Protecting the Connectivity of a Graph Under Non-Uniform Edge Failures

Felix Hommelsheim
University of Bremen, Germany

Zhenwei Liu
Zhejiang University, Hangzhou, China
University of Bremen, Germany

Nicole Megow
University of Bremen, Germany

Guochuan Zhang
Zhejiang University, Hangzhou, China

Abstract

We study the problem of guaranteeing the connectivity of a given graph by protecting or strengthening edges. Herein, a protected edge is assumed to be robust and will not fail, which features a non-uniform failure model. We introduce the (p, q)-Steiner-Connectivity Preservation problem where we protect a minimum-cost set of edges such that the underlying graph maintains p-edge-connectivity between given terminal pairs against edge failures, assuming at most q unprotected edges can fail. We design polynomial-time exact algorithms for the cases where p and q are small and approximation algorithms for general values of p and q. Additionally, we show that when both p and q are part of the input, even deciding whether a given solution is feasible is NP-complete. This hardness also carries over to Flexible Network Design, a research direction that has gained significant attention. In particular, previous work focuses on problem settings where either p or q is constant, for which our new hardness result now provides justification.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Design and analysis of algorithms

Keywords and phrases Network Design, Edge Failures, Graph Connectivity, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.51

Related Version *Full Version*: `https://arxiv.org/abs/2501.04540`

Funding *Zhenwei Liu*: Research supported in part by NSFC (No. 12271477).
Guochuan Zhang: Research supported in part by NSFC (No. 12271477).

1 Introduction

In today's interconnected world, the robustness of infrastructures is crucial, particularly in the face of potential disruptions. This applies to road networks, communication grids, and electrical systems alike, where the ability to maintain functionality despite failures is paramount. Resilience in critical infrastructures requires the incorporation of redundancy to withstand unforeseen challenges. Survivable Network Design (SND) addresses the fundamental need of ensuring not just connectivity but robust connectivity. It goes beyond the conventional concept of linking two entities by recognizing the need for multiple, resilient connections.

Beyond its practical applications, SND is a fundamental problem in combinatorial optimization and approximation algorithms. In its classical setting, we are given a graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}$ and connectivity requirements $k(s, t)$ for each vertex pair $s, t \in V$. The goal is to find a minimum-cost subset of edges $X \subseteq E$ such that $H = (V, X)$

is $k(s,t)$-connected for each $s,t \in V$, i.e., in H there are $k(s,t)$ many edge-disjoint paths for each vertex pair $s,t \in V$. This means that s and t are still connected in H after the deletion of any $k(s,t)-1$ edges of H. SND is APX-hard and the current best approximation factor of 2 is achieved by Jain's famous iterative rounding algorithm [32]. It is a long-standing open question whether this factor 2 can be improved, even for many special cases of SND.

Instead of building a new network from scratch, many real-world applications as well as the research community consider augmentation problems, in which we are already given some network, and the task is to robustify the network to withstand failures. The most common model is to increase the connectivity of a given network by adding more links [22,25]. However, adding new links to a real-world network may be costly or even impractical.

This motivates the investigation of the problem of increasing the connectivity of a network by *protecting* or strengthening edges, as has been proposed by Abbas et al. [1] in a practical context. In this paper, we formally introduce the problem *(p,q)-Steiner-Connectivity Preservation ((p,q)-SCP)*: Given an undirected graph $G=(V,E)$, possibly with parallel edges, nonnegative costs $c: E \to \mathbb{R}_+$ and k terminal pairs $(s_i,t_i) \in V \times V$. The task is to identify a minimum-cost set of edges $X \subseteq E$ such that for any edge set $F \subseteq E \setminus X$ with $|F| \le q$, there are p edge-disjoint paths between any terminal pair (s_i,t_i) in $(V, E \setminus F)$. In other words, the task is to protect a minimum-cost subset of the edges $X \subseteq E$ such that, no matter which q unprotected edges from $E \setminus X$ are removed from G, there are still p edge-disjoint paths between any terminal pair. We refer to the special case with a single terminal pair (s,t) by *(p,q)-s-t-Connectivity Preservation ((p,q)-stCP)* and the other extreme case in which *each* pair of vertices is a terminal pair as *(p,q)-Global-Connectivity Preservation ((p,q)-GCP)*. Using this notion, Abbas et al. [1] considered $(1,q)$-GCP and proposed to start from a minimum spanning tree and remove unnecessary edges, which does not admit bounded approximation factors. Zhang et al. [44] used mixed-integer linear programming to solve $(1,q)$-GCP and $(1,q)$-SCP. Bienstock and Diaz [12] considered a special case of $(1,q)$-SCP, the all-pair connectivity among a set of terminals, and showed a polynomial-time algorithm for $q \le 2$ and the NP-hardness for $q = 8$.

