Minimum Topological $s-t$ Cut Problems

Javad Tayyebi[1], Adrian Marius Deaconu[2] and Malihe Niksirat[3]

***Abstract*— This paper introduces a novel combinatorial structure known as the topological $s-t$ cut, a specialized type of $s-t$ cut applicable to acyclic networks. Unlike traditional $s-t$ cuts, topological $s-t$ cuts intersect with exactly one arc of any $s-t$ path d. A linear programming formulation is presented to tackle this problem and it is shown how to solve the problem using the well-known maximum flow algorithms on auxiliary networks. Additionally, a linear-time method is developed for finding minimum topological $s-t$ cuts in $s-t$ planar acyclic networks.**

I. Introduction

The concept of $s-t$ cuts is closely linked to the maximum flow problem through the Max-Flow Min-Cut Theorem, which states that the maximum flow in a network from s to t is equal to the capacity of the minimum $s-t$ cut [7], [9]. This fundamental relationship, along with the well-known minimum $s-t$ cut problem, facilitates efficient solutions to various optimization problems, including those in transportation networks, communication systems, and image segmentation [1], [5].

Several extensions of the minimum $s-t$ cut problem have been explored in the literature. In [19], the authors introduced the concept of the minimum logical $s-t$ cut problem. This variant imposes the additional constraint that all outgoing arcs from any node in the graph G cannot be removed simultaneously. In [23], the minimum label $s-t$ cut problem is examined, wherein the goal is to identify a minimum-size subset of labels such that removing all edges associated with these labels from G results in the disconnection of s and t. The paper presents two linear programming formulations and discusses various approximability results associated with this problem. Additionally, in [3], the authors investigate the constrained minimum $s-t$ cut problem, providing a necessary and sufficient condition for the existence of an integral optimal minimal solution. It is also demonstrated that determining whether this condition is satisfied is NP-hard.

By adapting $s-t$ cuts to include the topological order of the network, this paper introduces the novel concept of *topological $s-t$ cuts* on acyclic networks. These cuts intersect each $s-t$ path at exactly one arc, thereby preserving the unique features of the network. This enhancement facilitates better management of flow and resource allocation within complex network systems, thereby increasing the overall efficiency and performance of combinatorial optimization in real-world applications. The distinct property of topological $s-t$ cuts allows us to model some real-world situations, such as measuring the flow value. Hence, topological $s-t$ cuts can be utilized for counting purposes. By placing counters on the arcs that are part of a topological $s-t$ cuts, each counter monitors only one arc of each $s-t$ path. This setup facilitates flow measurement without confusion from reverse flow. Such a structure simplifies flow analysis, as each counter accounts for a single, unique flow, resulting in more accurate and efficient data collection. This counting system is particularly beneficial in contexts such as water distribution networks, where tracking the volume of water directed to different areas is critical for effective resource management, ultimately leading to improved operations and decision-making.

Apart from introducing the concept of topological $s-t$ cuts, the contribution of this paper can be divided into three parts:

1) a linear programming formulation of the minimum topological $s-t$ cut problem allowing for solutions of the problem in polynomial time through the use of interior-point methods [6], [21],
2) transforming the network into an auxiliary network in such a way that the minimum topological cut can be found by obtaining maximum flow,
3) defining the directed dual network and extending the approach of using the dual network to find the minimum cut.

II. Preliminaries

We consider a directed graph $G(V,A)$, where $V=\{1,2,\ldots,n\}$ represents the set of nodes and A denotes the set of arcs, which contains m arcs. Within this graph, two specific nodes are designated: a source node s and a destination node t.

