H^2, \ldots, H^\ell\}$ be a finite set of directed multi-graphs, each defined on the same vertex set as that of H. F is said to cover all (s,t)-cuts of H if and only if the following conditions hold.*

1. *For each (s,t)-cut C in H, there exists a (s,t)-cut C' in H^i for a unique $1 \le i \le \ell$ such that $C = C'$.*
2. *For each (s,t)-cut C of finite capacity in H^i for any $1 \le i \le \ell$, there exists a (s,t)-cut C' in H such that $C = C'$.*

Let H be a directed multi-graph with a designated source vertex s and a designated sink vertex t that has at most two (s,t)-mincuts – $\{s\}$ and (complement of) $\{t\}$. Our aim is to compute a small family $F = \{H^1, H^2, \ldots, H^\ell\}$ that covers all (s,t)-cuts of H and satisfies the following condition – for each pair C, C' of (s,t)-cuts in H^i for any $1 \le i \le \ell$, either their intersection or union is not a λ (s,t)-cut.

We shall show that there exists a family $F = \{H^U, H^I\}$ of only two graphs that *covers* all (s,t)-cuts of H and satisfies the said property. We build two graph H^U and H^I from H as follows. Let x be any arbitrary vertex other than s and t of H. H^I is formed by adding infinite capacity edge from vertex s to x. In a similar way, H^U is formed by adding infinite capacity edge from vertex x to t. The following lemma (proof in Appendix C) ensures that H^U and H^I together cover all $(\lambda+k)$ (s,t)-cuts of H. It exploits the following simple fact – Let x be a vertex of graph H and C be any (s,t)-cut; either $x \in C$ or $x \in \overline{C}$.

▶ **Lemma 17.** *Let $\lambda' > 0$ be a finite number. C is an (s,t)-cut in H of capacity λ' if and only if C is a cut of capacity λ' in H^U or H^I.*

It follows from the construction that the first (s,t)-mincut of H, that is $\{s\}$, is present in H^U while the second (s,t)-mincut of H, that is (complement of) $\{t\}$ is present in H^I. So using Lemma 17, we can state the following theorem.

▶ **Theorem 18.** *Let $H = (V,E)$ be a directed multi-graph on n vertices and m edges with a designated source vertex s and a designated sink vertex t. Suppose H has at most two*

(s,t)-mincuts, $\{s\}$ and $V \setminus \{t\}$. There exists a set of graphs $F = \{H^I, H^U\}$ satisfying the following properties.

1. *F covers all $(\lambda + k)$ (s,t)-cuts of H.*
2. *$V \setminus \{t\}$ is the only λ (s,t)-cut in H^I, and $\{s\}$ is the only λ (s,t)-cut in H^U.*

▶ Note 19. Covering technique cannot be applied to a graph H that has more than two (s,t)-mincuts. This is because, for any vertex x in H, there will be more than one (s,t)-mincuts that keep x on side of s and/or more than one (s,t)-mincuts that keep x on side of t. Therefore, there will be multiple (s,t)-mincuts in H^I or multiple (s,t)-mincuts in H^U, hence violating Theorem 18(2).

4 An alternate DAG structure storing all (s,t)-mincuts

We now present an alternate compact structure $\mathcal{D}_\lambda$ for representing and characterizing all (s,t)-mincuts of G.

Construction of $\mathcal{D}_\lambda$: Let G_λ be the quotient graph defined by all λ (s,t)-cuts of G (Theorem 13). D_λ is the graph obtained by reversing the direction of each inverted edge of G_λ. Exploiting just the elementary properties of (s,t)-mincuts alone (Lemma 10 and submodularity of cuts), we show that D_λ is acyclic. (proof in Appendix D). As the reader may notice, D_λ is related to the DAG of Picard and Queyranne [23] – the mapping of vertices to the nodes of the DAG is identical while the direction of each edge is flipped. Each (s,t)-mincut of G is characterized by a 1-transversal cut in D_λ as stated in the following lemma. (proof in Appendix E).

▶ **Lemma 20** (1-transversality property)**.** *An (s,t)-cut is an (s,t)-mincut of G if and only if it appears as a 1-transversal cut in $\mathcal{D}_\lambda$.*

The following lemma can be proved along similar lines as Lemma 20.

▶ **Lemma 21.** *An (s,t)-cut of G is an (s,t)-mincut if and only if it appears as an (s,t)-cut with no incoming edge in $\mathcal{D}_\lambda$.*

Interestingly, $\mathcal{D}_\lambda$ can serve as a compact data structure for efficiently answering the query $Q(u,v)$ for (s,t)-mincuts as follows. Let τ be a topological ordering of $\mathcal{D}_\lambda$. For each edge (μ,ν) of $\mathcal{D}_\lambda$, $\tau(\mu) < \tau(\nu)$. Hence, it is a simple corollary of Lemma 20 that every prefix of τ is a (s,t)-mincut. Therefore, given any edge $(u,v) \in E$, there is a (s,t)-mincut in G containing it if u and v appear in different nodes of $\mathcal{D}_\lambda$ and the node containing u appears before the node containing v. (proof in Appendix F).

5 Structure of $(\lambda+1)$ (s,t)-cuts

In order to construct a compact structure for $(\lambda+1)$ (s,t)-cuts, we take an approach similar to the construction of $\mathcal{D}_\lambda$ for (s,t)-mincuts and construct a graph $\mathcal{D}_{\lambda+1}$ as follows.

Construction of $\mathcal{D}_{\lambda+1}$: Let $G_{\lambda+1}$ be the quotient graph defined by all $(\lambda+1)$ (s,t)-cuts (Theorem 13). $D_{\lambda+1}$ is the graph obtained by reversing the direction of each inverted edge of $G_{\lambda+1}$. Henceforth $\mathcal{D}_{\lambda+1}$ for a graph H is denoted by $\mathcal{D}_{\lambda+1}(H)$.

$\mathcal{D}_\lambda$ characterizes (s,t)-mincuts of G as 1-transversal cuts. Unfortunately, it turns out that $\mathcal{D}_{\lambda+1}(G)$ is not sufficient for characterizing $(\lambda+1)$ (s,t)-cuts in terms of ℓ-tranversal cuts for some constant ℓ as stated in the following theorem. (Proof in Appendix I).

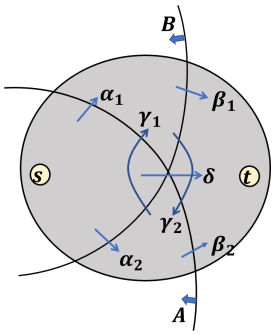

Figure 1 Contributing edges of a pair of (s,t)-cuts A and B between different regions.

▶ **Theorem 22** ($\Omega(n)$-transversality). *There exists a directed multi-graph H on n vertices with a designated source vertex s and a designated sink vertex t having only two (s,t)-mincuts and a $(\lambda+1)$ (s,t)-cut C such that C appears as a $\Omega(n)$-transversal cut in $\mathcal{D}_{\lambda+1}(H)$.*

We now provide a classification of all-pairs of $(\lambda+1)$ (s,t)-cuts. This classification will act as a crucial tool in the analysis of $\mathcal{D}_{\lambda+1}(G)$ as well as establishing properties of the compact structure for $(\lambda+1)$ (s,t)-cuts in Section 5.1.

Suppose A and B are any two $(\lambda+1)$ (s,t)-cuts. Using sub-modularity of cuts (Lemma 9), we arrive at the following inequality, $c(A\cap B)+c(A\cup B)\leq c(A)+c(B)=2(\lambda+1)$. Note that $c(A\cap B)\geq\lambda$ and $c(A\cup B)\geq\lambda$. This implies that the value of $c(A\cap B)+c(A\cup B)$ will always belong to $\{2\lambda,\ 2\lambda+1,\ 2\lambda+2\}$. Therefore, each pair (A,B) of $(\lambda+1)$ (s,t)-cuts can be classified uniquely into one of the three types based on the value of $c(A\cap B)+c(A\cup B)$. We now state the following lemma that will provide an alternate characterization of these three types based on the number of edges between $A\setminus B$ and $B\setminus A$.

▶ **Lemma 23.** *Suppose A and B are any two $(\lambda+1)$ (s,t)-cuts. Let γ_1 and γ_2 denote the number of edges from $A\setminus B$ to $B\setminus A$ and from $B\setminus A$ to $A\setminus B$ respectively. If $c(A\cap B)=\lambda+p$ and $c(A\cup B)=\lambda+q$ for some $p,q\geq 0$, then $\gamma_1+\gamma_2=2-p-q$.*

Proof. The cuts A and B partition G into at most 4 regions. Refer to Figure 1 for the illustration of these regions and the edges contributing to the respective cuts.

The following equations follow directly from Figure 1.

$$c(A\cap B)=\alpha_1+\alpha_2+\delta=\lambda+p \tag{1}$$
$$c(A\cup B)=\beta_1+\beta_2+\delta=\lambda+q \tag{2}$$
$$c(B)=\beta_1+\alpha_2+\gamma_2+\delta=\lambda+1 \tag{3}$$
$$c(A)=\beta_2+\alpha_1+\gamma_1+\delta=\lambda+1 \tag{4}$$

Using equations (1) and (3) we get $\alpha_1=\beta_1+\gamma_2+p-1$. Similarly using equations (1) and (4) we get $\alpha_2=\beta_2+\gamma_1+p-1$. By combining these two equalities, we get $\alpha_1+\alpha_2=\beta_1+\beta_2+\gamma_1+\gamma_2+2p-2$. Hence $\gamma_1+\gamma_2=2-p-q$, since $\alpha_1+\alpha_2=\beta_1+\beta_2+p-q$ from equations (2) and (1). ◀

Refer to Table 1 for two ways to characterize the three types of pairs of $(\lambda+1)$ (s,t)-cuts.

	$c(A \cap B) + c(A \cup B)$	$c(A \setminus B, B \setminus A) + c(B \setminus A, A \setminus B)$
Type-1	2λ	2
Type-2	$2\lambda + 2$	0
Type-3	$2\lambda + 1$	1

Table 1 Classification of any pair (A, B) of $(\lambda + 1)$ (s, t)-cuts.

It follows from Theorem 22 that transversality of $(\lambda + 1)$ (s, t)-cuts cannot be bounded by a constant even if a graph has at most two (s, t)-mincuts – $\{s\}$ and (complement of) $\{t\}$. In the following section, we shall first give a compact structure for $(\lambda + 1)$ (s, t)-cuts for graph G assuming that G has at most two (s, t)-mincuts. Finally, we shall extend it to general graphs with the help of the structure of $\mathcal{D}_\lambda$ in Section 5.1.1.

5.1 Compact representation and characterization of $(\lambda + 1)$ (s, t)-cuts

The characterization of (s, t)-mincuts crucially exploits the fact that $\mathcal{D}_\lambda$ is a DAG. Unfortunately, as shown in Figure 2(i), $\mathcal{D}_{\lambda+1}(G)$ can have cycles. The following lemma states the source of any 2-length cycle in $\mathcal{D}_{\lambda+1}$.

▶ **Lemma 24.** *If graph G does not have any pair of cuts from Type-1, then $\mathcal{D}_{\lambda+1}(G)$ cannot have a cycle of length two.*

Proof. Suppose we have a 2-length cycle $\langle \mu, \nu, \mu \rangle$ in $\mathcal{D}_{\lambda+1}(G)$. It follows from construction of $\mathcal{D}_{\lambda+1}$ that there exists a $(\lambda + 1)$ (s, t)-cut C for the edge (μ, ν) such that $\mu \in C$ and $\nu \in \overline{C}$. In a similar manner there exists a $(\lambda + 1)$ (s, t)-cut C' for the edge (ν, μ) such that $\nu \in C'$ and $\mu \in \overline{C'}$. It can be observed that (C, C') forms a pair of cuts from Type-1, a contradiction. ◀

In fact, the absence of pairs of $(\lambda + 1)$ (s, t)-cuts from Type-1 is a sufficient condition for acyclicity as stated in the following lemma.

▶ **Lemma 25** (acyclicity property)**.** *If graph G does not have any pair of cuts from Type-1, then $\mathcal{D}_{\lambda+1}(G)$ is a directed acyclic graph.*

Proof. We begin with stating the following assertion.
$\mathcal{A}(i)$: For any graph H that has no pair of $(\lambda + 1)$ (s, t)-cuts from Type 1, there is no cycle of length i in $\mathcal{D}_{\lambda+1}(H)$.
We shall now prove, by induction on i, that $\mathcal{A}(i)$ holds for all $i \geq 2$, and this would establish the lemma. The base case, $\mathcal{A}(2)$ follows directly from Lemma 24. Suppose $\mathcal{A}(j)$ holds for all $j < i$. We shall now prove that $\mathcal{A}(i)$ holds.

Suppose there is a cycle O of length i in $\mathcal{D}_{\lambda+1}(H)$ (see Figure 2(ii)). Consider any arbitrary edge (μ, ν) in this cycle. It follows from construction of $\mathcal{D}_{\lambda+1}(H)$ that there exists a $(\lambda + 1)$ (s, t)-cut C of H such that $\mu \in C$ and $\nu \in \overline{C}$. Let $\mathcal{U}$ be the set of vertices of H that are mapped to either μ or ν. Contracting the set $\mathcal{U}$ into a single vertex we obtain a new graph H' from H. It follows from Fact 2.1 and Lemma 15 that $\mathcal{D}_{\lambda+1}(H')$ has to be a quotient graph of $\mathcal{D}_{\lambda+1}(H)$. As a result, cycle O in $\mathcal{D}_{\lambda+1}(H)$ will be mapped to either a cycle of length strictly less than i or a single node in $\mathcal{D}_{\lambda+1}(H')$. The Induction Hypothesis rules out the possibility of the former case. We shall now rule out the possibility of the latter case. This will establish the validity of $\mathcal{A}(i)$.

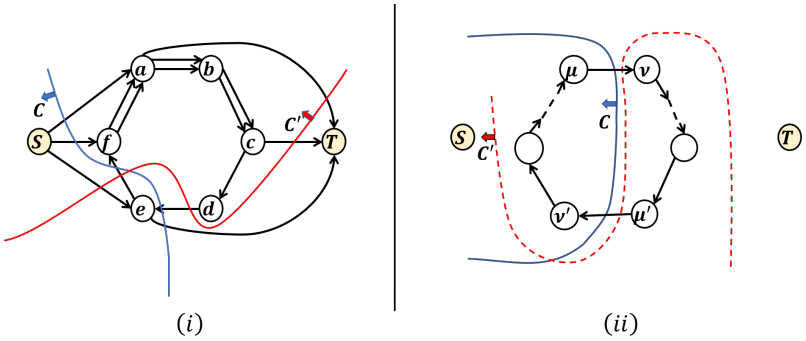

Figure 2 (i) An example of $\mathcal{D}_{\lambda+1}(G)$ with cycle $\langle a, b, c, d, e, f, a\rangle$. Note that here $\mathcal{D}_{\lambda+1}(G)$ turns out to be same as G. (ii) Cycle O in graph $\mathcal{D}_{\lambda+1}(H)$ from the proof of Lemma 25.