The distinction between protected and unprotected edges has a similar flavor as the (p,q)-Flexible Network Design ((p,q)-FND) problem [2,3,6–9,13,17]. The input is an edge-weighted undirected graph together with a set of terminal pairs, where the edges are either safe or unsafe. The goal is to find a minimum-cost subgraph such that any terminal pair remains p-edge-connected after the failure of any q *unsafe* edges. Also here different versions have been studied, e.g., (p,q)-GFND, the global connectivity version (each pair of vertices is a terminal pair) or (p,q)-stFND, the *s-t* version (only one terminal pair). In contrast to their model, in our setting we strengthen *existing* edges rather than building a network from scratch. For $p=1$, $(1,q)$-SCP reduces to $(1,q)$-FND. Given an instance of $(1,q)$-SCP, we construct an instance of $(1,q)$-FND as follows. We use the same underlying graph but replace each edge by two parallel edges: one unsafe edge of cost zero and one safe edge with cost equal to the cost of protecting the edge in the instance of $(1,q)$-SCP. Since the unsafe edges have cost zero, we can assume that any feasible solution includes all unsafe edges. Then, a feasible solution to $(1,q)$-FND can be transformed into a feasible solution to $(1,q)$-SCP with the same cost by protecting the selected safe edges and vice versa. For $p>1$, however, we are not aware of any reduction from (p,q)-SCP to (p,q)-FND.

1.1 Our Contribution

In this paper, we study Connectivity Preservation problems in terms of algorithms, complexity, and approximability.

The $(1, q)$-Steiner-Connectivity Preservation problem is APX-hard if q is part of the input: When q is sufficiently large, say, larger than $|E|$, any feasible solution to $(1, q)$-SCP includes at least one edge in any terminal-separating cut (precise definitions are given in Section 2). Hence, it is equivalent to Steiner Forest, which is APX-hard [11]. Similarly, (p, q)-GCP is APX-hard for any $p \geq 2$, as it contains the minimum p-edge-connected spanning subgraph problem [39] as a special case when q is sufficiently large. We strengthen this by showing that $(1, q)$-GCP is also NP-hard when q is part of the input.

Motivated by the problem hardness, we first study cases when p or q is small. We obtain polynomial-time exact algorithms for $(p, 1)$-SCP for any $p \geq 1$ and $(1, 2)$-SCP as well as a polynomial-time exact algorithm for $(2, 2)$-GCP. We emphasize that (p, q)-SCP generalizes (p, q)-GCP and hence any algorithm for (p, q)-SCP also works for (p, q)-GCP.

▶ **Theorem 1** (summarized). *There are polynomial-time exact algorithms for $(p, 1)$-Steiner-Connectivity Preservation for any $p \geq 1$ and $(1, 2)$-Steiner-Connectivity Preservation. Furthermore, there is a polynomial-time exact algorithm for $(2, 2)$-Global-Connectivity Preservation.*

The first result for $(p, 1)$-SCP is quite easily obtained by observing that a solution is only feasible if every edge contained in some terminal-separating cut of size at most p is protected.

The polynomial-time algorithm for $(1, 2)$-SCP is more involved and relies on a decomposition of terminal-separating cuts of size 2. We reduce the problem to the case in which G is assumed to be 2-edge-connected. Then, it remains to protect one edge in each terminal-separating cut of size 2. To do so, we decompose the problem into subproblems that consist either of a 2-edge-connected component or a cycle that can be solved independently.

The polynomial-time algorithm for $(2, 2)$-GCP is the most involved exact algorithm. We first show that we can assume without loss of generality that G is 3-edge-connected, which reduces the problem to selecting a minimum-cost set of edges containing at least 2 edges from each 3-edge-cut. Equivalently, we select a maximum-weight set of edges such that we pick at most 1 edge from each 3-edge-cut. Using the well-known tree-representation of minimum cuts [20], we model this problem as a multi-commodity flow problem on a tree: given a tree and a set of weighted paths on the tree, find a maximum weighted subset of paths that are pairwise edge-disjoint, which was first introduced in [28] for the unweighted case. We solve the weighted problem via dynamic programming, which might be of independent interest.

We complement our exact algorithms for small p and q with hardness and approximation results for more general cases. For (p, q)-SCP, if both p and q are part of the input, we show that there is no polynomial-time algorithm verifying the feasibility of any given solution, unless P=NP, even in the case of s-t-connectivity. This rules out an α-approximation algorithm for any α. Our technique is based on a reduction from k-Clique on d-regular graphs. Note that the corresponding solution verifying problems of (p, q)-stCP and (p, q)-stFND are essentially the same: given sets of protected (resp. safe) and unprotected (resp. unsafe) edges, decide whether there is an s-t-cut that has no more than $p + q - 1$ edges and has less than p protected (resp. safe) edges. Hence, the hardness of verifying a solution also carries over to (p, q)-s-t-Flexible Network Design and justifies why previous work on this problem only discusses the cases where either p or q is constant [3, 17].