A path is defined as a sequence of nodes $v_1-v_2-\ldots-v_q$ such that the arc (v_i,v_{i+1}) belongs to A for $i=1,2,\ldots,q-1$. A cycle is a type of path in which the first and last nodes are the same. A path is considered simple if it does not revisit any nodes. To simplify terminology, a simple path that originates from s and terminates at t is referred to as an $s-t$ path. If the arcs are associated with capacities c_{ij}, the capacity of a path is defined as the minimum capacity

*This work was not supported by any organization

[1]Javad Tayyebi is with the Department of Industrial Engineering, Faculty of Industrial and Computer Engineering, Birjand University of Technology, Birjand, Iran javadtayyebi@birjandut.ac.ir

[2]Adrian Marius Deaconu with the Department of Mathematics and Computer Science, Transilvania University of Brasov, 500091, Brasov, Romania a.deaconu@unitbv.ro

[3]Malihe Niksirat with the Department of Computer Science, Faculty of Industrial and Computer Engineering, Birjand University of Technology, Birjand, Iran niksirat@gmail.com

among its arcs, given by:

$$\mathbf{c}(P) = \min_{(i,j)\in P} c_{ij}.$$

Throughout this paper, we make the following assumption:

Assumption 1. Every node is placed along at least one $s-t$ path.

This assumption is not restrictive, as nodes that do not meet this assumption are deemed ineffective and unnecessary, allowing us to exclude them from the network.

For any subset S of V with $s \in S$ and $t \in V\backslash S$, a set C of arcs that have one endpoint in S and the other in $V\backslash S$ is referred to as an $s-t$ cut, denoted by $C = [S, \bar{S}] = [S, V\backslash S]$. Those arcs of $[S, \bar{S}]$ emanating from S and terminating at $\bar{S}$ are called forward arcs, and its other arcs are backward arcs. We denote by $(S, \bar{S})$ the set of forward arcs of $[S, \bar{S}]$ and by $(\bar{S}, S)$ the set of backward arcs of $[S, \bar{S}]$. The capacity of an $s-t$ cut $[S, \bar{S}]$ with respect to a capacity vector $\mathbf{c}$, denoted by $\mathbf{c}[S, \bar{S}]$, is the total capacity of its forward arcs, i.e.,

$$\mathbf{c}[S, \bar{S}] = \sum_{(i,j)\in(S,\bar{S})} c_{ij}.$$

A famous network optimization problem is to find an $s-t$ cut with minimum capacity. This problem is called the minimum $s-t$ cut problem, and it can be formulated as follows [2]:

$$\min \quad \sum_{(i,j)\in A} c_{ij}\theta_{ij} \tag{1a}$$

$$\text{s.t.} \quad \pi_i - \pi_j + \theta_{ij} \geq 0 \qquad \forall (i,j) \in A, \tag{1b}$$

$$\pi_t - \pi_s = 1, \tag{1c}$$

$$\theta_{ij} \geq 0 \qquad (i,j) \in A, \tag{1d}$$

$$\pi_i \geq 0 \qquad i \in V. \tag{1e}$$

Although the formulation in (1) represents a linear programming problem, it can be demonstrated that all variables take binary values of either zero or one. This conclusion arises from the property of total unimodularity in the coefficient matrix [2]. In this formulation, an arc (i,j) is considered a forward edge of the optimal st-cut $[S, \bar{S}]$ if and only if $\theta_{ij} = 1$. Furthermore, node i is included in set S if $\pi_i = 0$; otherwise, it belongs to $\bar{S}$.

The best-known algorithm of minimum $s-t$ cut problems, associated with Hao and Orlin, runs in $O(nm\log(\frac{n^2}{m}))$ time [11].

III. Notion of Topological $s-t$ Cuts

We recall that G is a directed acyclic graph (DAG) with finite capacities. For the DAG G, a topological ordering is defined as a linear ordering of its nodes such that for every arc (i,j), node i precedes node j in this ordering. This property ensures that all dependency relationships between the nodes are respected, allowing for tasks to be scheduled or processed in an appropriate sequence.

Definition III.1. In a DAG, an $s-t$ cut $C = [S, \bar{S}]$ is referred to as a topological $s-t$ cut if there exists a topological ordering of the nodes such that all nodes in S have labels less than those in $\bar{S}$.

By definition, any topological $s-t$ cut C satisfies the inequality $i < j$ for every arc $(i,j) \in C$. So, it maintains the topological ordering and does not include any backward arcs. On the other word, it does not contain any arcs that would violate the notion of the topological order by pointing from a node to another that should come before it. Specially, we have the following property.