The cut C must contain at least one more edge, say (μ', ν'), of cycle O because the cycle O must intersect the edge-set of C at least twice. It follows from the construction of $\mathcal{D}_{\lambda+1}(H)$ that there exists a $(\lambda+1)$ (s,t)-cut, say C', in $\mathcal{D}_{\lambda+1}(H)$ such that $\mu' \in C'$ and $\nu' \in \overline{C'}$. Since μ' and ν' belong to the same node in $\mathcal{D}_{\lambda+1}(H')$, the cut C' is not present in H'. This observation, in the light of Fact 2.1, implies that C' must separate μ and ν (dashed curve in Figure 2). This would imply that for the pair of cuts (C, C'), $c(C \setminus C', C' \setminus C) + c(C' \setminus C, C \setminus C') = 2$. Hence (C, C') forms a Type-1 pair of $(\lambda+1)$ (s,t)-cut in H, a contradiction. ◀

In order to have acyclic structures that collectively preserve all $(\lambda+1)$ (s,t)-cuts, Lemma 25 raises the following question – can we partition the set of $(\lambda+1)$ (s,t)-cuts such that each partition does not contain any pair of $(\lambda+1)$ (s,t)-cut from Type-1? The covering technique (Theorem 18) gives an affirmative answer to this question. In particular, it outputs just a pair of graphs $\{G^I, G^U\}$ such that all $(\lambda+1)$ (s,t)-cuts of G are covered by G^I and G^U together. Moreover, both G^U and G^I will contain exactly one (s,t)-mincut each, and this ensures that there are no pairs of $(\lambda+1)$ (s,t)-cuts from Type-1 in G^I or G^U. Therefore, it follows from Lemma 25 that $\mathcal{D}_{\lambda+1}(G^I)$ and $\mathcal{D}_{\lambda+1}(G^U)$ are acyclic. In the case when G has exactly one (s,t)-mincut, $\mathcal{D}_{\lambda+1}(G)$ itself is acyclic.

Not only $\mathcal{D}_{\lambda+1}(G^I)$ and $\mathcal{D}_{\lambda+1}(G^U)$ are acyclic but also help in characterizing $(\lambda+1)$ (s,t)-cuts as follows.

We show that each $(\lambda+1)$ (s,t)-cut of G is 3-transversal in $\mathcal{D}_{\lambda+1}(G^I)$ or in $\mathcal{D}_{\lambda+1}(G^U)$. Without loss of generality let us consider the graph $\mathcal{D}_{\lambda+1}(G^U)$. In order to accomplish 3-transversality of $(\lambda+1)$ (s,t)-cuts in $\mathcal{D}_{\lambda+1}(G^U)$, we first show that for any path in $\mathcal{D}_{\lambda+1}(G^U)$ whose first node is not S, there is no $(\lambda+1)$ (s,t)-cut which can keep both the first and the last node of the path on side of S, and remaining nodes on side of T. As a warm-up, the following lemma validates this assertion for all paths of length two.

▶ **Lemma 26.** *Let $P = \langle v_0 \neq S, v_1, v_2\rangle$ be a path in $\mathcal{D}_{\lambda+1}(G^U)$. There can not exist any $(\lambda+1)$ (s,t)-cut C such that $v_0, v_2 \in C$ and $v_1 \in \overline{C}$.*

Proof. We give a proof by contradiction. Assume that there is such a $(\lambda+1)$ (s,t)-cut C. It follows from the construction of $\mathcal{D}_{\lambda+1}(G^U)$ that there is a $(\lambda+1)$ (s,t)-cut C' for the edge (v_1, v_2) such that $v_1 \in C'$ and $v_2 \in \overline{C'}$. It can be observed that the edge (v_1, v_2) is an edge from $C' \setminus C$ to $C \setminus C'$. Therefore, $\{C, C'\}$ cannot be a pair from Type-2. $\{C, C'\}$ cannot be pair from Type-1 as well since there is no pair of $(\lambda+1)$ (s,t)-cuts from Type-1 in $\mathcal{D}_{\lambda+1}(G^U)$. So, $\{C, C'\}$ must be a pair from Type-3. Notice that $C \cup C'$ has capacity $\lambda+1$ since the only

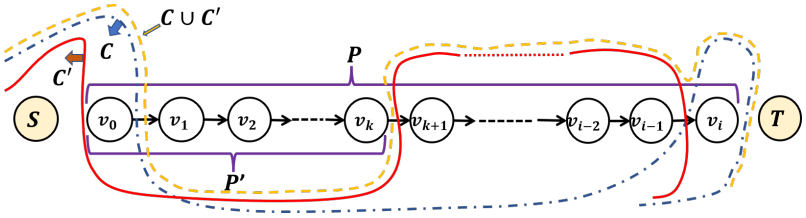

Figure 3 C: dash-dot curve, C': solid curve, $C \cup C'$: dashed curve, (v_{i-1}, v_i) is in $E(C) \cap E(C')$.

λ (s,t)-cut in G^U is $\{s\}$. Hence, $C \cap C'$ must be containing the node S only. Since $v_0 \in C$, therefore, v_0 cannot belong to C' and hence the edge-set of C' must intersect path P again at edge (v_0, v_1). Then, in that case, there are two edges (v_0, v_1) and (v_1, v_2) between $C \setminus C'$ and $C' \setminus C$. This implies that (C, C') are from Type-1, a contradiction. ◀

Generalizing Lemma 26 for any arbitrary length path, we give the following lemma which will provide the foundation for establishing 3-transversality of $(\lambda+1)$ (s,t)-cuts.

▶ **Lemma 27.** *Let $P = \langle v_0 \neq S, v_1, \ldots, v_i \rangle$ be a simple path in $\mathcal{D}_{\lambda+1}(G^U)$. There does not exist a $(\lambda+1)$ (s,t)-cut C such that $v_0, v_i \in C$ and $v_1, \ldots, v_{i-1} \in \overline{C}$.*

Proof. We begin with stating the following assertion.

$\mathcal{A}(i)$: There does not exist a $(\lambda+1)$ (s,t)-cut C such that $v_0, v_i \in C$ and $v_1, \ldots, v_{i-1} \in \overline{C}$ for a path $P = \langle v_0 \neq S, v_1, \ldots, v_{i-1}, v_i \rangle$ of length i.

We shall use proof by induction on the length of the path $i \geq 2$ to show that $\mathcal{A}(i)$ holds. The base case, $\mathcal{A}(2)$ follows directly from Lemma 26.

Let us now assume that $\mathcal{A}(j)$ holds for all $2 \leq j < i$. We shall prove that $\mathcal{A}(i)$ holds using a proof by contradiction. Assume to the contrary that there is a $(\lambda+1)$ (s,t)-cut C such that $v_0, v_i \in C$ and $v_1, \ldots, v_{i-1} \in \overline{C}$ (refer to Figure 3). It follows from the construction of $\mathcal{D}_{\lambda+1}(G^U)$ that there is a $(\lambda+1)$ (s,t)-cut C' for the edge (v_{i-1}, v_i) such that $v_{i-1} \in C'$ and $v_i \in \overline{C'}$. Along similar lines of the proof of Lemma 26 we can argue that (C, C') are pairs from Type-3 with $c(C \cup C') = \lambda + 1$, and edge-set of C' must be intersecting path P at least one more time before the edge (v_{i-1}, v_i). So, let (v_k, v_{k+1}), $0 \leq k < i-1$, be the first edge on P that intersects edge-set of C'. Now, there are two possible cases, either $k = 0$ and $v_{k+1} = v_1$ or $k > 0$. In the former case, it is easy to observe that two edges (v_0, v_1) and (v_{i-1}, v_i) lie between $C' \setminus C$ and $C \setminus C'$. Therefore, (C, C') must be from Type-1, a contradiction. In the latter case, observe that $v_0, \ldots, v_k \notin C'$ and $v_{k+1} \in C'$ because (v_k, v_{k+1}) is the nearest edge from v_0 in P that intersects edge-set of C'. Therefore, $v_0, v_{k+1} \in C \cup C'$ and $v_1, \ldots, v_k \notin C \cup C'$. So, we get a path $P' = \langle v_0, \ldots . v_{k+1} \rangle$ and a $(\lambda+1)$ (s,t)-cut $C \cup C'$ such that $v_0, v_{k+1} \in C \cup C'$ and $v_1, \ldots, v_k \in \overline{C \cup C'}$. Moreover, path P' has length strictly smaller than i because $k < i-1$. Therefore, $\mathcal{A}(k+1)$ fails to hold and $k+1 < i$, a contradiction. ◀

Now in the following lemma, using Lemma 27 we argue that there cannot exist any $(\lambda+1)$ (s,t)-cut C in $\mathcal{D}_{\lambda+1}(G^U)$ such that edge-set of C intersects a path P more than thrice.

▶ **Lemma 28** (3-transversality property). *Each $(\lambda+1)$ (s,t)-cut of G is 3-transversal in $\mathcal{D}_{\lambda+1}(G^I)$ or in $\mathcal{D}_{\lambda+1}(G^U)$.*

Proof. Without loss of generality let us consider $\mathcal{D}_{\lambda+1}(G^U)$. The proof is along similar lines for $\mathcal{D}_{\lambda+1}(G^I)$. We give a proof by contradiction. Assume to the contrary that there exists a $(\lambda+1)$ (s,t)-cut C in G and a path P in $\mathcal{D}_{\lambda+1}(G^U)$ such that edge-set of C intersects path P

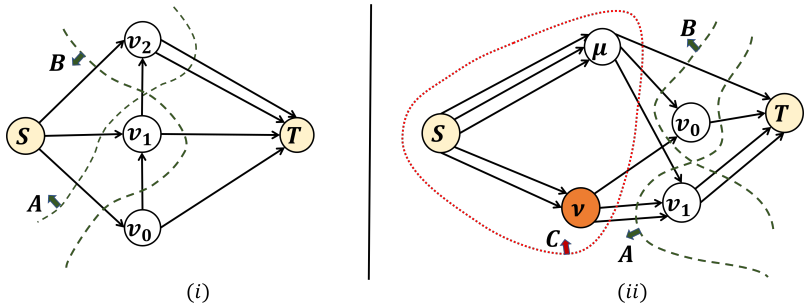

Figure 4 (i) The edge-set of $(\lambda+1)$ (s,t)-cut A intersects path $\langle S, v_1, v_2, T\rangle$ thrice; A and B are $(\lambda+1)$ (s,t)-cuts from Type-3. (ii) A 1-transversal cut $C = A \cap B$ with capacity greater than $\lambda+1$.

more than thrice. Then, path P can be divided into at least 5 contiguous disjoint subpaths $\{P_1, P_2, P_3, P_4, P_5\}$. For the cut C, observe that either each node of P_i is on side of S for all odd i and each node of P_i is on side of T for all even i or vice versa. In the former case, let us consider the subpath $P' = \langle P_3, P_4, P_5\rangle$ of P. For this path P', we have each node of P_3, P_5 in C and each node of P_4 in $\overline{C}$. Moreover, S cannot belong to P_3 because by construction of $\mathcal{D}_{\lambda+1}(G^U)$ there is no incoming edge to S. Therefore, we have a path P' with first node (not S) and last node in $(\lambda+1)$ (s,t)-cut C and other nodes are in $\overline{C}$. This contradicts Lemma 27. In a similar way we can argue the latter case using subpath $P' = \langle P_2, P_3, P_4\rangle$. ◀

We also show that there are $(\lambda+1)$ (s,t)-cuts which may not appear in $\mathcal{D}_{\lambda+1}$ as 1-transversal cut (refer to Figure 4(i)). This is because, for each 3-transversal $(\lambda+1)$ (s,t)-cut A, there exists a $(\lambda+1)$ (s,t)-cut B such that A, B are pairs from Type-3. Therefore, our bound of 3-transversality is tight.

5.1.1 Extension to general graphs

The compact structure and characterization of $(\lambda+1)$ (s,t)-cuts are valid if the graph has at most two (s,t)-mincuts. However, in general, graphs can have exponential number of (s,t)-mincuts. To tackle this difficulty we explore how a $(\lambda+1)$ (s,t)-cut is related to the $(\lambda+1)$ (s,t)-classes of a graph G. There are $(\lambda+1)$ (s,t)-cuts in G which do not subdivide any $(\lambda+1)$ (s,t)-class. We call them *degenerate* $(\lambda+1)$ (s,t)-cuts. These cuts appear in $\mathcal{D}_\lambda$ and have the following characterization. (proof in Appendix G).

▶ **Lemma 29.** *An (s,t)-cut of G is a degenerate $(\lambda+1)$ (s,t)-cut if and only if it appears as an (s,t)-cut with exactly one incoming edge in $\mathcal{D}_\lambda$.*

Henceforth, our main focus is on the non-degenerate $(\lambda+1)$ (s,t)-cuts – cuts that subdivide $(\lambda+1)$ (s,t)-classes. It follows from Lemma 12 that each such $(\lambda+1)$ (s,t)-cut subdivides precisely one $(\lambda+1)$ (s,t)-class. Therefore, the set of all $(\lambda+1)$ (s,t)-cuts can be partitioned into disjoint subsets where each subset subdivides exactly one $(\lambda+1)$ (s,t)-class. This partitioning allows us to work separately with each $(\lambda+1)$ (s,t)-class. Let $\mathcal{W}$ be a $(\lambda+1)$ (s,t)-class. In order to build a compact structure that stores all $(\lambda+1)$ (s,t)-cuts that subdivide $\mathcal{W}$, we now define a graph $G(\mathcal{W})$ associated with $\mathcal{W}$ as follows.

Construction of $G(\mathcal{W})$: Let τ be a topological ordering of $\mathcal{D}_\lambda$ where source node S (likewise T) has the smallest (likewise highest) topological number. $G(\mathcal{W})$ is obtained by forming a quotient graph of G using τ as follows. Let μ be the node of $\mathcal{D}_\lambda$ corresponding to $\mathcal{W}$. All nodes that precede μ in the topological ordering τ are contracted into a single source node S' and every node that succeeds μ are contracted to a single sink node T'.

It is easy to observe that $s \in S'$ (likewise $t \in T'$) since $s \in S$ (likewise $t \in T$). Henceforth without causing any ambiguity we denote an (S', T')-cut in $G(\mathcal{W})$ as an (s,t)-cut in $G(\mathcal{W})$. The following lemma (proof in Appendix H) establishes the mapping between the $(\lambda+1)$ (s,t)-cuts of G and $G(\mathcal{W})$.