▶ **Theorem 2.** *When both p and q are part of the input, verifying the feasibility of a solution to (p, q)-s-t-Connectivity Preservation or (p, q)-s-t-Flexible Network Design is* NP*-complete, even in perfect graphs. Hence, they do not admit an α-approximation for any α unless* P = NP.

On the algorithmic side, if q is a constant, we can enumerate all "bad" edge sets whose removal destroy the p-edge-connectivity. Since any feasible solution intersects with or hits these "bad" sets, it reduces to the hitting set problem and admits a q-approximation. Then we focus on the case where p is a constant. We first give a q-approximation for $(1, q)$-SCP and extend it to (p, q)-SCP based on a primal-dual framework [29,43]. In the framework, we iteratively protect one more edge in each *critical* cut (precise definition follows), which is very similar to the problem $(1, q)$-SCP. We obtain the following result.

▶ **Theorem 3.** *There is a polynomial-time $\mathcal{O}((p + q) \log p)$-approximation algorithm for (p, q)-Steiner-Connectivity Preservation when p is a constant.*

The global connectivity problem (p, q)-GCP has more symmetric structure, which enables us to remove the requirement of p being constant. Hence, the above result directly carries over to this case without the assumption on p. In addition, we design a different set-cover based augmentation process. This algorithm relies on the fact that there is only a polynomial number of critical cuts to be augmented, which is not true for (p, q)-SCP. Combining the two algorithms, we obtain the following result.

▶ **Theorem 4.** *There is a polynomial-time $\mathcal{O}(\log p \cdot \min\{p+q, \log n\})$-approximation algorithm for (p, q)-Global-Connectivity Preservation.*

We obtain further results for special cases by showing reductions to known problems. Since the *Augmenting Small Cuts* problem [8] generalizes $(1, q)$-Global-Connectivity Preservation, we obtain a 5-approximation building on [37]. Further, we show that $(1, q)$-s-t-Connectivity Preservation is equivalent to the *Minimum Shared Edge* problem; formally defined in Section 4. This reduction implies, due to earlier work, a fixed-parameter tractable (FPT) algorithm parameterized by q if the graph is undirected [23] and an XP-algorithm (slice-wise polynomial) for directed graphs [5]. Further, for directed graphs the equivalence of the two problems implies a strong inapproximability bound of $\Omega(2^{\log^{1-\epsilon} \max\{q,n\}})$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$ [38]. Since $(1, q)$-s-t-Connectivity Preservation is a special case of $(1, q)$-s-t-Flexible Network Design, namely, $(1, q)$-Flexible Network Design with only a single terminal pair, this strong hardness bound also holds for $(1, q)$-stFND, where the best-known lower bound on the approximation ratio is $\Omega(\log^{2-\varepsilon} q)$ unless $\mathsf{NP} \subseteq \mathsf{ZTIME}(n^{\mathsf{polylog}(n)})$ [3].

1.2 Related Work

The (p, q)-Steiner-Connectivity Preservation problem generalizes many well-known problems from survivable network design (SND), which itself generalizes a collection of connectivity problems such as the minimum spanning tree problem, the Steiner tree and forest problem, or the minimum k-edge-connected spanning subgraph problem (k-ECSS).

Many special cases of SND remain APX-hard. This includes many augmentation problems, where typically the task is to increase the connectivity of a graph by at least 1. If the set of edges to be added is unrestricted, the problem can be solved even in near-linear time [15,22], whereas the problem is APX-hard otherwise [25,34]. Well-studied such problems include the Connectivity Augmentation problem [40,42] and the Tree Augmentation problem [14,41].

A problem of similar flavor was introduced by Adjiashvili, Stiller and Zenklusen [4], who initiated the study of highly non-uniform failure models, called bulk-robustness. Therein, a family of edge sets $\mathcal{F}$ is given and the goal is to find a minimum-cost spanning subgraph H such that $H \setminus F$ is connected for any $F \in \mathcal{F}$. They proposed an $\mathcal{O}(\log n + \log m)$-approximation algorithm for a generalized matroid setting. They also studied an s-t-connectivity variant and obtained an $\mathcal{O}(w^2 \log n)$-approximation algorithm, where $w = \max_{F \in \mathcal{F}} |F|$. Recently, Chekuri and Jain [18] considered the connectivity between multiple vertex pairs and achieved an $\mathcal{O}(w^2 \log^2 n)$-approximation algorithm.