Lemma III.2. *An $s-t$ cut C is topological if and only if it contains exactly one common arc with every $s-t$ path.*

Proof: 1.(If Direction): By the notion of $s-t$ cut, it follows that C must share at least one arc with every $s-t$ path. Next, we argue that if a topological $s-t$ cut were to contain more than one arc with a given $s-t$ path, it would necessarily imply the presence of a backward arc. For a topological $s-t$ cut, all arcs must respect the topological ordering; therefore, if two arcs (u,v) and (v,w) from the $s-t$ path were both included in the cut C, it would mean that u appears before v and v appears before w. Since both arcs belong to the same topological $s-t$ cut, we can conclude that v cannot be part of the path from s to t without violating the topological order.

2. (Only If Direction): Conversely, suppose C contains a backward arc, and let P be an $s-t$ path that includes this arc. Clearly, P must contain at least two forward common arcs from C, which leads to a contradiction.

Using a topological $s-t$ cut is advantageous due to its ability to maintain the structure and directionality of dependencies in a DAG, simplify analysis and optimization problems, and improve clarity in communication. This makes topological $s-t$ cuts particularly valuable in fields such as operations research, scheduling, and computer science, where directed dependencies play a crucial role. Let us present an example to illustrate that any $s-t$ cut is not necessarily topological in a DAG.

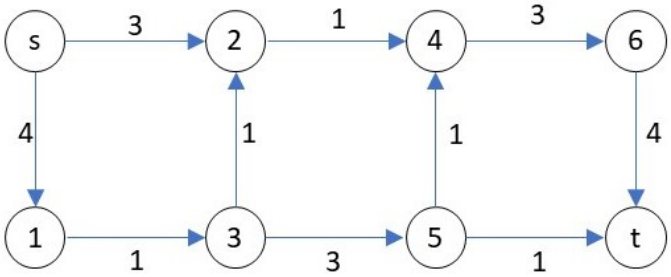

Fig. 1. An example of the topological $s-t$ cut

Example 1. Figure 1 illustrates a DAG with specified arc capacities. In this graph, the minimum $s-t$ cut has a capacity of 2, determined by the set $S = \{s, 1, 2\}$. This cut consists of two forward arcs, $(1,3)$ and $(2,4)$, as well as a backward arc $(3,2)$. Consequently, it does not qualify as a topological $s-t$ cut due to the presence of the backward arc. Notably, the $s-t$ cut defined by $S = \{s, 1, 2, 3, 5\}$ qualifies as a topological $s-t$ cut, featuring a minimum capacity of 3 among all such topological $s-t$ cuts.

Based on this example, existing algorithms designed for solving the minimum $s-t$-cut problem cannot be directly utilized to determine a minimum topological $s-t$ cut. Therefore, we must explore alternative methods for identifying a minimum topological $s-t$-cut. To address this issue, the next section will first present a linear programming formulation of the problem, demonstrating that it can be resolved in polynomial time. Additionally, an auxiliary network is introduced to identify a minimum topological $s-t$ cut by solving an ordinary maximum flow problem. Furthermore, a polynomial-time algorithm is developed specifically aimed at obtaining a minimum topological $s-t$ cut in an $s-t$ planar DAG.

IV. Formulation and Approach

The proposed formulation of the minimum topological $s-t$ cut is similar to the one in (1) for the ordinary minimum $s-t$ cut problem, with a slight modification. It is presented as follows:

$$\min \quad \sum_{(i,j)\in A} c_{ij}\theta_{ij} \tag{2a}$$
$$\text{s.t.} \quad \pi_i - \pi_j + \theta_{ij} = 0 \qquad \forall (i,j)\in A, \tag{2b}$$
$$\pi_t - \pi_s = 1, \tag{2c}$$
$$\theta_{ij} \geq 0 \qquad (i,j)\in A, \tag{2d}$$
$$\pi_i \geq 0 \qquad i \in V. \tag{2e}$$

The only difference is that the inequality (1b) appears in equality form, here. Since the coefficient matrix remains unchanged, it follows that this formulation also exhibits the property of total unimodularity, ensuring that all variables take binary values of zero or one [2]. The following theorem demonstrates the validity of this formulation.