▶ **Lemma 30.** *A $(\lambda+1)$ (s,t)-cut C in G subdivides a $(\lambda+1)$ (s,t)-class $\mathcal{W}$ into $(\mathcal{W}_1, \mathcal{W}\setminus\mathcal{W}_1)$ if and only if there exists a $(\lambda+1)$ (s,t)-cut $C' = \mathcal{W}_1 \cup \{S'\}$ in $G(\mathcal{W})$.*

It is easy to observe that graph $G(\mathcal{W})$ can have at most two (s,t)-mincuts – S' and complement of T'. Therefore, we construct pair of DAGs, $\mathcal{D}_{\lambda+1}((G(\mathcal{W}))^I)$ and $\mathcal{D}_{\lambda+1}((G(\mathcal{W}))^U)$, for each $(\lambda+1)$ (s,t)-class $\mathcal{W}$. Now we summarize these structural and characterization results in the following theorem.

▶ **Theorem 31.** *For any directed multi-graph G, on n vertices and m edges with a designated source vertex s and a designated sink vertex t, there exists a 2-level DAG structure of $\mathcal{O}(m)$ size–(i) $\mathcal{D}_\lambda$ and (ii) a pair of DAGs associated with each node of $\mathcal{D}_\lambda$, such that all $(\lambda+1)$ (s,t)-cuts of G are compactly stored and characterized as follows.*

1. *a non-degenerate $(\lambda+1)$ (s,t)-cut of G subdividing a $(\lambda+1)$ (s,t)-class $\mathcal{W}$ is a 3-transversal cut in one of the two DAGs associated with $\mathcal{W}$, and*
2. *an (s,t)-cut of G is a degenerate $(\lambda+1)$ (s,t)-cut if and only if it has exactly one incoming edge in $\mathcal{D}_\lambda$.*

We now explore the possibility of answering query $Q(u,v)$ using the pair of DAGs, $\mathcal{D}_{\lambda+1}(G^I)$ or $\mathcal{D}_{\lambda+1}(G^U)$. Recall that for the case of (s,t)-mincuts, just storing a topological ordering of $\mathcal{D}_\lambda$ is sufficient to answer query $Q(u,v)$ because each 1-transversal cut in $\mathcal{D}_\lambda$ is also an (s,t)-mincut. However, a 1-transversal cut in $\mathcal{D}_{\lambda+1}(G^I)$ or $\mathcal{D}_{\lambda+1}(G^U)$ needs not be a $(\lambda+1)$ (s,t)-cut. For example, cut $A \cap B$ as shown in Figure 4(ii) is 1-transversal but has capacity $\lambda+2$. Note that A and B form a $(\lambda+1)$ (s,t)-cuts from Type-2. In the following section, we provide a compact data structure that can answer query $Q(u,v)$ even when (u,v) is not necessarily an edge.

6 Compact data structures for reporting $(\lambda+1)$ (s,t)-cuts

In this section we address the very fundamental problem of reporting any $(\lambda+1)$ (s,t)-cut C for a given pair of vertices $\{u,v\}$ such that $u \in C$ and $v \in \overline{C}$, if exists. In order to answer the query it is sufficient to verify if v is separated by at least one of the cuts from $N_1(u)$. Note that in case of (s,t)-mincut, it suffices to store the $N_0(u)$ which turns out to be unique for each vertex u. This structure occupies $\mathcal{O}(n^2)$ space and achieves $\mathcal{O}(|C|)$ time for reporting the (s,t)-mincut C such that $u \in C$ and $v \in \overline{C}$. Unfortunately, $N_1(u)$ can have more than one elements (as shown in Figure 4 that both A and B belong to $N_1(\nu)$). This is because unlike for (s,t)-mincuts, the set of all $(\lambda+1)$ (s,t)-cuts that keep a vertex u on the side of s is not closed under intersection and union operations.

In order to design compact data structure for reporting $(\lambda+1)$ (s,t)-cut, as discussed in Section 5.1.1, we work on each $(\lambda+1)$ (s,t)-class. Let $\mathcal{W}$ be a $(\lambda+1)$ (s,t)-class in graph G and $u, v \in \mathcal{W}$. Although $N_1(u)$ can have multiple elements, it can be shown using sub-modularity of cuts (Lemma 9) that $N_1(u,v)$ is unique (proof in Appendix J). However, this fact alone guarantees $\mathcal{O}(|\mathcal{W}|^3)$ space data structure for determining the existence of $N_1(u,v)$ for any pair of vertices $u, v \in \mathcal{W}$. In order to achieve more compact data structure we explore how $N_1(u,v)$ is related to $N_1(u,w)$ for any $w \in \mathcal{W}$. The following lemma provides an important insight into this relationship.

▶ **Lemma 32.** *If $N_1(u,v) \neq N_1(u,w)$ for any $\{u,v,w\}$ in $\mathcal{W}$, then $\mathcal{W} \subseteq N_1(u,v) \cup N_1(u,w)$.*

Proof. The proof is by contradiction. Let $C = N_1(u,v)$ and $C' = N_1(u,w)$. Assume that there is a vertex in $\mathcal{W}$ which is also in $\overline{C \cup C'}$. In that case both $C \cap C'$ and $C \cup C'$ subdivide $\mathcal{W}$; therefore, capacity of each of them is strictly larger than λ. So it immediately follows from sub-modularity of cuts (Lemma 9) that $c(C \cap C')$ is $(\lambda+1)$. Therefore we have a $(\lambda+1)$ (s,t)-cut $C \cap C'$ which is a proper subset of at least one of C and C'. Therefore, C or C' fails to satisfy Definition 6 – a contradiction. ◀

Now let $N_1(u) = \{C_1, C_2, \ldots, C_l\}$. Consider any vertex $v \in \mathcal{W}$. If v belongs to $\bigcap_{i=1}^{l} C_i$, then $N_1(u,v)$ does not exist; otherwise, Lemma 32 implies that v is separated from u by exactly one of the cuts from $N_1(u)$. As a result the sets $\overline{C_i} \cap \mathcal{W}$ for each $i \in [l]$ are disjoint. In order to determine the position of any other vertex, $w \in \mathcal{W}$, with respect to $N_1(u,v)$, we formulate the following query.

$$\textsc{belong}(u,v,w) = \begin{cases} 1 & \text{if } N_1(u,v) \text{ exists and } w \in N_1(u,v) \\ 0 & \text{otherwise.} \end{cases}$$

For each vertex $x \in \mathcal{W} \setminus \{u\}$, if $x \in \overline{C_i}$ for any $i \in [l]$ then mark x with label i. Now if label of v and w are same, then w is not present in $N_1(u,v)$, otherwise w belongs to $N_1(u,v)$. In this way we can answer query $\textsc{belong}(u,v,w)$ in $\mathcal{O}(1)$ time. Therefore, the following lemma is immediate.

▶ **Lemma 33.** *Let $\mathcal{W}$ be a $(\lambda+1)$ (s,t)-class and $u \in \mathcal{W}$. There exists an $\mathcal{O}(|\mathcal{W}|)$ size data structure $\mathcal{N}_{\lambda+1}(u)$ that can determine in $\mathcal{O}(k)$ time whether there exists a $(\lambda+1)$ (s,t)-cut C such that $u \in C$ and $v_1, \ldots, v_k \in \overline{C}$ for any $v_1, \ldots, v_k \in \mathcal{W}$. If C exists, the data structure can output $\overline{C} \cap \mathcal{W}$ in $\mathcal{O}(|\overline{C} \cap \mathcal{W}|)$ time.*

Moreover, given vertices $u, v_1, \ldots, v_k \in \mathcal{W}$, $\mathcal{N}_{\lambda+1}(u)$ can be used to report a $(\lambda+1)$ (s,t)-cut C, if exists, in $\mathcal{O}(|\overline{C}|)$ time using an auxiliary $\mathcal{O}(n)$ space topological ordering of $\mathcal{D}_\lambda$ of G such that $u \in C$ and $v_1, \ldots, v_k \in \overline{C}$ (see Appendix K). Now based on Lemma 33 we construct the following data structure for $(\lambda+1)$ (s,t)-class $\mathcal{W}$.

Description of $\mathcal{N}_{\lambda+1}$: It consists of $\mathcal{N}_{\lambda+1}(u)$ for each u in $\mathcal{W}$.

$\mathcal{N}_{\lambda+1}$ for a $(\lambda+1)$ (s,t)-class $\mathcal{W}$ occupies $\mathcal{O}(|\mathcal{W}|^2)$ space. Each $(\lambda+1)$ (s,t)-class is disjoint from each other. Hence by constructing $\mathcal{N}_{\lambda+1}$ for each $(\lambda+1)$ (s,t)-class of G we get an $\mathcal{O}(n^2)$ size data structure and it completes the proof of Theorem 3. We complement this result with a conditional lower bound of $\tilde{\Omega}(n^2)$ space based on Conjecture 4 (refer to Appendix R).

7 Dual edge sensitivity oracle for (s,t)-mincuts

In this Section we shall present an oracle that can efficiently report (s,t)-mincut upon the failure of a pair of edges in graph G. We say that an edge (u,v) belongs to a $(\lambda+1)$ (s,t)-class $\mathcal{W}$ if both $u, v \in \mathcal{W}$.

7.1 Handling dual edge failures

Consider the failure of two edges $e = (x,y), e' = (x',y')$ in G. Suppose at least one of $\{e, e'\}$ does not belong to any $(\lambda+1)$ (s,t)-class. In this case the value of (s,t)-mincut decreases by at least 1 if $N_0(x,y)$ or $N_0(x',y')$ exists. The value of (s,t)-mincut decreases by exactly 2 if and only if both e, e' are contributing to a single (s,t)-mincut. To determine the existence of

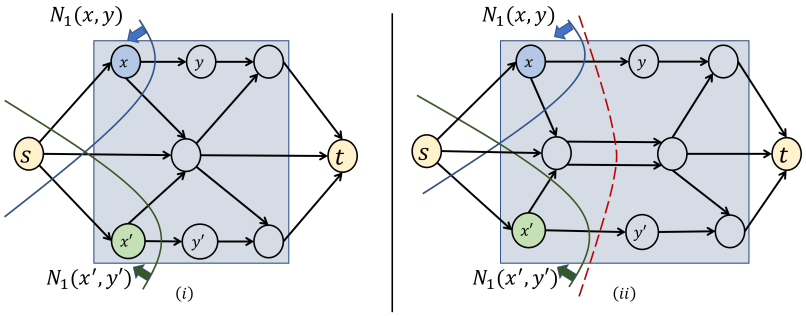

Figure 5 Graph (i) has no $\lambda+1$ (s,t)-cut containing edges (x,y) and (x',y'), but Graph (ii) does have one such cut shown by dashed curve.

such an (s,t)-mincut, since (s,t)-mincuts are closed under union operation, it is sufficient to verify whether $y \notin N_0(x',y')$ and $y' \notin N_0(x,y)$. So we construct a data structure, denoted by $\mathcal{N}_\lambda$, which consists of $N_0(u)$ for each vertex u of G. This $\mathcal{O}(n^2)$ space data structure achieves $\mathcal{O}(1)$ time to report the resulting value of (s,t)-mincut on failure of $\{e,e'\}$ in this case. (see Appendix L).

If e and e' belong to distinct $(\lambda+1)$ (s,t)-classes, then it follows from Lemma 12 that the (s,t)-mincut value remains unchanged. Let us consider case when both e and e' belong to the same $(\lambda+1)$ (s,t)-class, say $\mathcal{W}$. It follows as a simple corollary of Lemma 30 that we just need to verify if there is a $(\lambda+1)$ (s,t)-cut in $G(\mathcal{W})$ in which e and e' are contributing. Note that the only $(\lambda+1)$ (s,t)-class of $G(\mathcal{W})$ is $\mathcal{W}$.

Consider the data structure $\mathcal{N}_{\lambda+1}$ for $\mathcal{W}$ in $G(\mathcal{W})$. For the existence of a $(\lambda+1)$ (s,t)-cut in which both e and e' are contributing, observe that a necessary condition is that $y \notin N_1(x',y')$ and $y' \notin N_1(x,y)$. These conditions can be verified using $\mathcal{N}_{\lambda+1}$ in $\mathcal{O}(1)$ time. Upon checking these conditions, a natural approach would be to explore whether the union of these two cuts is a $(\lambda+1)$ (s,t)-cut. Unfortunately, we can not infer anything conclusively from the union of these cuts as illustrated by the two graphs in Figure 5. In both these graphs, $N_1(x,y) \cup N_1(x',y')$ is not a $(\lambda+1)$ (s,t)-cut . However, for graph (i), no $(\lambda+1)$ (s,t)-cut exists that contains the two edges; for graph (ii), there is still a $(\lambda+1)$ (s,t)-cut containing the two edges. Note that in these graphs, $N_1(x,y)$ and $N_1(x',y')$ are pairs of $(\lambda+1)$ (s,t)-cuts from Type-2.

Looking at this hurdle carefully, we get the following insight. Since y, y' both lie outside $N_1(x',y') \cup N_1(x,y)$, hence $c(N_1(x,y) \cup N_1(x',y'))$ has to be $> \lambda$. Therefore, if $c(N_1(x,y) \cap N_1(x',y'))$ is also $> \lambda$, then using sub-modularity of cuts (Lemma 9), their union is bound to be a $(\lambda+1)$ (s,t)-cut and this will serve our purpose. Unfortunately, $G(\mathcal{W})$ does not ensure that $c(N_1(x,y) \cap N_1(x',y'))$ is greater than λ as shown in Figure 5.

In order to materialize the above insight, we use covering technique (Theorem 18) to build the pair of graphs $\{G(\mathcal{W})^I, G(\mathcal{W})^U\}$ that partition all $(\lambda+1)$ (s,t)-cuts of $G(\mathcal{W})$. It follows from Theorem 18(2) that the capacity of the intersection (likewise union) of each pair of $(\lambda+1)$ (s,t)-cut in $G(\mathcal{W})^I$ (likewise $G(\mathcal{W})^U$) is greater than λ. This is because the only (s,t)-mincut of $G(\mathcal{W})^I$ is $\overline{T'}$ and the only (s,t)-mincut of $G(\mathcal{W})^U$ is S'. Theorem 18(1) states that $\{G(\mathcal{W})^I, G(\mathcal{W})^U\}$ covers all the $(\lambda+1)$ (s,t)-cuts of $G(\mathcal{W})$. Therefore, if $y' \notin N_1(x,y)$ and $y \notin N_1(x',y')$ in $G(\mathcal{W})^I$ or $x \notin N_1(y',x')$ and $x' \notin N_1(y,x))$ in $(G(\mathcal{W})^U)^\mathcal{T}$ then there is a $(\lambda+1)$ (s,t)-cut in $G(\mathcal{W})$ in which e, e' are contributing. Now we shall establish the converse of this assertion.