The aforementioned Flexible Network Design problem can be viewed as a problem between survivable network design and bulk-robustness, as it divides the edge set into safe and unsafe edges and only q unsafe edges can fail simultaneously. Since the work by Adjiashvili, Hommelsheim and Mühlenthaler [2] for $(1,1)$-GFND, there has been a lot of work on (p,q)-GFND. Most research focused on the case where either p or q is a small constant. Boyd et al. [13] obtained $(q+1)$-approximation for $(1,q)$-GFND, a 4-approximation for $(p,1)$-GFND and $\mathcal{O}(q\log n)$-approximation for (p,q)-GFND. Very recently, Bansal et al. [9] showed an improved 7-approximation algorithm for $(1,q)$-GFND. We refer to [6,8,9,17] for a collection of results, including constant approximation for $(p,2),(p,3)$-GFND, $\mathcal{O}(q)$-approximation for $(2,q)$-GFND and $\mathcal{O}(p)$-approximation for $(p,4)$-GFND for even p. Parallel to (p,q)-GFND, Adjiashvili et al. [3] considered the s-t-connectivity variant, to which some results in [17] translate.

Another closely related problem is the Capacitated k-Connected Subgraph problem (Cap-k-ECSS) [16]. In this problem, we are given an undirected graph $G=(V,E)$ with edge costs $c:E\to\mathbb{R}_+$ and edge capacities $u:E\to\mathbb{Z}_+$. The goal is to find a minimum-cost spanning subgraph in which the capacity of any cut is at least k. Boyd et al. [13] pointed out that $(1,q)$-GFND (hence also $(1,q)$-GCP) can be reduced to Cap-$(q+1)$-ECSS by setting the capacity of safe and unsafe edges to $q+1$ and 1, respectively. For Cap-k-ECSS, the best-known approximation algorithms are $\mathcal{O}(\log n)$-approximation by Chakrabarty et al. [16] and $\mathcal{O}(\log k)$-approximation by Bansal et al. [9].

2 Preliminaries: Notation, Cut Formulation, Hardness

Graph notation. For an undirected graph $G=(V,E)$ and a vertex set $S\subset V$, we use $\delta_G(S)$ to denote the set of edges with exactly one endpoint in S. We write $\delta(S)$ if the underlying graph is clear from the context. An edge cut C is a subset of edges such that $G\setminus C$ has at least 2 connected components. If $|C|=1$, we call $\{e\}=C$ a *bridge*. Further, if there is some terminal pair (s_i,t_i) such that s_i and t_i are in different connected components in $G\setminus C$, we say C is terminal-separating. Let $e=(u,v)\in E$. We use the notation of G/e to denote the graph obtained from G by contracting e, i.e., by deleting e and identifying u,v. Let $G'=(V',E')$ be some subgraph of G. We slightly abuse the notation of contraction and use G/G' to represent the graph obtained from G by contracting all edges in E'. Let $e\in E(G)$. We use $G-e$ to denote the graph $G\setminus\{e\}$. Similarly, we define $G+e$.

An equivalent cut formulation. Given an instance of (p,q)-Steiner-Connectivity Preservation, we define $\mathcal{S}=\{S\subset V\mid \exists i,|S\cap\{s_i,t_i\}|=1,|\delta(S)|\le p+q-1\}$ as the set of *critical* (terminal-separating) cuts. Our problem can be equivalently formulated as the following cut-based integer program, in which we have to cover all critical cuts:

$$\begin{aligned}\min\quad & \sum_{e\in E} c_e x_e \\ \text{s.t.}\quad & \sum_{e\in\delta(S)} x_e \ge p \quad \forall S\in\mathcal{S} \\ & x_e\in\{0,1\} \qquad \forall e\in E.\end{aligned}\tag{CutIP}$$

▶ **Proposition 5.** (CutIP) *characterizes the feasible solutions of (p,q)-SCP. Moreover, a solution is feasible if and only if any critical cut contains at least p protected edges.*

Proof. We show that an edge set X is a feasible solution if and only if for any vertex set $S \in \mathcal{S}$, $|X \cap \delta(S)| \geq p$. Consider a feasible solution X and suppose that there is some $S \in \mathcal{S}$ with $|\delta(S)| \leq p+q-1$ and $|\delta(S) \cap X| < p$. Then, after removing no more than q edges from $\delta(S) \setminus X$, the remaining graph has a cut with less than p edges. Further, this cut separates some terminal pair. Thus, X is not a feasible solution.

Suppose for all $S \in \mathcal{S}$ we have $|\delta(S) \cap X| \geq p$. We show that after removing at most q unprotected edges, the remaining graph is still p-edge-connected between any terminal pair. For any cut $\delta(S)$ with $|\delta(S)| \geq p+q$, there are at least p remaining edges since we remove at most q edges. Fix any terminal pair s, t and any edge set $D \subseteq (E \setminus X)$ with $|D| \leq q$. We show that $|\delta(S) \setminus D| \geq p$ for any s-t-cut S, which implies p-edge-connectivity between s, t. If $|\delta(S)| \geq p+q$, then $|\delta(S) \setminus D| \geq p$. If $|\delta(S)| \leq p+q-1$, then $|\delta(S) \cap X| \geq p$ by the constraint of (CutIP). Thus it also holds that $|\delta(S) \setminus D| \geq p$. ◀

Given a partial solution $X \subseteq E$, we call a cut *safe* (w.r.t. X) if it is not critical or it contains at least p edges in X. Otherwise, we call it *unsafe*.