Theorem IV.1. *The linear programming formulation* (2) *identifies a minimum topological $s-t$ cut.*

Proof: The interpretation of all variables is the same as in (1). It suffices to show that constraint (2b) prevents the existence of a backward arc in the $s-t$ cut. By contradiction, assume there is a backward arc (l,k). This implies that $\pi_l = 1$ and $\pi_k = 0$. Consequently, this constraint can be expressed as $1+\theta_{lk}=0$. Given that $\theta_{lk}\geq 0$, this constraint cannot be satisfied, leading to a contradiction.

Example 2. We ran the two formulations (1) and (2) for the instance provided in Example 1. Table I presents the optimal solutions. It is clear that the solution of (1) corresponds to the minimum $s-t$ cut, while the solution of (2) yields the minimum topological $s-t$ cut, correctly.

Since finding a minimum topological $s-t$ cut admits a linear programming formulation, it follows that the problem can be solved in polynomial time by the well-known interior point algorithms [6], [21]. However, we focus on developing an efficient algorithm specifically based on the dual of this formulation.

Formulation	Arcs with $\theta_{ij}=1$	Nodes with $\pi_i = 1$	Optimal value
(1)	$(1,3),(2,4)$	$3,4,5,6,t$	2
(2)	$(2,4),(5,4),(5,t)$	$4,6,t$	3

TABLE I
THE RESULTS CALCULATED FROM EXAMPLE 2

$$\max \quad z = v \tag{3a}$$
$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \tag{3b}$$
$$\begin{cases} v & i = s, \\ 0 & i \neq s,t, \\ -v & i = t, \end{cases} \quad \forall i \in V, \tag{3c}$$
$$x_{ij} \leq c_{ij} \quad \forall (i,j)\in A, \tag{3d}$$

in which x_{ij} represent the decision variable on the arc (i,j). This problem is recognized as the well-known maximum flow problem, with the distinction that the decision variable x_{ij} is lower unbounded, meaning it can take on any negative value. Since negative flow lacks meaningful interpretation, we define two nonnegative decision variables, denoted by x'_{ij} and x'_{ji}, in such a manner that:

$$x_{ij} = x'_{ij} - x'_{ji}.$$

It is straightforward to observe that the original problem can be transformed into the following maximum flow problem.

$$\max \quad z = v \tag{4a}$$
$$\text{s.t.} \quad \sum_{j:(i,j)\in A} (x'_{ij} - x'_{ji}) - \sum_{j:(j,i)\in A} (x'_{ji} - x'_{ij}) \tag{4b}$$
$$= \begin{cases} v & i = s, \\ 0 & i \neq s,t, \\ -v & i = t, \end{cases} \quad \forall i \in V, \tag{4c}$$
$$0 \leq x'_{ij} \leq c_{ij} \quad \forall (i,j)\in A, \tag{4d}$$
$$0 \leq x'_{ji} (< +\infty) \quad \forall (i,j)\in A. \tag{4e}$$

This problem is defined on the auxiliary network $\bar{G} = (V, \bar{A}, \bar{\mathbf{c}})$, where each arc (i,j) from the original network is retained with its corresponding capacity. Additionally, for each arc (i,j) in the original network, there exists an arc (j,i) in the auxiliary network with infinite capacity, denoted as $\bar{c}_{ji} = +\infty$.

The objective is to determine the maximum flow in this auxiliary network $\bar{G}$. Upon finding the maximum flow, we can then employ a traversal algorithm, such as the breadth-first search (BFS) method or its variants, to identify the minimum $s-t$ cut. This approach is a formalized methodology in network flow theory [2], and it is described in Algorithm 1.

Theorem IV.2. *Algorithm 1 finds a minimum topological $s-t$ cut in polynomial time.*

Proof: If one uses polynomial-time algorithms for solving the maximum flow problem, it is straightforward to show that the complexity of Algorithm 1 is also polynomial time. To demonstrate that the minimum $s-t$ cut is topological, note that if it contains a backward arc (j,i) of the original network, then the capacity of (i,j) is infinite. Consequently, the maximum flow value becomes $+\infty$. This indicates there exists a $s-t$ path with infinite capacities in $\bar{G}$. Therefore, there must be a path from t to s in the original network, which contradicts the fact that the network is acyclic and Assumption 1.