Suppose there exists a $(\lambda+1)$ (s,t)-cut C in $G(\mathcal{W})$ to which both edges (x,y) and

(x', y') are contributing. Without loss of generality assume that C is present in $G(\mathcal{W})^I$. It can be observed that the cuts $N_1(x, y)$ and $N_1(x', y')$ in $G(\mathcal{W})^I$ are subsets of C. Hence $y \notin N_1(x', y')$ and $y' \notin N_1(x, y)$ in $G(\mathcal{W})^I$. If C is present in $G(\mathcal{W})^U$, exactly the same analysis can be carried out on $(G(\mathcal{W})^U)^{\mathcal{T}}$. So we can state the following lemma.

▶ **Lemma 34.** *A pair of edges $e = (x, y)$, $e' = (x', y')$ from $(\lambda + 1)$ class $\mathcal{W}$ are outgoing edges of a $(\lambda + 1)$ (s, t)-cut in $G(\mathcal{W})$ if and only if (i) $y \notin N_1(x', y')$ and $y' \notin N_1(x, y)$ in $G(\mathcal{W})^I$ or (ii) $x \notin N_1(y', x')$ and $x' \notin N_1(y, x)$ in $(G(\mathcal{W})^U)^{\mathcal{T}}$.*

Using the data structure $\mathcal{N}_{\lambda+1}$, it requires a constant number of BELONG queries to verify the conditions mentioned in Lemma 34. Therefore, the data structure $\mathcal{F}$ for dual edge failure is as follows.

$\mathcal{F}$ consists of the following data structures:
- $\mathcal{N}_\lambda$ for graph G.
- $\mathcal{N}_{\lambda+1}$ for $G(\mathcal{W})^I$ and $(G(\mathcal{W})^U)^{\mathcal{T}}$ for each $(\lambda + 1)$ (s, t)-class $\mathcal{W}$.

It can be observed that the resulting (s, t)-mincut value can be reported in $\mathcal{O}(1)$ time upon failure of any pair of edges from G using $\mathcal{F}$ (Handling of dual edge insertion is in Appendix M). We summarize the results of this section in the following theorem.

▶ **Theorem 35.** *A directed multi-graph $G = (V, E)$, on $|V| = n$ vertices and $|E| = m$ edges with a designated source vertex s and a designated sink vertex t, can be preprocessed for constructing $\mathcal{O}(n^2)$ space Oracle $\{\mathcal{F}, \mathcal{I}\}$ that takes $\mathcal{O}(1)$ time to report the value of resultant (s, t)-mincut upon*
1. *failure of any given pair of edges $(x, y), (x', y') \in E$ using $\mathcal{F}$, or*
2. *insertion of any given pair of edges $(x, y), (x', y') \in V \times V$ using $\mathcal{I}$.*

▶ Remark 36. For a $(\lambda + 1)$ (s, t)-class $\mathcal{W}$ and any pair of disjoint subsets $A, B \subset \mathcal{W}$, $\mathcal{F}$ can determine in $\mathcal{O}(|A||B|)$ time if there exists a $(\lambda + 1)$ (s, t)-cut C such that $A \subseteq C$ and $B \subseteq \overline{C}$. If C exists, then it is possible to report C using $\mathcal{F}$ in $\mathcal{O}((|A| + |B|)|\mathcal{W}| + |\overline{C}|)$ time. (see Appendix N)

8 Conditional lower bound for dual edge sensitivity for (s, t)-mincuts

The problem of reachability in directed graph is as follows – Given a simple directed graph G with n vertices and m edges, preprocess it to form a data structure which can efficiently report if a given vertex v is reachable from another vertex u. The reachability in G is same as reachability in G_{SCC}, a directed acyclic graph which can be obtained by contracting each of the Strongly Connected Components to a single vertex. Henceforth, we shall assume that G is a directed acyclic graph. We transform the directed acyclic graph G into a graph $\mathcal{D}$ as follows.

Construction of $\mathcal{D}$: Create two additional vertices, namely s and t. Suppose Δ_v denotes the difference in the number of incoming and outgoing edges of any vertex v of G. For each vertex v in G, if $\Delta_v > 0$ we add Δ_v edges from v to t. Likewise, if $\Delta_v < 0$ we add Δ_v edges from s to v. Lastly, add two additional edge(s) from s to v and v to t for all v in G. Observe that the number of edges in this graph is only $\mathcal{O}(m)$. Thus, we state the following lemma.

▶ **Lemma 37.** *For a directed graph G with n vertices and m edges, there exists a directed acyclic multi-graph $\mathcal{D}$ with $\mathcal{O}(m)$ edges and $n + 2$ vertices such that a vertex v is reachable from a vertex u in G if and only if vertex v is reachable from vertex u in $\mathcal{D}$.*

The graph $\mathcal{D}$, that we have constructed, has a very interesting property. $\mathcal{D}$ is identical to the DAG $\mathcal{D}_\lambda$ that stores all (s,t)-mincuts in graph $\mathcal{D}$. We crucially exploit this property to derive an equivalence between reachability queries in graph $\mathcal{D}$ and dual edge failure (or insertion) query for (s,t)-mincut in $\mathcal{D}$. Since, the reachability structure of $\mathcal{D}$ and G is identical, we state the following lemma. (detailed proof in Appendix O).

▶ **Lemma 38.** *Let G be a directed graph. A vertex v is reachable from a vertex u in G if and only if the value of (s,t)-mincut reduces by exactly 1 on removal of the edges $\{(s,u),(v,t)\}$ from graph $\mathcal{D}$ which is obtained from G using Lemma 37.*

Proof sketch. We know that $\mathcal{D}$ is same as $\mathcal{D}_\lambda$ of $\mathcal{D}$. So, removal of any edge from $\mathcal{D}$ reduces value of (s,t)-mincut by 1. Any (s,t)-cut in which both edges, $\{(s,u),(v,t)\}$, are contributing cannot be 1-transversal because of the u to v path. Every (s,t)-mincut is 1-transversal (Lemma 20), therefore, there cannot exist (s,t)-mincut in which both edges are contributing.

If there is no path from u to v, then both edges contribute to (s,t)-cut $C = \overline{R(\{u\}) \cup T}$; where $R(\{u\})$ defines the set of vertices reachable from u. We can show that C is also an (s,t)-mincut. Thus, upon removal of edges $\{(s,u),(v,t)\}$ the value of (s,t)-mincut reduces by 2. ◀

Using Conjecture 4 and Lemma 38 we state the following conditional lower bound.

▶ **Theorem 39.** *Assuming Directed Reachability Hypothesis holds, any data structure that can report the value of (s,t)-mincut for a designated source s and a designated sink t upon failure or addition (refer to Appendix P) of any pair of edges in a directed multi-graph with n vertices and m edges must either use $\tilde{\Omega}(n^2)$ space, or $\tilde{\Omega}(m)$ time.*

References

1. Surender Baswana and Abhyuday Pandey. Sensitivity oracles for all-pairs mincuts. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 581–609. SIAM, 2022. doi:10.1137/1.9781611977073.27.
2. András A. Benczúr. A representation of cuts within 6/5 times the edge connectivity with applications. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 92–102. IEEE Computer Society, 1995. doi:10.1109/SFCS.1995.492466.
3. Keerti Choudhary. An optimal dual fault tolerant reachability oracle. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 130:1–130:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.130.
4. Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008. doi:10.1137/S0097539705429847.
5. 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.
6. Yefim Dinitz. Maintaining the 4-edge-connected components of a graph on-line. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993, Proceedings*, pages 88–97. IEEE Computer Society, 1993. doi:10.1109/ISTCS.1993.253480.
7. 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 Frank Thomson Leighton

and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 509–518. ACM, 1995. doi:10.1145/225058.225268.

8 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.

9 Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 506–515. SIAM, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496826.

10 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.

11 Harold N. Gabow. Efficient splitting off algorithms for graphs. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 696–705. ACM, 1994. doi:10.1145/195058.195436.

12 Zvi Galil and Giuseppe F. Italiano. Maintaining the 3-edge-connected components of a graph on-line. *SIAM J. Comput.*, 22(1):11–28, 1993. doi:10.1137/0222002.

13 Naveen Garg and Vijay V. Vazirani. A polyhedron with all$s$ - t cuts as vertices, and adjacency of cuts. *Math. Program.*, 70:17–25, 1995. doi:10.1007/BF01585926.

14 Isaac Goldstein, Tsvi Kopelowitz, Moshe Lewenstein, and Ely Porat. Conditional lower bounds for space/time tradeoffs. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures - 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 - August 2, 2017, Proceedings*, volume 10389 of *Lecture Notes in Computer Science*, pages 421–436. Springer, 2017. doi:10.1007/978-3-319-62127-2_36.

15 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019. doi:10.1145/3274663.

16 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.52.

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

18 Jason Li. Deterministic mincut in almost-linear time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 384–395. ACM, 2021. doi:10.1145/3406325.3451114.

19 Jason Li and Debmalya Panigrahi. Deterministic min-cut in poly-logarithmic max-flows. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 85–92. IEEE, 2020. doi:10.1109/FOCS46700.2020.00017.

20 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.

21 Merav Parter and David Peleg. Sparse fault-tolerant BFS structures. *ACM Trans. Algorithms*, 13(1):11:1–11:24, 2016. doi:10.1145/2976741.

22 Mihai Patrascu. Unifying the landscape of cell-probe lower bounds. *SIAM J. Comput.*, 40(3):827–847, 2011. doi:10.1137/09075336X.

23 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.

24 Vijay V. Vazirani and Mihalis Yannakakis. Suboptimal cuts: Their enumeration, weight and number (extended abstract). In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*, pages 366–377. Springer, 1992. doi:10.1007/3-540-55719-9_88.

A Proof of Lemma 10

Proof. It follows from sub-modularity of cuts (Lemma 9), $c(A \cap B) + c(A \cup B) \leq 2\lambda$. Since $c(A \cap B), c(A \cup B) \geq \lambda$, hence $A \cap B$ and $A \cup B$ are (s,t)-mincuts.

Notice that A and B splits V into four subsets. We refer to Figure 1 for the illustration of these regions and the edges among them.

Since $c(A \cap B) = c(B) = c(A)$, therefore $\alpha_1 = \beta_1 + \gamma_2$ and $\alpha_2 = \beta_2 + \gamma_1$. From these two equations we get, $\alpha_1 + \alpha_2 = \beta_1 + \gamma_2 + \beta_2 + \gamma_1$. Adding δ on each side of the equation, $\gamma_1 + \gamma_2 = \lambda - \lambda = 0$ since $c(A \cup B) = \alpha_1 + \alpha_2 + \delta = \lambda$ and $c(A \cup B) = \beta_1 + \beta_2 + \delta = \lambda$. ◀

B Proof of Lemma 12

Proof. Let C be a $(\lambda + k)$ (s,t)-cut which subdivides two $(\lambda + k)$ (s,t)-classes μ and ν. Note that μ and ν must be separated by a (s,t)-cut C' of capacity $(\lambda + k - 1)$. By sub-modularity of cuts (Lemma 9) we have, $c(C \cap C') + c(C \cup C') \leq 2\lambda + 2k - 1$. Since C subdivides both of them, therefore $c(C \cap C'), c(C \cup C') \geq \lambda + k$. Hence $c(C \cap C') + c(C \cup C') \geq 2\lambda + 2k$, a contradiction. ◀

C Proof of Lemma 17

Proof. Let C be a (s,t)-cut of capacity λ' in H. The vertex x is either in C or in $\overline{C}$. Without loss of generality assume that $x \in C$. In graph H^I this cut C is preserved since the infinite capacity edge is from s to x. Evidently its capacity also remains unchanged in H^I.

Without loss of generality let C be a (s,t)-cut of capacity λ' in H^I. It follows from the construction of H^I, $x \in C$. Therefore, removal of the infinite capacity edge, say e_∞, from H^I does not affect the (s,t)-cut C because e_∞ is not a contributing edge of C. Hence C is (s,t)-cut of capacity λ' in H. ◀

D Acyclicity of $\mathcal{D}_\lambda$

In order to prove acyclicity we require the following lemma.

▶ **Lemma 40** (transitivity Property). *Let $\{x,u,v\}$ be any three nodes of $\mathcal{D}_\lambda$ such that there exists a directed path $\langle x,(x,u),u,(u,v),v\rangle$. Let A,B be two (s,t)-mincuts containing edges (x,u) and (u,v) respectively, then $x \in B$.*

Proof. Suppose $x \notin B$. In that case edge (x,u) is from $A \setminus B$ to $B \setminus A$, which leads to contradiction because of Lemma 10. ◀

An induction on the length of a path leads to the following corollary of Lemma 40.

▶ **Corollary 41.** *Let $P = \langle v_0, (v_0, v_1), v_1, \ldots, (v_{l-1}, v_l), v_l \rangle$ be a path in $\mathcal{D}_\lambda$. Then each node v_i, $0 \leq i \leq l-1$ in P belongs to the (s,t)-mincut containing edge (v_{l-1}, v_l).*

Here we state the proof for acyclicity.

▶ **Lemma 42** (acyclicity property)**.** *$\mathcal{D}_\lambda$ is a directed acyclic graph.*

Proof. We give a proof by contradiction. Assume to the contrary that $\mathcal{D}_\lambda$ is not acyclic. Then there exists cycle $\langle v_0, e_1, v_1, \ldots, e_r, v_r, e_0, v_0 \rangle$ in $\mathcal{D}_\lambda$. Let us look at an edge $e_{i-1} = (v_{i-2}, v_{i-1})$, $0 \leq i \leq r$. Since the orientation is from v_{i-2} to v_{i-1}, it follows from the construction of $\mathcal{D}_\lambda$ that there is a (s,t)-mincut B such that $v_{i-2} \in B$ and $v_{i-1} \in \overline{B}$. Also we have a directed path P from v_i to v_{i-2} since it is a directed cycle. It follows from Corollary 41 that all the nodes in the directed path P belongs to B, hence $v_i \in B$ also. Let us consider edge e_i. Again it follows from the construction of $\mathcal{D}_\lambda$ that there is a (s,t)-mincut A such that $v_{i-1} \in A$ and $v_i \in \overline{A}$. Therefore, we have an edge (v_{i-1}, v_i) between $A \setminus B$ and $B \setminus A$, a contradiction due to Lemma 10. ◀

E Proof of Lemma 20 : 1-transversality property of (s,t)-mincuts

In G_λ, we show that S and (complement of) T defines (s,t)-mincuts of G using the following lemma.

▶ **Lemma 43.** *Each edge entering into S or leaving T in G_λ is an inverted edge.*

Proof. We give a proof by contradiction. Let $e = (u, v)$ be an edge of G_λ such that $v \in S$ and $u \notin S$. If $v = s$ then trivially e is inverted. So we assume that $v \neq s$. Assume to the contrary that the edge e is not inverted. Then there exists a (s,t)-mincut C such that $u \in C$ and $v \in \overline{C}$. Since s and v are separated by C, therefore v and s cannot belong to the same $(\lambda+1)$ (s,t)-class S, a contradiction. ◀

The following corollary is immediate using Lemma 43 as (s,t)-mincuts are closed under intersection and union.

▶ **Corollary 44.** *S as well as complement of T are (s,t)-mincuts.*

For other nodes except S and T of G_λ, we establish the following important property.