NP-hardness. In addition to the aforementioned complexity observations, we now show that $(1,q)$-GCP is NP-hard, even in the unweighted setting where we protect a minimum number of edges. Observe that any spanning tree of G is a feasible solution, which implies $\textsc{Opt} \leq |V|-1$. However, we show that it is NP-complete to distinguish whether $\textsc{Opt} = |V|-1$ or $\textsc{Opt} < |V|-1$, by a reduction from the largest bond problem [21]. Therein, we are given an undirected graph $G = (V, E)$ and an integer $k \geq 1$. A bond is an edge set represented by $\delta(S)$ for some $S \subset V$ with both $G[S]$ and $G[V \setminus S]$ being connected. The task is to decide whether there is a bond of size at least k.

We outline the idea for the hardness proof as follows. Given an instance of the largest bond problem, we reduce it to an instance of $(1,q)$-GCP using the same graph with $q := k-1$. If there is a bond $\delta(S)$ of size at least $k = q+1$, then protecting a spanning tree of $G[S]$ and $G[V \setminus S]$ is feasible, as the cut $\delta(S)$ is not critical, which implies $\textsc{Opt} < |V|-1$. If $\textsc{Opt} < |V|-1$, then the protected edges in the optimal solution induce multiple connected components. We can find a bond of size at least $q+1$ by contracting the connected components induced by the protected edges and computing the minimum cut of the resulting graph.

▶ **Theorem 6.** *Unweighted* $(1,q)$*-Global-Connectivity Preservation is* NP*-hard.*

Proof. Given an instance of the largest bond problem, we construct an instance of $(1,q)$-Global-Connectivity Preservation using the same graph with $q = k-1$.

If there is a bond $\delta(S)$ of size at least k, then the optimal solution value of the $(1,q)$-Global-Connectivity Preservation instance is no more than $|V|-2$, as we can simply protect a spanning tree of $G[S]$ and a spanning tree of $G[V \setminus S]$), which exist since $\delta(S)$ is a bond.

If there is no bond of size at least k, we claim that the optimal solution of the instance of $(1,q)$-Global-Connectivity Preservation must be a spanning tree using $|V|-1$ edges. Suppose the optimal solution is not a spanning tree, and it consists of connected components $S_1, S_2, \ldots, S_t$, $t \geq 2$. After contracting $S_1, S_2, \ldots, S_t$, the graph has to be k-edge-connected, by feasibility. Let G' be this graph. Note that in any graph there must be a minimum cut $Y \subseteq E$ such that $G \setminus Y$ consists of exactly 2 connected component. Hence, there is also such a minimum cut Y in G' and this cut has size at least k, as G' is k-edge-connected. But then Y corresponds to a bond of size at least k in the original graph, a contradiction. ◀

3 Exact Algorithms for small q

In this section we design three polynomial-time exact algorithms for different cases depending on p and q, i.e., we prove Theorems 7, 10, and 17, which together imply Theorem 1.

3.1 $(p, 1)$-Steiner-Connectivity Preservation

To give some intuition, we first show a simple algorithm for $(p, 1)$-Steiner-Connectivity Preservation. By Proposition 5, an instance is feasible if and only if there is no terminal-separating cut of size less than p. Hence, from now on we assume the instance is feasible.

The set of critical cuts is given by $\mathcal{S} = \{S \subset V \mid \exists i, |S \cap \{s_i, t_i\}| = 1, |\delta(S)| \leq p\}$. Hence, any feasible solution must contain *all* edges of any critical cut. Therefore, the only inclusion-wise minimal solution consists of all edges in any terminal-separating cut of size p and it remains to find all such edges. To this end, we assign different *capacities* to protected edges and unprotected edges such that any safe cut has a strictly larger capacity than that of any unsafe cut. The algorithm works as follows.

Algorithm 1. Let X be the current partial solution; initially $X = \emptyset$. In each iteration, we set the capacity of the edges to $\frac{p+1}{p}$ for all $e \in X$ and 1 otherwise. For every terminal pair s, t, we solve the Minimum s-t-Cut Problem using standard polynomial-time algorithms. If we find a terminal-separating cut of capacity less than $p + 1$, then this defines an unsafe critical cut $\delta(S)$ and we protect all edges in it, i.e., we add $\delta(S)$ to X and repeat. If each terminal-separating cut has capacity at least $p + 1$, output X.