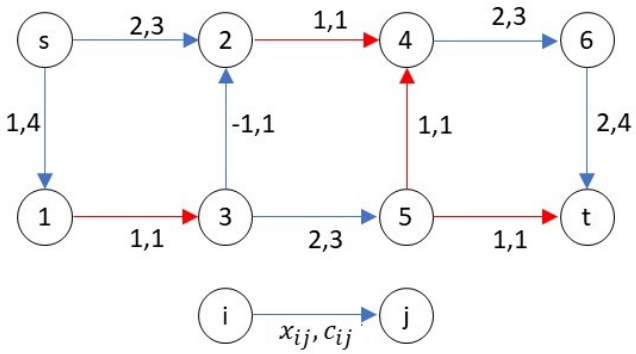

Fig. 2. The optimal solution of problem (3) for Example 1

Algorithm 1 To find a minimum topological $s-t$ cut

Input: An acyclic network $G=(V,A,\mathbf{c})$.
Output: A minimum topological $s-t$ cut C.
Initialize $\bar{A}=\emptyset$ and $C=\emptyset$.
for each arc $(i,j)\in A$ **do** ▷ To construct the auxiliary network $\bar{G}=(V,\bar{A},\bar{\mathbf{c}})$
    Add (i,j) to $\bar{A}$ with capacity $\bar{c}_{ij}=c_{ij}$.
    Add (j,i) to $\bar{A}$ with capacity $\bar{c}_{ij}=+\infty$.
end for
Find maximum flow x' in the auxiliary network $\bar{G}$.
Perform a BFS in the residual network given by $\bar{G}$ and x' to identify reachable nodes from s.
Let S be the set of nodes reachable from s.
for each arc $(i,j)\in\bar{A}$ **do**
    if $i\in S$ and $j\notin S$ **then**
        Add (i,j) to C.
    end if
end for

Example 3. To illustrate the performance of Algorithm 1, we will run it on the instance given in Example 1. Suppose we have constructed the corresponding auxiliary network $\bar{G}$. Figure 2 depicts the optimal solution to problem 3 for this example. The red arcs are saturated. It is important to note that the maximum flow is achieved by sending three units of flow along the $s-t$ paths: $s-1-3-5-t$, $s-2-4-6-t$, and $s-2-3-5-4-6-t$. Additionally, observe that the last $s-t$ path employs the reverse arc of $(3,2)$, which has infinite capacity. For this reason, it has a negative value of flow. The arcs that are reachable in the residual network are $S=\{s,1,2,3,5\}$, which yields the same minimum topological $s-t$ cut $C=\{(2,4),(5,4),(5,t)\}$.

V. $s-t$ Planar Networks

A significant result in undirected $s-t$ planar networks is that finding the shortest path in the dual network corresponds to a minimum $s-t$ cut in the original network. In spite of the many works performed on planar networks (see Table II for a survey), this result has not been extended to directed planar networks. Here, we orientate every edge of the dual network as it is performed in [14]. Then, we show that the shortest path in this network is actually a minimum topological $s-t$ cut and not necessarily an (ordinary) minimum $s-t$ cut.

As stated, our proposed algorithm is a variation of the method utilized to find a minimum $s-t$ cut in undirected $s-t$ planar graphs (see Section 8.4 of [2] for more study). For this purpose, it is needed to introduce a dual network $G^*(F^*,A^*)$.

We begin by adding a new arc between the nodes s and t, ensuring that it remains within outer face of the graph. Assuming that node s is located to the west and node t is located to the east, this arc will be placed to the north. This addition creates a new region within the network, which we refer to as the additional face, while maintaining the planarity of the original network.

To construct the dual network G^*, we place a node within each face of the graph G. Since each arc corresponds to the boundary between two faces, we create an (undirected) edge by connecting the vertices corresponding to those faces in the dual network. We define the node associated with the additional face as the dual origin s^*, and the node related to the outer face as the dual destination t^*. In this dual representation, we observe the presence of the arc (s^*,t^*); however, we remove this arc from the network by design. We assign a direction to each edge in the dual network. The direction of any arc from G^* can be determined by applying a 90-degree counterclockwise rotation to the corresponding directed arc from G (which intersects). Consequently, the horizontal axis represents the directed arc in G, while the vertical axis represents the directed arc in G^*. If the costs assigned to the arcs in the dual network G^* are set equal to the costs of their corresponding arcs in the original network, then the following result is immediate.