▶ **Lemma 45.** *Let $\mu \neq \{S, T\}$ be a node of G_λ. If we ignore inverted edges, the number of incoming edges of μ is the same as the number of outgoing edges of μ.*

Proof. Suppose μ has α incoming and β outgoing edges which are not inverted. For each edge $e_i^{in} = (u_i, \mu)$ of α edges there is a (s,t)-mincut C_i^{in} such that $u_i \in C_i^{in}$ and $\mu \in \overline{C_i^{in}}$ by definition, $i \in [\alpha]$. We know that $\bigcup_{i=1}^{\alpha} C_i^{in}$ is a (s,t)-mincut. This directly implies that $\alpha \leq \beta$. Similarly each outgoing edge $e_i^{out} = (\mu, v_i)$ of β edges there is a (s,t)-mincut C_i^{out} such that $\mu \in C_i^{out}$ and $v_i \in \overline{C_i^{out}}$. In a similar way we can show that $\bigcap_{i=1}^{\beta} C_i^{out}$ is a (s,t)-mincut, implies $\alpha \geq \beta$. Thus $\alpha = \beta$. ◀

Now we prove the 1-transversality property.

Proof. Without causing ambiguity we call a 1-transversal cut as a transversal cut in this proof. Let C be a (s,t)-mincut of G. Observe that C separates each $x \in C$ from each $y \in \overline{C}$. So, it follows from the construction of $\mathcal{D}_\lambda$ that C defines a (s,t)-cut in $\mathcal{D}_\lambda$. Assume that it is not transversal in $\mathcal{D}_\lambda$. In that case edge-set of C must be intersecting a directed path of

$\mathcal{D}_\lambda$ at least thrice. This implies there exists at least one edge (u, v) such that $u \in \overline{C}$ and $v \in C$. Since the orientation of edge (u, v) in $\mathcal{D}_\lambda$ is from u to v then there must exist another (s, t)-mincut C' of G such that $u \in C'$, $v \in \overline{C'}$. So we have $u \in C' \setminus C$, $v \in C \setminus C'$ and (u, v) is an edge, a contradiction from Lemma 10.

To prove the converse part, we state and prove the following assertion.

$\mathcal{A}(i)$: Each transversal cut C in $\mathcal{D}_\lambda$ consisting of i nodes is a (s, t)-mincut of G.

We now prove $\mathcal{A}(i)$ by induction on i. In the base case $i = 1$, S is the only transversal cut of $\mathcal{D}_\lambda$ and it is a (s, t)-mincut of G as stated in Corollary 44. Suppose $A(j)$ holds for each $j < i$.

Let C be a transversal cut C in $\mathcal{D}_\lambda$ consisting of i nodes. Since $\mathcal{D}_\lambda$ is a acyclic graph, there exists a topological ordering. Let us consider a node μ such that μ has the highest topological order and $\mu \in C$. Observe that all outgoing edges of μ are incident on $V \setminus C$. Therefore, removing node μ from C makes another transversal cut C' of $\mathcal{D}_\lambda$. Since C' consists of $i - 1$ nodes, it follows from induction hypothesis that C' is a (s, t)-mincut. An outgoing edge of μ in $\mathcal{D}_\lambda$ is not an outgoing edge of C in G if the edge is inverted. Similarly an incoming edge of μ in $\mathcal{D}_\lambda$ is not an outgoing edge of C' in G if it is an inverted edge. Therefore, it follows from Lemma 45 that the capacity of C is same as C'. Hence C is a (s, t)-mincut. ◀

F Single edge sensitivity : $\mathcal{O}(n)$ space, $\mathcal{O}(1)$ time

Given an edge e of G, if edge e is an outgoing edge of some (s, t)-mincut then value of (s, t)-mincut reduces by 1. The graph $\mathcal{D}_\lambda$ is acyclic as stated in Lemma 42. This ensures the existence of a topological ordering of the set of nodes of $\mathcal{D}_\lambda$. Let us consider a topological ordering τ of $\mathcal{D}_\lambda$ such that node S (likewise node T) is the lowest (likewise highest) topologically ordered node. Now we use the following lemma to construct our $\mathcal{O}(n)$ size data structure.

▶ **Lemma 46.** *An edge $e = (x, y)$ in G is an outgoing edge of an (s, t)-mincut if and only if the node containing x appears before the node containing y in topological ordering τ.*

Proof. If e is an outgoing edge of a (s, t)-mincut then it appears in $\mathcal{D}_\lambda$ with orientation unchanged. Hence the node containing x appears before the node containing y in each topological ordering of $\mathcal{D}_\lambda$.

Since x and y are appearing in different nodes, therefore the edge is appearing in $\mathcal{D}_\lambda$. Assume to the contrary that there is no (s, t)-mincut where edge e is an outgoing edge. Then there exists a (s, t)-mincut C such that $y \in C$ and $x \in \overline{C}$. Hence edge e is an inverted edge. Therefore, in $\mathcal{D}_\lambda$ the orientation of edge e is from y towards x. Evidently in each topological ordering of $\mathcal{D}_\lambda$ the node containing y appears before the node containing x, a contradiction to the premise. ◀

Using Lemma 46 and topological ordering τ we can report the value of (s, t)-mincut upon failure of any edge in $\mathcal{O}(1)$ time. Moreover if the value decreases then report the vertices belonging to those nodes which appear before the node containing x (including the node containing x) in the topological ordering τ as the resultant (s, t)-mincut because any prefix of τ is a (s, t)-mincut.

In case of insertion of an edge e if we can ensure that e contributes to each (s, t)-mincut then the value is of (s, t)-mincut is increased. Therefore the following fact is necessary as well as sufficient to verify whether (s, t)-mincut value increases upon insertion of an edge.

▶ **Fact F.1.** *Let the* $(\lambda+1)$ *(s,t)-class containing source s be $\mathcal{S}$ and* $(\lambda+1)$ *(s,t)-class containing sink t be $\mathcal{T}$. On insertion of an unit capacity edge $e=(x,y)$, the value of (s,t)-mincut increases by* 1 *if and only if $x \in \mathcal{S}$ and $y \in \mathcal{T}$.*

G Proof of Lemma 29 : Degenerate $(\lambda+1)$ (s,t)-cuts

In order to prove Lemma 29, we explore the relation between a $(\lambda+1)$ (s,t)-cut and a (s,t)-mincut in the following lemma.

▶ **Lemma 47.** *Suppose A, B be any two (s,t)-cut such that one of $\{A,B\}$ is a (s,t)-mincut and other is a $(\lambda+1)$ (s,t)-cut. Let γ_1 and γ_2 denotes the number of edges from $A \setminus B$ to $B \setminus A$ and from $B \setminus A$ to $A \setminus B$ respectively. If $c(A \cap B) = \lambda + p$ and $c(A \cup B) = \lambda + q$ for some $p, q \geq 0$, then $\gamma_1 + \gamma_2 = 1 - p - q$.*

The proof of Lemma 47 is along similar lines as the proof for Lemma 23. The following corollary is immediate from Lemma 47.

▶ **Corollary 48.** *Suppose A and B are two (s,t)-cuts such that one of $\{A,B\}$ is a (s,t)-mincut and other is a $(\lambda+1)$ (s,t)-cut.*

1. *The number of edges between $A \setminus B$ and $B \setminus A$ is exactly* 1 *if and only if $c(A \cap B) = c(A \cup B) = \lambda$.*
2. *The number of edges between $A \setminus B$ and $B \setminus A$ is at most* 1.

Now we give the proof for Lemma 29.

Proof. The (s,t)-cut C cannot have zero incoming edges, otherwise it is a transversal cut and from Lemma 20, C must be a (s,t)-mincut. Assume to the contrary that C has at least two incoming edges– $e=(\mu,\nu)$ and $e'=(\mu',\nu')$. It follows from construction of $\mathcal{D}_\lambda$ that there exists a (s,t)-mincut, say C_1, such that $\mu \in C_1$, $\nu \in \overline{C_1}$. Since $\nu \in C$ and $\mu \in \overline{C}$, therefore edge e is between $C \setminus C_1$ and $C_1 \setminus C$. It follows from Corollary 48(1) that $(C \cap C_1)$ and $(C \cup C_1)$ are (s,t)-mincuts. It is easy to observe that the position of edge e' is one of the following three –(i) $C_1 \setminus C$ to $C \cap C_1$, (ii) $C_1 \setminus C$ to $C \setminus C_1$ or (iii) $\overline{C \cup C_1}$ to $(C \cup C_1) \setminus (C_1 \setminus C) = C$. In case (i), (s,t)-mincut $C \cap C_1$ and in case (iii), the (s,t)-mincut $C \cup C_1$ has incoming edge, contradiction from Lemma 21. Otherwise in case (ii) we have two edges between $C \setminus C_1$ and $C_1 \setminus C$ – a contradiction from Corollary 48(2).

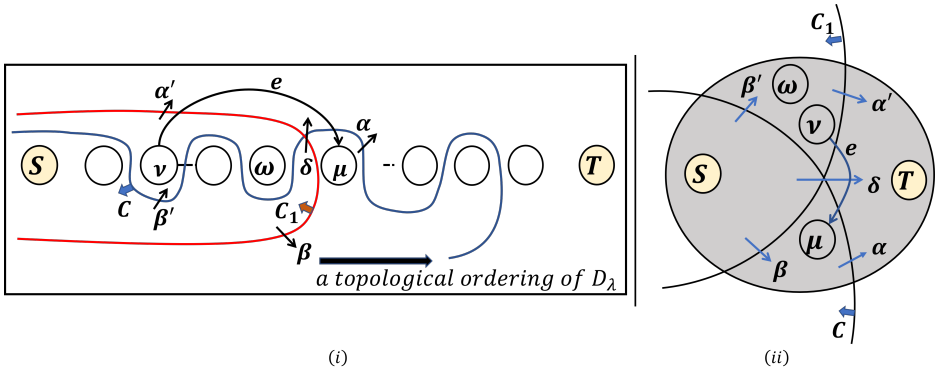

Figure 6 (i) (s,t)-cut C has exactly one incoming edge (ν,μ) (ii) The edges between different regions formed by C and C_1.

Let us now prove the converse part. We first define the following function f on edge set of $\mathcal{D}_\lambda$.

$$f((\nu,\mu),C) = \begin{cases} 1, & \text{if edge } (\nu,\mu) \text{ of } \mathcal{D}_\lambda \text{ is a contributing edge of } (s,t)\text{-cut } C \\ 0, & \text{otherwise.} \end{cases}$$

Let C be a (s,t)-cut with exactly one incoming edge (ν,μ) where ν and μ are nodes of $\mathcal{D}_\lambda$. Let us consider a topological ordering τ of $\mathcal{D}_\lambda$. In τ, node ν must appear before node μ. Since $\mu \in C$ and $\nu \in \overline{C}$, therefore in τ if we go towards S from μ, we must reach a node ω such that $\omega \notin C$. It follows from Lemma 20 that the prefix of τ till ω is a (s,t)-mincut, say C_1 (refer to Figure 6(i) for better understanding). We consider (s,t)-cut $C \cap C_1$. The only incoming edge of C is from ν to μ. Since $\mu \notin C \cap C_1$, $C \cap C_1$ is transversal cut and hence it is (s,t)-mincut as stated in Lemma 20. Also (s,t)-cut $C_1 \cup C$ is a transversal cut since both μ and ν belongs to $C_1 \cup C$. Again from Lemma 20 $(C_1 \cup C)$ is a (s,t)-mincut. Observe that the edge (ν,μ) is from $C_1 \setminus C$ to $C \setminus C_1$. Notice that there cannot be any edge from $C \setminus C_1$ to $C_1 \setminus C$, otherwise the topological ordering is not valid. The following equations are immediate from the Figure 6(ii).

$$c(C \cap C_1) = \beta' + \beta + \delta = \lambda \tag{5}$$
$$c(C \cup C_1) = \alpha' + \alpha + \delta = \lambda \tag{6}$$
$$c(C_1) = \alpha' + f((\nu,\mu),C_1) + \beta + \delta = \lambda, \tag{7}$$

We want to find out $c(C) = \beta' + f((\nu,\mu),C) + \alpha + \delta$. Let us consider the case when (ν,μ) is an inverted edge. Then $c(C_1) = \alpha' + \beta + \delta = \lambda$ because (ν,μ) becomes an incoming edge of C_1. Comparing this equation with Equation 5 we get, $\alpha' = \beta'$. Since edge (ν,μ) is inverted, (ν,μ) is an outgoing edge of C and hence $c(C) = \beta' + 1 + \alpha + \delta$. Hence replacing β' by α' we have $c(C) = \alpha + \alpha' + \delta + 1 = \lambda + 1$ from Equation 6.
Now we consider the case when (ν,μ) is not an inverted edge. In this case $c(C_1) = \alpha' + 1 + \beta + \delta = \lambda$ because (ν,μ) is an outgoing edge of C_1. Comparing this equation with Equation 6 we get, $\alpha = \beta + 1$. Since (ν,μ) is not inverted, hence $c(C) = \alpha + \beta' + \delta$ because (ν,μ) is an incoming edge of C. Replacing α by $\beta + 1$ we get, $c(C) = \beta + 1 + \beta' + \delta = \lambda + 1$ using Equation 5. This completes our proof. ◀

H Proof of Lemma 30

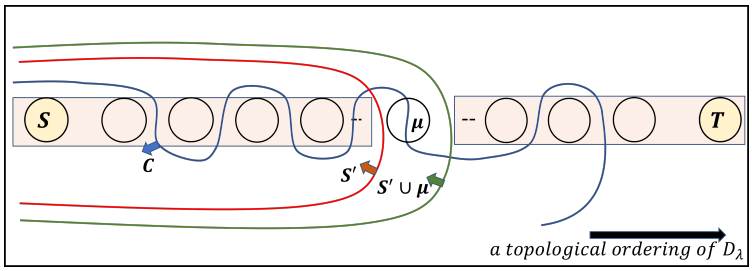

Figure 7 Illustration for the proof of Lemma 30.

Proof. Without loss of generality we assume that $\mu \neq \{S,T\}$. The following fact is immediate because it follows from Lemma 20 that every prefix of a topological ordering of $\mathcal{D}_\lambda$ is a (s,t)-mincut.