▶ **Theorem 7.** *Algorithm 1 is a polynomial-time exact algorithm for $(p, 1)$-Steiner-Connectivity Preservation.*

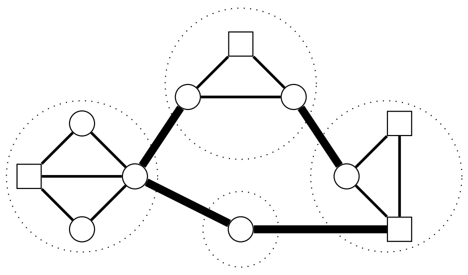
Proof. Algorithm 1 runs obviously in polynomial time. Note that we can decide the feasibility of the given instance by enumerating terminal pairs and checking whether there is a terminal-separating cut of size less than p. We now assume there is none and the instance is feasible.

Let $X \subseteq E$ be a partial solution. We claim that the capacity function in Algorithm 1 distinguishes safe and unsafe cuts with respect to X. Specifically, a cut is unsafe if and only if its capacity is strictly less than $p + 1$. By the feasibility of the instance, any terminal-separating cut has at least p edges. Let C be any terminal-separating cut. If $|C| > p$ or $C \subseteq X$, then the capacity of C is at least $p+1$. If $|C| = p$ and $|C \cap X| < p$, then its capacity is smaller than $p + 1$. Thus we can find an unsafe terminal-separating cut by enumerating terminal pairs s, t and computing a minimum s-t cut with respect to the capacity function. By the preceding discussion, Algorithm 1 finds an optimum solution in polynomial time. ◀

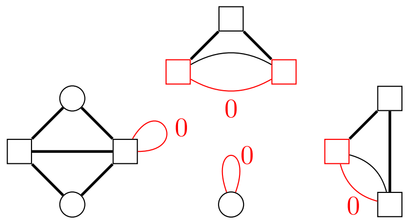
3.2 $(1, 2)$-Steiner-Connectivity Preservation

In this subsection we present a polynomial-time algorithm for $(1, 2)$-Steiner-Connectivity Preservation. The set of critical cuts is $\mathcal{S} = \{S \subset V \mid \exists i, |S \cap \{s_i, t_i\}| = 1, |\delta(S)| \leq 2\}$. Hence, we distinguish between bridges and 2-edge-cuts. We first show that we can reduce to the case that the input graph G is 2-edge-connected.

Given any bridge e of G, if e separates some terminal pair, any feasible solution has to include e. In this case, we pay $c(e)$ and consider the new instance defined by G/e. If there is no such terminal pair, then any inclusion-wise minimal feasible solution should not include e, which implies that we can delete e and consider the two connected components of $G - e$ individually. As a result, we can assume that the input graph G is 2-edge-connected. Note that if the graph is 3-edge-connected, then there is no critical cut and we are done.



(a) The original instance: the bold lines are the edges in the cycle and the dotted bubbles are the 2-edge-connected components. Each pair of rectangles is a terminal pair.



(b) Reduction to independent instances: red edges are pseudo-edges added to the 2-edge-connected components with zero cost. Red rectangles are new terminals.

Figure 1 Illustration of the decomposition (Lemma 8, Lemma 9).

Given a terminal-separating 2-edge-cut $\{e_1, e_2\}$ of G, at least one of e_1 and e_2 has to be contained in any feasible solution. However, deciding which edge to protect is non-trivial. We show how to further decompose our instance into smaller and independent instances according to the following structural lemma. See Figure 1a for an illustration.

▶ **Lemma 8.** *Given an undirected graph G which is 2-edge-connected but not 3-edge-connected, and a 2-edge-cut $\{e_1, e_2\}$ of G, there is a polynomial-time algorithm to decompose G into disjoint 2-edge-connected subgraphs $G_1, \ldots, G_k$ such that after contracting $G_1, \ldots, G_k$ the resulting graph $G/\bigcup_{i=1}^{k} G_i$ forms a cycle and e_1, e_2 belong to this cycle.*

Proof. Consider the graph $G' := G \setminus \{e_2\}$, which is connected but not 2-edge-connected. Let G'' arise from G' by contracting each 2-edge-connected component. Note that G'' is isomorphic to a tree. Since $G = G' \cup \{e_2\}$ is 2-edge-connected, G'' must be a path. Further, e_2 connects the two end-vertices of the path and e_1 is a path edge (e_1 is a bridge of G''). Let the nodes of the path G'' be $v_1, \ldots, v_k$ and for $1 \leq i \leq k$ let G_i be the 2-edge-connected component represented by v_i, respectively. We conclude that $G/\bigcup_{i=1}^{k} G_i$ forms a cycle and e_1, e_2 belong to this cycle. ◀

Given a decomposition as in Lemma 8, we claim that the problem reduces to solving certain subproblems defined by $G_1, \ldots, G_k$ (plus some additional pseudo-edge for each component) and the subproblem restricted to the cycle C. To do so, we view our problem as finding a minimum-cost edge set that intersects all 2-edge-cuts. Observe that any inclusion-wise minimal 2-edge-cut is either two edges on the cycle C, or two edges in G_i for some i. Hence, we can solve our problem by solving (i) the subproblem defined by 2-edge-cuts on the cycle C and (ii) the subproblems defined by 2-edge-cuts in each component G_i separately.