Theorem V.1. *The shortest s^*-t^* path in G^* corresponds to the minimum topological $s-t$ cut in G.*

Proof: At first, it is essential to observe that any topological $s-t$ cut of G corresponds to a (directed) s^*-t^* path in G^* and vice versa. The proof that any s^*-t^* path P^* in G^* serves as an $s-t$ cut in G is well-documented in the literature. The designation of it being topological arises from the fact that the arcs of P^* are directed from s^* to t^*, which ensures that any arc of the $s-t$ cut is directed from

TABLE II

HISTORY OF MAXIMUM FLOW AND MINIMUM CUT PROBLEMS FOR PLANAR NETWORKS

Year	Subject	Description	Reference
1979	Maximum flow in planar networks	Introduced algorithms for computing maximum flow in planar networks efficiently using planarity.	[12]
1983	Minimum s-t cut algorithm	Presented an algorithm to compute the minimum s-t cut in planar undirected networks in $O(nlog^2(n))$ time.	[18]
1985	Maximum flow algorithm for undirected planar networks	Developed an $O(nlog^2(n))$ algorithm for maximum flow in undirected planar networks.	[10]
1987	Parallel algorithms	Discussed parallel algorithms for computing minimum cuts and maximum flows in planar networks.	[13]
1989	Multicommodity flow in planar networks	Studied max-flow min-cut theorem for multicommodity flows in certain planar directed networks.	[15]
1995	Flow in planar graphs with multiple sources and sinks	Investigated maximum flow problems with multiple sources and sinks in planar graphs.	[16]
1997	Maximum (s,t)-flows in planar networks	Introduced an efficient algorithm for maximum (s,t)-flows in planar networks in $O(n \log n)$ time.	[20]
2011	Improved algorithms for min cut and max flow	Presented improved algorithms for minimum cut and maximum flow in undirected planar graphs.	[22]
2011	Minimum s-t cut when source and sink are close	Discussed an efficient solution for computing the minimum s-t cut in undirected planar graphs with close source and sink.	[17]
2016	Distributed algorithms for planar networks	Covered low-congestion shortcuts and minimum cut algorithms in planar networks in a distributed setting.	[8]
2017	Multiple-source multiple-sink maximum flow	Proposed near-linear time algorithms for maximum flow in directed planar graphs with multiple sources and sinks.	[4]

s to t by construction. Given that the length of the $s^* - t^*$ path in G^* equals the capacity of a topological $s-t$ cut in G, the result follows directly.

According to this theorem, we can utilize well-known shortest path algorithms to find the minimum topological st-cut in polynomial-time (super-linear) time. However, we can derive an additional result indicating that G^* can be simplified into a DAG, imposing Assumption 1, thus allowing us to find the shortest path in linear time.

To illustrate this, observe that any directed cycle C^* in G^* surrounds a subgraph $\tilde{G}$ of G, which is connected to the remaining component $G \setminus \tilde{G}$ via arcs which are directed from $\tilde{G}$ to $G \setminus \tilde{G}$ (or vice versa). Consequently, this component can be removed from G since it does not participate in any $s-t$ path.

To eliminate the cycles in G^*, we must also remove their corresponding subgraphs in G. This is achieved by labeling all nodes that are accessible from s and all nodes from which there exists at least one path to t. We then remove nodes that are not labeled in both cases. This guarantees that every node in G lies along some st- path, preventing the existence of any subgraph where all arcs are either outgoing or incoming. Consequently, G^* does not contain any cycles.

Algorithm 2 describes the approach, formally.

Algorithm 2 Find Minimum Topological $s-t$ Cut

1: **Input:** A planar DAG $G(V, A)$ with capacity vector $\mathbf{c}$.
2: **Output:** A minimum topological $s-t$ cut C.
3: Use the Breadth-First-Search algorithm to label all nodes which are accessible from s and all nodes which have access to t.
4: Eliminate all nodes not labeled twice. ▷ To satisfy Assumption 1
5: Add an artificial arc (s, t) to G with capacity $+\infty$.
6: Construct the directed dual network $G^*(F^*, A^*)$.
7: Find a shortest path P^* from s^* to t^*.
8: Find the set C of arcs in the original network corresponding to those of P^*.