▶ **Fact H.1.** *The (s,t)-cut S' and $S' \cup \mu$ are (s,t)-mincuts of G.*

By sub-modularity of cuts (Lemma 9) the $S' \cup C$ and $(S' \cup \mu) \cap C$ are $(\lambda+1)$ (s,t)-cuts of G. The (s,t)-mincut S' contains all the nodes that precedes μ in the topological ordering. Hence our construction of $G(\mathcal{W})$ keeps the (s,t)-cuts S' and $(S' \cup \mu)$ intact in $G(\mathcal{W})$. Therefore the (s,t)-cut $(C \cup \{S'\}) \cap (S' \cup \mu)$ is a $(\lambda+1)$ (s,t)-cut (refer to Figure 7) of $G(\mathcal{W})$ using sub-modularity of cuts (Lemma 9). It is easy to observe that $(C \cup \{S'\}) \cap (S' \cup \mu)$ is the (s,t)-cut $\mathcal{W}_1 \cup \{S'\}$ of $G(\mathcal{W})$. Let $C' = S' \cup \mathcal{W}_1$ is $(\lambda+1)$ (s,t)-cut of $G(\mathcal{W})$. From Fact H.1, we know S' is a (s,t)-mincut of G. Therefore $S' \cup \mathcal{W}_1$ is also a $(\lambda+1)$ (s,t)-cut of G and it subdivides $\mathcal{W}$ into $(\mathcal{W}_1, \mathcal{W} \setminus \mathcal{W}_1)$. ◀

I Proof of Theorem 22 : $\Omega(n)$-transversality

We shall show that for any $k \geq 2$, there exists a directed multi-graph $\mathcal{G}$, with a designated source vertex s and designated sink vertex t, consisting of two (s,t)-mincuts of capacity $2k-1$ such that there exists a $(\lambda+1)$ (s,t)-cut in $\mathcal{G}$ whose edge-set intersects a simple path in $\mathcal{D}_{\lambda+1}(\mathcal{G})$ exactly $2k+1$ times. In order to construct $\mathcal{G}$, we first construct the following graph $\mathcal{H}$.

Construction of $\mathcal{H}$: $\mathcal{H}$ is a cycle $\langle a,b,c,d,e,f,a\rangle$ on 6 vertices such that each consecutive pair of vertices in path $\langle d,e,f,a,b\rangle$ has exactly two parallel edges.

Now we give the construction of graph $\mathcal{G}$.

Description of $\mathcal{G}$: $\mathcal{G}$ consists of $k-2$, $k \geq 2$, instances, say $\{H_2, \ldots, H_{k-1}\}$, of graph $\mathcal{H}$ with a source vertex s and a sink vertex t. The source s and sink t are connected to each instance H_i, $2 \leq i \leq k-1$, of $\mathcal{H}$ as follows. There are edges to d_i and e_i from s, and edges from a_i and b_i to t, for each $2 \leq i \leq k-1$. Consecutive graphs H_i and H_{i+1}, for $2 \leq i \leq k-2$, are connected by the following pair of edges (e_{i+1}, d_i) and (f_i, a_{i+1}).
There is a vertex, denoted by a_k, which is connected to s, t and H_{k-1} as follows. There is an edge from s, an edge to t and a pair of edges $(a_k, d_{k-1}), (f_{k-1}, a_k)$.
There is another component consisting of 6 vertices in $\mathcal{G}$ as follows. There is a cycle $\langle a_1, b_1, c_1, d_1, a_1\rangle$ on 4 vertices and there is also a path $\langle d_1, e_1, f_1, b_1\rangle$ on 4 vertices. Let us denote this component by H_1. H_1 is connected with s, t and H_2 as follows. There are edges from s to c_1 and d_1, edges to t from b_1 and c_1, and a pair of edges $(e_2, d_1), (f_1, a_2)$. We refer to Figure 8 for better understanding.

The value of (s,t)-mincut in graph $\mathcal{G}$ is $2k-1$. This is because for each H_i, $1 \leq i \leq k-1$, there are two incoming edges from s; and for a_k there is only one edge from s (similar arguments for edges incident on t). Also, every other (s,t)-cut value is greater than $2k-1$. Therefore, $\mathcal{G}$ has exactly two (s,t)-mincuts of capacity $2k-1$. These are $\{s\}$ and complement of $\{t\}$. The following lemma ensures that $\mathcal{G}$ is same as $\mathcal{D}_{\lambda+1}(\mathcal{G})$.

▶ **Lemma 49.** *For each edge (μ,ν) of graph $\mathcal{G}$ there is $(\lambda+1)$ (s,t)-cut $C_{(\mu,\nu)}$ such that $\mu \in C_{(\mu,\nu)}$ and $\nu \in \overline{C_{(\mu,\nu)}}$.*

Proof. Let us first consider the edges of each H_i, $2 \leq i \leq k-1$.

1. *edges (a_i, b_i):* Let C be a (s,t)-cut that keeps only vertex b_i on side of t. Except edge (b_i, t), each edge incident on t is an outgoing edge of C, and also both parallel edges (a_i, b_i) are outgoing edges of C. Hence the capacity of C is $2k-1-1+2 = 2k$.
2. *edge (b_i, c_i):* Let C be a (s,t)-cut that keeps only vertex c_i on side of t. Each edge incident on t is an outgoing edge of C, also edge (b_i, c_i) is an outgoing edge of C. Hence the capacity of C is $2k-1+1 = 2k$.

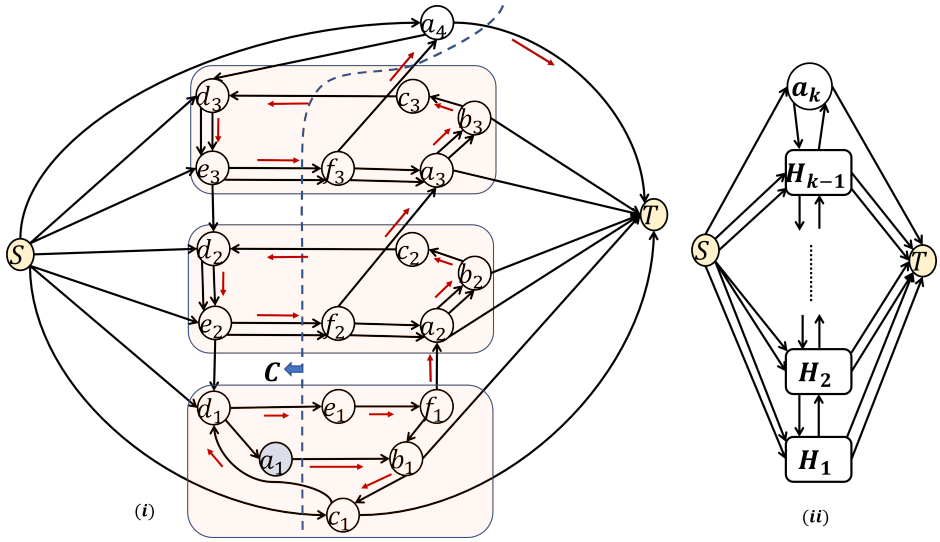

Figure 8 (i) $(\lambda+1)$ (s,t)-cut C intersecting path $\langle a_1,\ldots,f_1,a_2,\ldots f_2,\ldots,f_3,a_4,T\rangle$ in $\mathcal{D}_{\lambda+1}(\mathcal{G})$ 9 times when $\mathcal{G}$ has 2 instances of $\mathcal{H}$. (ii) Graph $\mathcal{G}$ with $k-2$ instances of $\mathcal{H}$.

3. *edge* (c_i,d_i)*:* Let C be a (s,t)-cut that keeps only vertex c_i on side of s. Each edge incident on s is an outgoing of C, also edge (c_i,d_i) is an outgoing edge. Hence the capacity of C is $2k-1+1=2k$.
4. *edges* (d_i,e_i)*:* Let C be a (s,t)-cut that keeps only vertex d_i on side of s. Except edge (s,d_i), each edge incident on s is an outgoing of C, and also both parallel edges (d_i,e_i) are outgoing edges of C. Hence the capacity of C is $2k-1-1+2=2k$.
5. *edges* (e_i,f_i)*:* Let C be a (s,t)-cut that keeps only two vertices $\{d_i,e_i\}$ on side of s. Except edges (s,d_i) and (s,e_i), each edge incident on s is an outgoing of C, and three outgoing edges of vertex e_i (includes edge (e_i,d_{i-1})) are also outgoing edges of C. Hence the capacity of C is $2k-1-2+3=2k$.
6. *edges* (f_i,a_i)*:* Let C be a (s,t)-cut that keeps only two vertices $\{a_i,b_i\}$ on side of t. Except two edges (a_i,t) and (b_i,t), each edge incident on t is an outgoing of C, and also three incoming edges of a_i (includes edge (f_{i-1},a_i)) are outgoing edges of C. Hence the capacity of C is $2k-1-2+3=2k$.

Now we consider the two edges incident on a_k.

1. *edge* (a_k,d_{k-1})*:* Let C be a (s,t)-cut that keeps only vertex a_k on side of s. Except edge (s,a_k), each edge incident on s is an outgoing of C, and also edges (a_k,d_{k-1}) and (a_k,t) are outgoing edges of C. Hence the capacity of C is $2k-1-1+2=2k$.
2. *edge* (f_{k-1},a_k)*:* Let C be a (s,t)-cut that keeps only vertex a_k on side of t. Except edge (a_k,t), each edge incident on t is an outgoing of C, and also edges (s,a_k) and (f_{k-1},a_k) are outgoing edges of C. Hence the capacity of C is $2k-1-1+2=2k$.

Finally we consider the edges of H_1.

1. *edge* (a_1,b_1)*:* Let C be a (s,t)-cut that keeps only vertices $\{a_1,d_1\}$ on side of s. Except edge (s,d_1), each edge incident on s is an outgoing of C, and also edges (d_1,e_1) and (a_1,b_1) are outgoing edges of C. Hence the capacity of C is $2k-1-1+2=2k$. The edge (d_1,e_1) is also an outgoing edge of C.
   If we include vertex e_1 in C to define a new (s,t)-cut C', then edge (e_1,f_1) is an outgoing edge of C'. C' is also a (s,t)-cut of capacity $2k$ because instead of edge (d_1,e_1), edge (e_1,f_1) is contributing.

In a similar way if we exclude vertex a_1 from C to define a new (s,t)-cut C'', then edge (d_1, a_1) is an outgoing edge of C''. C'' is also a (s,t)-cut of capacity $2k$ because instead of edge (a_1, b_1), the edge (d_1, a_1) is contributing.

2. *edge* (f_1, b_1)*:* Let C be a (s,t)-cut that keeps only vertex b_1 on side of t. Except edge (b_1, t), each edge incident on t is an outgoing of C, and also edges (a_1, b_1) and (f_1, b_1) are outgoing edges of C. Hence the capacity of C is $2k - 1 - 1 + 2 = 2k$.
3. *edge* (c_1, d_1)*:* Let C be a (s,t)-cut that keeps only vertex c_1 on side of s. Except edge (s, c_1), each edge incident on s is an outgoing of C, and also edges (c_1, d_1) and (c_1, t) are outgoing edges of C. Hence the capacity of C is $2k - 1 - 1 + 2 = 2k$.
4. *edge* (b_1, c_1)*:* Let C be a (s,t)-cut that keeps only vertex c_1 on side of t. Except edge (c_1, t), each edge incident on t is an outgoing of C, and also edges (b_1, c_1) and (s, c_1) are outgoing edges of C. Hence the capacity of C is $2k - 1 - 1 + 2 = 2k$.

Therefore, we showed that for each edge (μ, ν) of graph $\mathcal{G}$ there is $(\lambda + 1)$ (s,t)-cut $C_{(\mu,\nu)}$ such that $\mu \in C_{(\mu,\nu)}$ and $\nu \in \overline{C_{(\mu,\nu)}}$. ◀

It follows from Lemma 49 that $\mathcal{D}_{\lambda+1}(\mathcal{G})$ is same as $\mathcal{G}$. Let us now consider a (s,t)-cut C in $\mathcal{G}$ defined by the following set of vertices. For H_1, include vertices $\{a_1, d_1\}$, for H_i, $2 \leq i \leq k-1$, include vertices $\{d_i, e_i\}$ and finally include the vertex a_k. Observe that, (s, c_1) is an outgoing edge of C. Then for H_1, two edges (a_1, b_1) and (d_1, e_1) are outgoing from C. For each H_i, $2 \leq i \leq k-1$, two parallel edges from e_i to f_i are outgoing from C. Finally the edge (a_k, t) is outgoing. Therefore, the capacity of C is $1 + 2 + 2(k-2) + 1 = 2k$, thus a $(\lambda + 1)$ (s,t)-cut of $\mathcal{G}$.

There is a path $\mathcal{P} = \langle a_1, b_1, c_1, d_1, e_1, f_1, a_2, \ldots, f_2, \ldots, a_{k-1}, \ldots, f_{k-1}, a_k, T\rangle$ in $\mathcal{D}_{\lambda+1}(\mathcal{G})$ which intersects the edge-set of (s,t)-cut C exactly $2k+1$ times as follows. In H_1, edge-set of C intersects P thrice. Then at each H_i, $2 \leq i \leq k-1$, edge-set of C intersects path P twice. Finally two times at edge (f_{k-1}, a_k) and at edge (a_k, t). Hence the total number of intersections is $3 + 2(k-2) + 2 = 2k + 1$. Therefore, the following lemma is immediate.

▶ **Lemma 50.** *There is a graph $\mathcal{G}$, with a designated source vertex s and a designated sink vertex t, having two (s,t)-mincuts of capacity $2k-1$ such that in $\mathcal{D}_{\lambda+1}$ of $\mathcal{G}$ there exists a directed path $\mathcal{P}$ and a $(\lambda+1)$ (s,t)-cut C where edge-set of C intersects path $\mathcal{P}$ exactly $2k+1$ number of times.*

It follows from the construction of $\mathcal{G}$ that k can be $\Omega(n)$ if we use $\Omega(n)$ instances of $\mathcal{H}$ to construct $\mathcal{G}$. Therefore, Lemma 50 implies Theorem 22.

J Proof of uniqueness of nearest $(\lambda + 1)$ (s,t)-cut

Proof. We give a proof by contradiction. Assume to the contrary that there exists two nearest $(\lambda+1)$ (s,t)-cuts C and C' from u to v. Since $u \in C \cap C', C \cup C'$, $v \in \overline{C \cap C'}, \overline{C \cup C'}$ therefore the $c(C \cap C')$ and the $c(C \cup C')$ are strictly larger than λ. So it follows immediately from sub-modularity of cuts (Lemma 9) that $c(C \cap C') = c(C \cup C') = \lambda + 1$. That is, $C \cap C'$ is also a $(\lambda+1)$ (s,t)-cut. Since it is a proper subset of at least one of C or C', this contradicts the definition of nearest $(\lambda + 1)$ (s,t)-cut. ◀

K Reporting a $(\lambda + 1)$ (s,t)-cut using $\mathcal{N}_{\lambda+1}$

To efficiently report a $(\lambda + 1)$ (s,t)-cut the following lemma is used.