The subproblem (i) is a Steiner Forest problem on a cycle. This follows from the observation that any feasible solution must contain a path consisting of only protected edges between each terminal pair. We can solve the min-cost Steiner Forest problem on a cycle by enumerating which cycle-edge is not in the optimum solution and breaking the cycle into a path. On a path, the solution is the union of the unique paths between the terminal pairs. Then, we recursively solve the subproblems (ii) in each G_i. However, we cannot simply recurse on G_i since a 2-edge-cut of G_i may not be a 2-edge-cut of G. Instead, we recourse on a new graph obtained by adding a zero-cost edge e_i to G_i. This edge e_i connects the two vertices that are incident to the edges of C, which represents the connection in G_i between these vertices via the cycle C. We formalize this idea in the following lemma (See Figure 1b).

▶ **Lemma 9.** *Given a decomposition as in Lemma 8, an optimum solution to* $(1, 2)$*-SCP can be obtained by combining optimum solutions of the following subproblems:*

(i) *protect a minimum-cost edge set that intersects with any terminal-separating* 2*-edge-cut on the cycle, and*

(ii) *for each* G_i*, let* $u_i, v_i \in V(G_i)$ *be the two vertices incident to the two edges in the cycle. Solve the problem on* $G'_i = G_i \cup \{(u_i, v_i)\}$ *with* $c_{(u_i,v_i)} = 0$*. Keep the terminal pairs with both terminals in* G_i*. For terminal pairs* (s_i, t_i) *with* $s_i \in G_i, t_i \notin G_i$*, replace it with* $(s_i, u_i), (s_i, v_i)$*.*

Proof. We first show that given any feasible solution X of G, the corresponding edges of X on each subproblem is a feasible solution for the subproblem. For the cycle subproblem (ii) this is trivial. For any subproblem $G'_i = G_i \cup \{(u_i, v_i)\}$, we show $X \cap G_i \cup \{(u_i, v_i)\}$ is a feasible solution. Observe that any 2-edge-cut C in G' cannot contain the edge $\{(u_i, v_i)\}$ since G_i is 2-edge-connected. Thus, C must also be a 2-edge-cut in G. If C separates some terminal pair in G'_i, so does it in G, which implies $C \cap X \neq \emptyset$. Therefore, each terminal-separating 2-edge-cut of G'_i is safe.

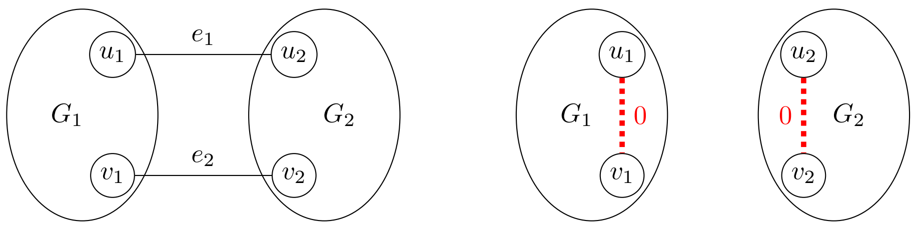
Given feasible solutions of the subproblems, we show how to obtain a feasible solution of G without increasing the cost. Let X be the edges protected in the subproblems except the new edges (u_i, v_i). Thus, the cost of X is at most the sum of the cost of the solutions to the individual subproblems. It remains to argue that X is feasible for G. Let C be any terminal-separating 2-edge-cut of G. If C is on the cycle, by the feasibility of the subproblem on the cycle, $C \cap X \neq \emptyset$. If C is in G_i for some i, C must also be a terminal-separating 2-edge-cut of G'_i. Thus $C \cap X \neq \emptyset$. We conclude that X is feasible for G. ◀

Algorithm 2. We first protect terminal-separating bridges, contract them, and consider the 2-edge-connected components separated by non-terminal-separating bridges individually. This reduces to the case that G is 2-edge-connected. Then, as long as we find a terminal-separating 2-edge-cut (which is the only type of critical cut), we decompose the problem into a subproblem on a cycle and a collection of subproblems in smaller 2-edge-connected components. Then we recursively solve the individual subproblems. The decomposition stops either if G is 3-edge-connected (and hence we are done as there is no critical cut) or each component on the cycle consists of a single vertex, i.e., if G is a cycle. The cycle case is solved by enumerating which edge of the cycle is *not* contained in an optimum solution and then solving a Steiner Forest problem on a path where the optimal solution is trivial. Among all such solutions, we output the one with minimum cost.

▶ **Theorem 10.** *Algorithm 2 is a polynomial-time exact algorithm for* $(1, 2)$*-Steiner-Connectivity Preservation.*

We remark that Bienstock and Diaz [12] studied a special case of $(1, q)$-SCP. They showed that it is NP-hard when $q = 8$ and they conjectured the NP-hardness for $q = 3$.