Example 4. Let us illustrate Algorithm 2 for the network presented in Example 1. Since all nodes are labeled twice, any node is not eliminated from the network. Figure 3 provides us with the dual of this network. Figure 3.a presents the original network with blue arcs, in which an artificial arc (s, t) is added, and the dual network is depicted by orange nodes and orange arcs. The shortest path from s^* to t^* is $s^* - 3^* - 2^* - t^*$ with a length equal to 3. Its corresponding topological $s-t$ cut is $C = \{(2, 4), (5, 4), (5, t)\}$, which is identical to the solution presented in Example 1. Now, it is notable that the ordinary minimum $s-t$ cut, which has the forward arcs $(1, 3), (2, 4)$, and the backward arc $(3, 2)$, does not correspond to an $s^* - t^*$ path in G^*.

Theorem V.2. *On a planar network, if Assumption 1 is satisfied, then Algorithm 2 finds a minimum topological $s-t$ cut in linear time. Otherwise, it operates in $O(m + n \log n)$ time.*

Proof: The proof relies on the fact that the most efficient algorithm for solving shortest path problems runs in linear time for DAGs, while it has a time complexity of $O(m + n \log n)$ for general graphs [2].

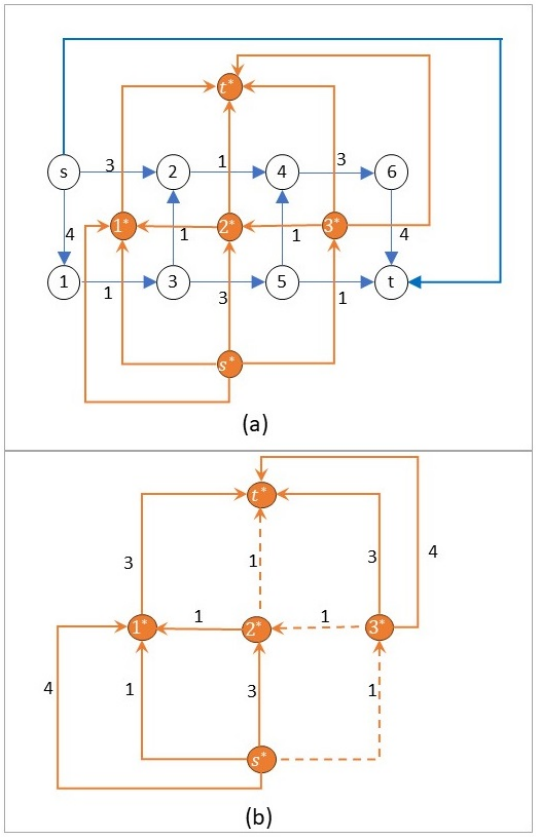

Fig. 3. (a) The original network depicted with blue arcs, alongside the dual network represented with orange arcs. (b) The dual network shown with orange arcs, highlighting the $s^* - t^*$ shortest path marked by hashed orange arcs

VI. Concluding Remarks

This paper introduced a novel combinatorial structure called topological $s-t$ cut, which is specifically applicable to acyclic networks. Unlike traditional $s-t$ cuts, topological $s-t$ cuts were shown to intersect with exactly one arc of any $s-t$ path, providing a more nuanced approach to network analysis. We presented a linear programming formulation to address the minimum topological $s-t$ cut problem and demonstrated how to solve it by utilizing well-established techniques involving maximum flow problems on auxiliary networks.

Additionally, we developed a linear-time algorithm for finding minimum topological $s-t$ cuts in planar acyclic networks, which significantly enhances existing methodologies for network optimization. This work not only advanced the understanding of topological cuts within the context of acyclic networks but also opens doors for further exploration in the field of combinatorial optimization.

In terms of future work, several avenues remain promising. First, the scalability of the proposed algorithms in more complex network structures, such as cyclic networks, warrants investigation. Furthermore, exploring the potential applications of topological $s-t$ cuts in real-world scenarios, such as transportation systems or communication networks, could reveal their practical significance. Finally, extending this research to investigate other combinatorial structures that can benefit from similar topological considerations may yield fruitful results. Overall, we believe that this foundational work will inspire further research and innovation in network design and optimization.