▶ **Lemma 51.** *Suppose τ be a topological ordering of $\mathcal{D}_\lambda$ where source node S has the smallest topological number and sink node T has the highest topological number. Let μ be the node corresponding to the $(\lambda+1)$ (s,t)-class $\mathcal{W}$. Let C be a (s,t)-mincut defined by the prefix of τ till before node μ. Then $C \cup (N_1(u,v) \cap \mathcal{W})$ is a $(\lambda+1)$ (s,t)-cut.*

Proof. Let C' be $N_1(u,v)$. Lemma 12 implies that C' cannot subdivide any other nodes and hence $c(C \cap C')$ must be λ. Since $C \cup C'$ is subdividing μ, therefore, it follows from sub-modularity of cuts (Lemma 9) that $(C \cup C')$ is a $(\lambda+1)$ (s,t)-cut. It is easy to observe that $(C \cup C') = C \cup (N_1(u,v) \cap \mathcal{W})$. ◀

Let τ be a topological ordering of $\mathcal{D}_\lambda$ where source node S has the smallest topological number and sink node T has the highest topological number. Let μ be the node corresponding to $\mathcal{W}$. Suppose a $(\lambda+1)$ (s,t)-cut subdivides $\mathcal{W}$ into $(\mathcal{W}_1, \mathcal{W} \setminus \mathcal{W}_1)$. We want to report a $(\lambda+1)$ (s,t)-cut $C_{\lambda+1}$ given the set $\mathcal{W} \setminus \mathcal{W}_1$. Let C be the (s,t)-mincut defined by the set of nodes from S to μ in τ. It follows from Lemma 51 that $\overline{C} \cup (\mathcal{W} \setminus \mathcal{W}_1)$ is a $(\lambda+1)$ (s,t)-cut. Therefore, we report this cut in $\mathcal{O}(|\overline{C} \cup (\mathcal{W} \setminus \mathcal{W}_1)|)$ time, which is $\mathcal{O}(\overline{C_{\lambda+1}})$.

L A step towards dual edge sensitivity data structure with $\mathcal{D}_\lambda$

We try to address the problem of dual edge sensitivity of (s,t)-mincuts using $\mathcal{D}_\lambda$. Unfortunately it will be shown that $\mathcal{D}_\lambda$ can be used only for a partial solution of the problem.

L.1 Partial solution to dual edge failure

Let $e = (x,y)$ and $e' = (x',y')$ be any pair of edges of graph G. We know how to determine if either e or e' is contributing to a (s,t)-mincut. Now we have to determine if there is a single (s,t)-mincut C from which both edges e and e' are outgoing as stated in Fact 1.1. In this case it is necessary that $y' \notin N_0(x,y)$ and $y \notin N_0(x',y')$. Also, we know that the union of both the nearest (s,t)-mincuts is also a (s,t)-mincut. Since $\{y,y'\}$ are not in any of the nearest (s,t)-mincuts, hence they are also not in their union. Therefore, the following lemma is a necessary and sufficient condition.

▶ **Lemma 52.** *A pair of edges $e = (x,y)$, $e' = (x',y')$ are contributing to (s,t)-mincut $N_0(x,y) \cup N_0(x',y')$ if and only if $y' \notin N_0(x,y)$ and $y \notin N_0(x',y')$.*

Let us consider a vertex $u \in V$. It is easy to show that $N_0(u)$ is unique because if there are multiple then their intersection becomes a (s,t)-mincut which leads to contradiction by Definition 5. Therefore, $N_0(u) = N_0(u,v)$ for any vertex $v \in V$. Now a data structure is required that can efficiently determine if a vertex w belongs to a $N_0(u)$. Here we construct the data structure, denoted by $\mathcal{N}_\lambda$, for (s,t)-mincuts as follows.

Construction of $\mathcal{N}_\lambda(G)$: store $N_0(u)$ for each $u \in V$.

This $\mathcal{O}(n^2)$ space structure is used to verify the conditions mentioned in Lemma 52 in $\mathcal{O}(1)$ time.

Observe that till this point we are able to solve the dual edge failure query when both edges are not belonging to any $(\lambda+1)$ (s,t)-class. If they are belonging to $(\lambda+1)$ (s,t)-classes, then it might be the case that both edges are outgoing from a single $(\lambda+1)$ (s,t)-cut. This verification cannot be done using $\mathcal{N}_\lambda$ and hence it is required for us to study the structure of $(\lambda+1)$ (s,t)-cuts.

L.2 Partial solution to dual edge insertion

Let us now consider the insertion of a pair of edges. Let $e = (x, y), e' = (x', y')$ be a pair of edges which are added in G where $\{x, y, x', y'\} \in V$. The (s, t)-mincut value increases upon insertion of a single edge is based on the Fact F.1. Let $\mathcal{S}$ (likewise $\mathcal{T}$) be the $(\lambda + 1)$ (s, t)-class containing source s (likewise sink t). It is a necessary condition from Fact F.1 that to increase the value of (s, t)-mincut at least one edge must be leaving $\mathcal{S}$ and at least one must be entering $\mathcal{T}$, otherwise value of (s, t)-mincut remains unchanged. Without loss of generality assume e leaves $\mathcal{S}$ and e' enters $\mathcal{T}$.

Suppose $y \neq \mathcal{T}$ and $x' \neq \mathcal{S}$, then value of (s, t)-mincut increases by 1 on the following condition.

▶ **Lemma 53.** *The value of (s, t)-mincut in G increases by 1 upon insertion of edges $\{(x, y), (x', y')\}$ where $x \in \mathcal{S}$ and $y' \in \mathcal{T}$ if and only if $x' \in N_0(y)$.*

Proof. Suppose (s, t)-mincut value increases by 1 but $x' \notin N_0(y)$. Let C be the $N_0(y)$. Insertion of edges $\{e, e'\}$ does not increase the capacity of C because $y \in C$ and $x' \notin C$. Hence the (s, t)-cut C remains intact and therefore the value of (s, t)-mincut remains unchanged, contradicts our assumption.

Suppose $x' \in N_0(y)$. Insertion of edge (x, y) increases value of each (s, t)-mincut that keeps y on side of t. So, only those (s, t)-mincuts remain unchanged upon insertion of edge (x, y) which keep y on side of s. Since, (s, t)-mincuts are closed under intersection, therefore, $N_0(y)$ must be the subset of each (s, t)-mincut that keeps y on side of s. Since $x' \in N_0(y)$, therefore upon insertion of edge (x', y'), the value of each (s, t)-mincut, that keeps y on side of s, increases by 1. This implies that the value of (s, t)-mincut in G also increases by 1. ◀

Using data structure $\mathcal{N}_\lambda$ we can verify the condition of Lemma 53 in $\mathcal{O}(1)$ time.

We now discuss when both query edges leaves $\mathcal{S}$ and enters $\mathcal{T}$. Surely in this case the value of (s, t)-mincut increases by 1 from Fact F.1. However since both edges are from $\mathcal{S}$ to $\mathcal{T}$, it is not necessarily true that the (s, t)-mincut value increases by 2 always. If there exists a $(\lambda + 1)$ (s, t)-cut between s and $\{x, x'\}$ or between $\{y, y'\}$ and t then the (s, t)-mincut value increases only by 1. This also motivates us to study all the $(\lambda + 1)$ (s, t)-cuts of G. So, this data structure $\mathcal{N}_\lambda$ can solve the dual edge insertion queries when both edges are not added from S to T. Now we state the following theorem as a partial solution to the dual edge sensitivity problem of (s, t)-mincut.

▶ **Theorem 54.** *For a directed multi-graph G on n vertices and m edges with a designated source vertex s and a designated sink vertex t, there exists an $\mathcal{O}(n^2)$ size data structure $\mathcal{N}_\lambda$ that can,*

1. *answer the resulting (s, t)-mincut value in $\mathcal{O}(1)$ time upon failure of a pair of edges from G when both the edges are not belonging to the same $(\lambda + 1)$ (s, t)-class,*
2. *answer the resulting (s, t)-mincut value upon insertion of a pair of edges in G when both edges are not from $\mathcal{S}$ ($(\lambda + 1)$ (s, t)-class containing source s) to $\mathcal{T}$ ($(\lambda + 1)$ (s, t)-class containing sink t), and,*
3. *report a resulting (s, t)-mincut C in $\mathcal{O}(|C|)$ time.*

M Handling Dual edge insertions

Let $e = (x, y), e' = (x', y')$ be a pair of edges which are added in G where $\{x, y, x', y'\} \in V$. If both edges are not from $\mathcal{S}$ ($(\lambda + 1)$ (s, t)-class containing s) to $\mathcal{T}$ ($(\lambda + 1)$ (s, t)-class containing t), it follows from Theorem 54 that $\mathcal{N}_\lambda$ can report the resulting (s, t)-mincut value.

We shall now consider the case when both edges are from $\mathcal{S}$ to $\mathcal{T}$. The following lemma plays a crucial role in this case.

▶ **Lemma 55.** *The (s,t)-mincut value increases by 2 if and only if both the following conditions holds,*

1. *there does not exist any $(\lambda+1)$ (s,t)-cut that separates $\{s\}$ from $\{x,x'\}$.*
2. *there does not exist any $(\lambda+1)$ (s,t)-cut that separates $\{t\}$ from $\{y,y'\}$.*

Proof. Suppose there exists a $(\lambda+1)$ (s,t)-cut A that separates s from $\{x,x'\}$. Since $\{x,x'\} \in \overline{A}$ therefore addition of the edges e and e' does not increase the value of cut A. Hence after adding both the edges the resulting (s,t)-mincut value is at most $(\lambda+1)$ since $c(A)$ is $(\lambda+1)$.

Directly from premise we can state that any (s,t)-cut that keeps $\{x,x'\}$ on side of t has capacity at least $\lambda+2$. Similarly any (s,t)-cut that keeps $\{y,y'\}$ on side of s also has capacity at least $\lambda+2$. The two query edges e and e' are now contributing to each (s,t)-cut C such that $\{x,x'\} \in C$ and $\{y,y'\} \in \overline{C}$. Since all (s,t)-mincuts have $\{x,x'\}$ on side of s and $\{y,y'\}$ on side of t, therefore both edges are contributing to each (s,t)-mincut. Hence evidently the (s,t)-mincut is increased by 2. ◀

Without loss of generality let us consider the $(\lambda+1)$ (s,t)-class $\mathcal{S}$ that contains source s. It follows from Lemma 55 that we only have to look at $N_1(s)$ among all $(\lambda+1)$ (s,t)-cuts that subdivides $\mathcal{S}$. Using Lemma 33 we state the following corollary.

▶ **Corollary 56.** *There exists an $\mathcal{O}(|\mathcal{S}|)$ size data structure, denoted by $\mathcal{N}_{\lambda+1}(s)$, which stores $N_1(s)$ that subdivides $\mathcal{S}$.*

Based on Corollary 56 and Theorem 54, now we describe the data structure $\mathcal{I}$ for dual edge insertion.

$\mathcal{I}$ consists of the following data structures,
- $\mathcal{N}_\lambda$ for graph G.
- $\mathcal{N}_{\lambda+1}(s)$ for $\mathcal{S}$ in G and $(G)^{\mathcal{T}}$.

The data structure $\mathcal{N}_{\lambda+1}(s)$ can report in $\mathcal{O}(1)$ time if $N_1(s,x)$ exists. Then using a constant number of BELONG(s,x,x') queries, we can verify conditions of Lemma 55.

N Generalizing query Q

Let G be a directed multi-graph on n vertices. Suppose u and v be any pair of vertices in G that belong to a $(\lambda+1)$ (s,t)-class $\mathcal{W}$. To report a $(\lambda+1)$ (s,t)-cut C such that $u \in C$ and $v \in \overline{C}$, we require the data structure $\mathcal{N}_{\lambda+1}$ for $\mathcal{W}$. Now we try to generalize this problem in the following query.

Q': Let $\mathcal{U} = \{u_1, u_2, \ldots, u_k\}$ and $\mathcal{V} = \{v_1, v_2, \ldots v_l\}$ are two disjoint subsets of $\mathcal{W}$. Determine whether there exists a $(\lambda+1)$ (s,t)-cut C such that $\{u_1, u_2, \ldots, u_k\} \in C$ and $\{v_1, v_2, \ldots v_l\} \in \overline{C}$.

We shall show that data structure $\mathcal{F}$ can efficiently answer query Q' and also report such a cut if exists. The Algorithm 1 is executed twice to answer Q' – first for graph $G(\mathcal{W})^I$ and then for graph $(G(\mathcal{W})^U)^{\mathcal{T}}$.

The following lemma proves the correctness of Algorithm 1.

Algorithm 1 Answering query Q'

```
1: procedure QUERY Q'(U, V)
2:     for each u ∈ U do
3:         Let v be any arbitrary vertex from V.
4:         for each w ∈ V do
5:             if BELONG(u, v, w) == 1 then
6:                 Output "(λ + 1) (s, t)-cut does not exist."
7:             end if
8:         end for
9:     end for
10:    C ← W
11:    for each i = 1 to |U| do
12:        Consider a vertex u_i from U and a vertex v from V.
13:        C ← C ∩ N_1(u_i, v)
14:    end for
15:    return C.
16: end procedure
```

▶ **Lemma 57.** *Let $\mathcal{W}$ be a $(\lambda+1)$ (s,t)-class of G. For a pair of subsets $\mathcal{U},\mathcal{V} \subset \mathcal{W}$, There exists a $(\lambda+1)$ (s,t)-cut C in G such that $\mathcal{U} \subseteq C$ and $\mathcal{V} \subseteq \overline{C}$ if and only if $\mathcal{V} \subseteq \overline{\bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v)}$ in $G(\mathcal{W})^I$ or $\mathcal{U} \subseteq \overline{\bigcup_{i\in|\mathcal{V}|,\ u\in\mathcal{U}} N_1(v_i,u)}$ in $(G(\mathcal{W})^U)^{\mathcal{T}}$.*

Proof. If C exists, then it follows from Definition 16 and Lemma 30 that C must be appearing either in $G(\mathcal{W})^I$ or in $G(\mathcal{W})^U$. Without loss of generality assume that C is in $G(\mathcal{W})^I$. It is easy to observe that $N_1(u_i,v_i)$ is a subset of C for each $i \in |\mathcal{U}|$. Therefore, if $\mathcal{V} \subseteq \overline{C}$, it is immediate that $\mathcal{V} \subseteq \overline{\bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v)}$ in $G(\mathcal{W})^I$. The similar arguments work for $(G(\mathcal{W})^U)^{\mathcal{T}}$.