References

[1] Abdolahzadeh, A., Aman, M., & Tayyebi, J. (2020). Minimum st-cut interdiction problem. Computers & Industrial Engineering, 148, 106708.

[2] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). Network flows: Theory, applications and algorithms. Prentice-Hall, Englewood Cliffs, New Jersey, USA.

[3] Aissi, H., & Mahjoub, A. R. (2024). On the minimum s-t cut problem with budget constraints. Mathematical Programming, 203(1), 421-442.

[4] Borradaile, G., Klein, P. N., Mozes, S., Nussbaum, Y., & Wulff-Nilsen, C. (2017). "Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time." SIAM Journal on Computing, 46(4), 1280-1303.

[5] Chen, X., & Pan, L. (2018). A survey of graph cuts/graph search based medical image segmentation. IEEE Reviews in Biomedical Engineering, 11, 112-124.

[6] Darvay, Z. (2003). New interior point algorithms in linear programming. Advances in Modeling and Optimization, 5(1), 51-92.

[7] Dantzig, G., & Fulkerson, D. R. (2003). On the max flow min cut theorem of networks. Linear Inequalities and Related Systems, 38, 225-231.

[8] Ghaffari, M., & Haeupler, B. (2016). "Distributed algorithms for planar networks II: Low-congestion shortcuts, MST, and min-cut." In Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 202-219.

[9] Ghrist, R., & Krishnan, S. (2013, December). A topological max-flow-min-cut theorem. In 2013 IEEE Global Conference on Signal and Information Processing (pp. 815-818). IEEE.

[10] Hassin, R., & Johnson, D. B. (1985). "An $O(nlog^2n)$ algorithm for maximum flow in undirected planar networks." SIAM Journal on Computing, 14(3), 612-624.

[11] Hao, J. X., & Orlin, J. B. (1994). A faster algorithm for finding the minimum cut in a directed graph. Journal of Algorithms, 17(3), 424-446.

[12] Itai, A., & Shiloach, Y. (1979). "Maximum flow in planar networks." SIAM Journal on Computing, 8(2), 135-150.

[13] Johnson, D. B. (1987). "Parallel algorithms for minimum cuts and maximum flows in planar networks." Journal of the ACM (JACM), 34(4), 950-967.

[14] Khuller, S., Naor, J., & Klein, P., The Lattice Structure of Flow in Planar Graphs, SIAM Journal on Discrete Mathematics 6, 3, 477-490 (1993).

[15] Nagamochi, H., & Ibaraki, T. (1989). "Max-flow min-cut theorem for the multicommodity flows in certain planar directed networks." Electronics and Communications in Japan, 72(3), 58-72.

[16] Miller, G. L., & Naor, J. (1995). "Flow in planar graphs with multiple sources and sinks." SIAM Journal on Computing, 24(5), 1002-1017.

[17] Kaplan, H., & Nussbaum, Y. (2011). "Minimum s-t cut in undirected planar graphs when the source and the sink are close." In 28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011).

[18] Reif, J. H. (1983). "Minimum s-t cut of a planar undirected network in $O(nlog^2(n))$ time." SIAM Journal on Computing, 12(1), 71-81.

[19] Valizadeh, M., & Tadayon, M. H. (2022). Logical s-t min-cut problem: An extension to the classic s-t min-cut problem. Iranian Journal of Mathematical Sciences and Informatics, 17(2), 253-271.

[20] Weihe, K. (1997). "Maximum (s, t)-flows in planar networks in $O(|V|log|V|)$ time." Journal of Computer and System Sciences, 55(3), 454-475.

[21] Wright, S. J. (1997). Primal-dual interior-point methods. Society for Industrial and Applied Mathematics.

[22] Italiano, G. F., Nussbaum, Y., Sankowski, P., & Wulff-Nilsen, C. (2011). "Improved algorithms for min cut and max flow in undirected planar graphs." In Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing, 313-322.

[23] Zhang, P., & Tang, L. (2020). Minimum label s-t cut has large integrality gaps. Information and Computation, 275, 104543.