The proof of the converse part is as follows. Without loss of generality assume that $\mathcal{V} \subseteq \overline{\bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v)}$ in graph $G(\mathcal{W})^I$. It follows from Theorem 18 that the only (s,t)-mincut of $G(\mathcal{W})^I$ is complement of $\{t\}$. Therefore, capacity of the intersection of any pair of $(\lambda+1)$ (s,t)-cuts in $G(\mathcal{W})^I$ is greater than λ. Using sub-modularity of cuts(Lemma 9), the capacity of their union has to be $\lambda+1$. Therefore the set $\bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v)$ defines a $(\lambda+1)$ (s,t)-cut. Also, $\mathcal{U} \subseteq \bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v_i)$ and $\mathcal{V} \subseteq \overline{\bigcup_{i\in|\mathcal{U}|,\ v\in\mathcal{V}} N_1(u_i,v)}$ in $G(\mathcal{W})^I$. Since a $(\lambda+1)$ (s,t)-cut exists in $G(\mathcal{W})^I$, it follows from Definition 16 and Lemma 30 that there exists a $(\lambda+1)$ (s,t)-cut C in G such that $\mathcal{U} \subseteq C$ and $\mathcal{V} \subseteq \overline{C}$. The similar arguments work for the following case, $\mathcal{U} \subseteq \overline{\bigcup_{i\in|\mathcal{V}|,\ u\in\mathcal{U}} N_1(v_i,u)}$ in $(G(\mathcal{W})^U)^{\mathcal{T}}$. . ◀

Query time: Algorithm 1 achieves $\mathcal{O}(kl)$ time to answer query Q', and reporting the cut C takes $\mathcal{O}((k+l)|\mathcal{W}| + |\overline{C}|)$ time using $\mathcal{O}(n)$ extra space (refer to Appendix K).

O Proof of Lemma 38

In order to prove Lemma 38 let us first show that graph $\mathcal{D}$ is same as graph $\mathcal{D}_\lambda$ of $\mathcal{D}$. The following lemma states the relation among capacities of 1-transversal cuts in $\mathcal{D}$.

▶ **Lemma 58.** *Each 1-transversal cut in $\mathcal{D}$ has equal capacity.*

Proof. The following fact is immediate from the construction of graph $\mathcal{D}$.

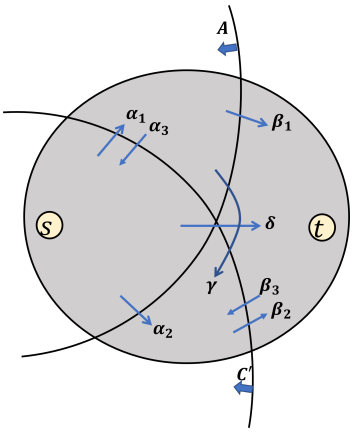

Figure 9 Edges between different regions formed by a 1-transversal cut A and non 1-transversal cut C'

▶ **Fact O.1.** *Each vertex in $\mathcal{D}$ must appear at least in one path from s to t.*

Let C and C' be a pair of 1-transversal cuts in $\mathcal{D}$. We claim that there are no edges between $C \setminus C'$ and $C' \setminus C$. Without loss of generality assume there is an edge (u, v) such that $u \in C \setminus C'$ and $v \in C' \setminus C$. It follows from Fact O.1 that there is a path from s to u. Therefore, the path $\langle s, \ldots, u, v, \ldots, t\rangle$ intersects edge-set of C' at least thrice – it is not possible because C' is a 1-transversal cut.
Let us consider the set $C \setminus C'$. It has incoming edges only from $C \cap C'$ and outgoing edges only to $C \cup C'$. Since each vertex has same number of incoming and outgoing edges, therefore, the number incoming edges of $C \setminus C'$ is same as the number of outgoing edges of $C \setminus C'$. Similarly, the number of incoming and outgoing edges of $C' \setminus C$ is same. Evidently the capacity of C and C' is same. ◀

The following property holds in $\mathcal{D}$.

▶ **Lemma 59.** *A (s,t)-cut is 1-transversal cut in $\mathcal{D}$ if and only if it is a (s,t)-mincut in $\mathcal{D}$.*

Proof. Let C be a 1-transversal cut in $\mathcal{D}$ but assume to the contrary it is not a (s,t)-mincut. Let C' be a (s,t)-mincut in $\mathcal{D}$. It follows from Lemma 58 that C' cannot be 1-transversal, otherwise C becomes a (s,t)-mincut. Since C' is not 1-transversal then there is a path P such that edge-set of C' intersects P at least thrice. Let (u, v) be an edge of P such that $u \in \overline{C'}$ and $v \in C'$. We know $\mathcal{D}$ is a DAG. Let τ be a topological ordering of $\mathcal{D}$. In τ, u must precede v as (u, v) edge exists. We consider prefix of τ till u, say A. This set A defines a 1-transversal cut in $\mathcal{D}$ and $u \in A$, $v \in \overline{A}$. It is easy to observe that A and C' forms four disjoint partition of the vertex set of $\mathcal{D}$. We refer to Figure 9 for the illustration of these regions and edges among them. Note that there cannot be any incoming edge of A, otherwise A does not remain a 1-transversal cut. The following equations follow directly from Figure 9.

$$c(A) = \alpha_2 + \beta_1 + \delta + \gamma = \lambda + k,\ k > 0 \tag{8}$$

$$c(C') = \alpha_1 + \beta_2 + \delta = \lambda \tag{9}$$

We get the following equation from Equation 8 and Equation 9.

$$\alpha_2 + \beta_1 + \gamma = \alpha_1 + \beta_2 + k \tag{10}$$

Since each vertex has equal incoming and outgoing edges therefore, we have the following equations from Figure 9.

$$\beta_2 = \alpha_2 + \beta_3 + \gamma \tag{11}$$
$$\alpha_1 = \alpha_3 + \beta_1 + \gamma \tag{12}$$

We get the following equality by adding Equation 11 and Equation 12, $\beta_2 + \alpha_1 = \alpha_2 + \beta_3 + \alpha_3 + \beta_1 + 2\gamma$. Now replacing $\beta_2 + \alpha_1$ in Equation 10 we get, $\alpha_3 + \beta_3 + \gamma = -k$. Since $k > 0$, $\alpha_3 + \beta_3 + \gamma < 0$, a contradiction.

We shall now prove the converse part. Assume to the contrary that there is a (s,t)-mincut C which is not 1-transversal cut in $\mathcal{D}$. Since C is not 1-transversal, then there is a path P which intersects edge-set of C at least thrice. Let (u,v) be an edge of P such that $u \in \overline{C}$ and $v \in C$. Let τ be a topological ordering of $\mathcal{D}$. In τ, u must precede v since (u,v) is an edge. Prefix of τ till vertex v, say C', defines a 1-transversal cut of $\mathcal{D}$. We showed that each 1-transversal cut is a (s,t)-mincut. Therefore, C' is also a (s,t)-mincut. It is easy to observe that edge (u,v) is from $(C' \setminus C)$ to $(C \setminus C')$ – a contradiction from Lemma 10. ◀

Now we give the proof for Lemma 38.

Proof. Since both edges $\{(s,u),(v,t)\}$ appears in $\mathcal{D}$, their removal reduces (s,t)-mincut value by at least 1. It follows from Lemma 37 that there is a path, say P, from u to v in $\mathcal{D}$. So any (s,t)-cut C in $\mathcal{D}$, in which both edges (s,u) and (v,t) are contributing, cannot be a 1-transversal cut because P intersects $E(C)$ more than once. Therefore, it follows from Lemma 59 that there is no (s,t)-mincut in $\mathcal{D}$ in which both edges are contributing. So, removal of these two edges reduces the (s,t)-mincut value by at most 1.

Assume to the contrary that (s,t)-mincut in $\mathcal{D}$ reduces by 1 but there is no path from u to v in G. Let us consider the set of vertices $R(\{u\})$ that are reachable from u in $\mathcal{D}$. It can be observed that the set $C = \overline{R(\{u\}) \cup T}$ defines a 1-transversal cut in $\mathcal{D}$. It follows from Lemma 59 that C is a (s,t)-mincut in $\mathcal{D}$. It follows from Lemma 37 that there is no path from vertex u to vertex v in $\mathcal{D}$. Evidently, $v \notin R(\{u\})$. Hence observe that both edges (s,u) and (v,t) are contributing to this (s,t)-mincut C. Therefore, value of (s,t)-mincut in $\mathcal{D}$ reduces by exactly 2 – a contradiction. ◀

P Conditional lower bound for dual edge insertion

We show in the following lemma that reachability queries in G can be answered using the dual edge insertion query for (s,t)-mincuts in $\mathcal{D}$.

▶ **Lemma 60.** *Let G be a directed graph. A vertex v is reachable from a vertex u in G if and only if the value of (s,t)-mincut increases by 1 on insertion of edges $\{(s,v),(u,t)\}$ in $\mathcal{D}$ which is obtained from G using Lemma 37.*

Proof. Suppose there is a path from u to v in G. For each (s,t)-cut that keeps u on the side of s, its capacity increases by 1 upon the insertion of edge (u,t). Therefore, after the insertion of edge (u,t) each (s,t)-mincut must keep u on side of t. Since (s,t)-mincuts appears as a 1-transversal cut of $\mathcal{D}$ (Lemma 59), it implies that each (s,t)-mincut of $\mathcal{D}$ keeps the set of vertices reachable from u in $\mathcal{D}$ on side of t. The following fact is immediate from Fact F.1.

▶ **Fact P.1.** *Value of (s,t)-mincut in a directed graph increases by 1 upon insertion of an edge if and only if the edge contributes to each (s,t)-mincut in the graph.*

We have $v \in R(\{u\})$ from the premise and Lemma 37. Therefore, after insertion of edge (u,t), insertion of edge (s,v) increases the value of (s,t)-mincut in $\mathcal{D}$ by 1 using Fact P.1.

Suppose v is not reachable from u in G. Let us consider the set $R(\{u\}) \cup T$ in $\mathcal{D}$. The set $C = \overline{R(\{u\}) \cup T}$ defines a 1-transversal cut. Hence C is a (s,t)-mincut of $\mathcal{D}$ as stated in Lemma 59. Since $v \notin R(\{u\})$, therefore insertion of edges $\{(s,v),(u,t)\}$ does not increase the capacity of (s,t)-cut C. As a result value of (s,t)-mincut in $\mathcal{D}$ remains unchanged by the insertion of edges $\{(s,v),(u,t)\}$. ◀

Lemma 60 completes the proof of Theorem 39.

Q Generalized Flow Tree for 2×2 mincuts

We shall show in the following lemma how to transform a reachability query to the following query in G – Given any pair of vertices x, y in G, does there exists a (s,t)-mincut C in G such that a vertex $x \in C$ and a vertex $y \in \overline{C}$?

▶ **Lemma 61.** *Suppose u and v are two vertices of a directed graph G. A vertex v is reachable from u in G if and only if each (s,t)-cut C in $\mathcal{D}$ (obtained from G using Lemma 37) with $v \in C$ and $u \in \overline{C}$ has capacity greater than the capacity of (s,t)-mincut.*

Proof. Assume to the contrary that v has a path from u in G but there is a (s,t)-mincut C in $\mathcal{D}$ such that $v \in C$ and $u \in \overline{C}$. It follows from Lemma 37 that path P exists in $\mathcal{D}$. Therefore, the edge-set of the (s,t)-mincut C intersects the path $\langle s, \ldots, u, v\rangle$ (from Fact O.1) more than once in $\mathcal{D}$. So, C is not a 1-transversal cut in $\mathcal{D}$ – contradiction from Lemma 59.

Again we assume to the contrary that suppose each (s,t)-cut C in $\mathcal{D}$ with $v \in C$ and $u \in \overline{C}$ has capacity greater than the capacity of (s,t)-mincut but vertex v is not reachable from u in G. Let us look at the set of vertices $R(\{u\})$ that are reachable from u in $\mathcal{D}$. The set $C' = \overline{R(\{u\}) \cup t}$ defines a 1-transversal cut in $\mathcal{D}$ and hence it follows from Lemma 59 that C' is a (s,t)-mincut in $\mathcal{D}$. Since v is not reachable from u, therefore $v \notin R(\{u\})$. Therefore, for (s,t)-mincut C', $u \in \overline{C'}$ and $v \in C'$ – a contradiction. ◀

Using Conjecture 4 and Lemma 61 we state the following conditional lower bound.

▶ **Theorem 62.** *Assuming Directed Reachability Hypothesis holds, any data structure that can determine whether there is a (s,t)-mincut in G that keeps u on side of s and v on side of t, for a given pair of vertices $\{u,v\}$ and a designated source vertex s and a designated sink vertex t, in a directed multi-graph on n vertices must either use $\tilde{\Omega}(n^2)$ space, or linear query time.*

R Conditional lower bound on determining the existence of a $(\lambda+1)$ (s,t)-cut

Let G be a directed multi-graph on n vertices and m edges with a designated source vertex s and a designated sink vertex t. We first construct the DAG structure $\mathcal{D}_\lambda$ from G that stores all (s,t)-mincuts of G.

Construction of $\mathcal{D}_1$: The graph $\mathcal{D}_1$ is obtained by modifying $\mathcal{D}_\lambda$ as follows. We add one dummy source s' and connect using $\lambda - 1$ directed edges from s' to S. Similarly we add a dummy sink t' and connect using $\lambda - 1$ directed edges from T to t'. Observe that $\{s'\}$ and complement of $\{t'\}$ are only two (s',t')-mincut of $\mathcal{D}_1$.

▶ **Lemma 63.** *A (s,t)-cut C in $\mathcal{D}_\lambda$ is a (s,t)-mincut in G if and only if $C \cup \{s'\}$ is a minimum+1 (s',t')-cut in $\mathcal{D}_1$.*

Proof. Let us consider a (s,t)-mincut C of D_λ. Evidently from construction of D_1, $\{S, s'\} \in C \cup s'$ and $\{T, t'\} \in \overline{C \cup \{s'\}}$. Therefore the addition of edges from s' to S does not increase the capacity of $C \cup \{s'\}$. Similarly addition of edges from T to t' also does not increase the capacity of $C \cup \{s'\}$. Since the (s',t')-mincut is $\lambda - 1$, therefore $C \cup \{s'\}$ is a minimum+1 (s',t')-cut.

Let C be a minimum+1 (s',t')-cut of D_1, $c(C) = \lambda$. From the construction it is obvious that both $\{S, s'\} \in C$ and $\{T, t'\} \in \overline{C}$. Therefore $C \setminus \{s'\}$ defines a valid (s,t)-cut of D_λ. Moreover, it follows from the construction of $\mathcal{D}_1$ that the capacity of $C \setminus \{s'\}$ remains unchanged. Hence $C \setminus \{s'\}$ is a valid (s,t)-mincut of D_λ. ◀

The following corollary is immediate from Lemma 63.

▶ **Corollary 64.** *For any given pair of vertices $\{u, v\}$ in G, there is a (s,t)-mincut in G that keeps u on side of s and v on side of t if and only if there is a minimum+1 (s',t')-cut in $\mathcal{D}_1$ that keeps u on side of s' and v on side of t'.*

Using Corollary 64 and Theorem 62, we state the following conditional lower bound.

▶ **Theorem 65.** *For any directed multi-graph on n vertices with a designated source vertex s and a designated sink vertex t, any data structure that can report , for a given pair of vertices $\{u, v\}$, if there exists a $(\lambda+1)$ (s,t)-cut that keeps u on side of s and v on side of t occupies $\tilde{\Omega}(n^2)$ space, or takes linear query time unless reachability hypothesis is violated.*