Interestingly, $(1, 2)$-Global-Connectivity Preservation admits an easier algorithm. We view the problem as finding a minimum-cost edge set hitting all 2-edge-cuts, which reduces to a special case of Minimum Weighted Vertex Cover. Therein, each edge e of G corresponds to a vertex v_e in the Vertex Cover instance G' and there is an edge between two vertices v_e and $v_{e'}$ if and only if $\{e, e'\}$ forms a 2-edge-cut in G. The Vertex Cover instance G' has a special structure with each connected component being a complete graph. To see this observe that, if both $\{e_1, e_2\}$ and $\{e_1, e_3\}$ are 2-edge-cuts, then $\{e_2, e_3\}$ is also a 2-edge-cut ([35, Lemma 2.37]). The optimal vertex cover solution in a complete graph is trivial: select all vertices except the largest-weighted one. We conclude with the following result.



(a) The original instance. **(b)** Reduction to two sub-instances.

Figure 2 Illustration of Lemma 12: Equivalence to solving two new instances on $G_1 \cup \{(u_1, v_1)\}$ where (u_1, v_1) is an edge with zero cost and $G_2 \cup \{(u_2, v_2)\}$ where $\{(u_2, v_2)\}$ has zero cost.

▶ **Lemma 11.** *The greedy algorithm that selects from each* 2*-edge-cut the cheaper edge solves* $(1, 2)$*-Global-Connectivity Preservation.*

3.3 (p, q)-Global-Connectivity Preservation when $p, q \leq 2$

We now present a polynomial-time algorithm for $(2, 2)$-GCP. Note that for all other $p, q \leq 2$, our previous results imply a polynomial-time algorithm for (p, q)-GCP. We outline our algorithm as follows. By Proposition 5, we can assume the input graph G to be 2-edge-connected, as otherwise, the instance is infeasible. Further, a feasible solution contains at least two edges in each 2-edge-cut and in each 3-edge-cut. We first show that if there is some 2-edge-cut, it is equivalent to solving two smaller independent instances (Lemma 12). Hence, we can assume that the input graph G is 3-edge-connected. Then we represent all the 3-edge-cuts using a standard tree representation [20, 30] and it remains to solve a weighted multi-commodity flow problem on the tree (introduced formally later, Lemma 14). Finally, we solve the weighted multi-commodity flow problem via dynamic programming (Lemma 16).

Suppose G is not 3-edge-connected and there is some 2-edge-cut $\{e_1, e_2\}$, i.e., $G \setminus \{e_1, e_2\}$ consists of 2 connected components G_1, G_2. Let $e_1 = (u_1, u_2)$ with $u_1 \in G_1$ and $u_2 \in G_2$, $e_2 = (v_1, v_2)$ with $v_1 \in G_1$ and $v_2 \in G_2$ (see Figure 2). We create two new instances: I_1 on $G_1 \cup \{(u_1, v_1)\}$ where (u_1, v_1) is an edge with zero cost and I_2 on $G_2 \cup \{(u_2, v_2)\}$, where $\{(u_2, v_2)\}$ has zero cost. We show that it suffices to solve I_1, I_2 independently and combine their solutions to get a solution to the original instance.

▶ **Lemma 12.** $\text{Opt}(I) = c(e_1) + c(e_2) + \text{Opt}(I_1) + \text{Opt}(I_2)$.

Proof. Given a feasible solution X of I, we show that $X_1 = X \cap E(G_1) + (u_1, v_1)$ and $X_2 = X \cap E(G_2) + (u_2, v_2)$ are feasible solutions for I_1 and I_2, respectively, implying $c(X) = c(e_1) + c(e_2) + c(X_1) + c(X_2)$ and $\text{Opt}(I) \geq c(e_1) + c(e_2) + \text{Opt}(I_1) + \text{Opt}(I_2)$. We show feasibility for X_1; the feasibility for X_2 is analogous. It suffices to show that for any critical cut in $G_1 + (u_1, v_1)$, at least 2 edges are protected. Consider any critical cut C of $G_1 + (u_1, v_1)$. If C does not contain the new edge (u_1, v_1), then it is also a critical cut of G and therefore this cut is safe. Hence, we assume that C contains the new edge (u_1, v_1) and let $C = \{f_1, f_2, (u_1, v_1)\}$. Note that f_2 might not exist if $|C| = 2$. We show that $\{f_1, f_2, e_1\}$ is a critical cut in G. Consider $G_1' = (G_1 \setminus \{f_1, f_2\}) + (u_1, v_1)$ and observe that (u_1, v_1) is a bridge in G_1', as C is a cut in $G_1 + (u_1, v_1)$. Hence, u_1 and v_1 are only connected in G_1' via the edge (u_1, v_1). This means that also in $G \setminus \{f_1, f_2\}$, any path connecting u_1 and v_1 must use e_1. Hence, e_1 is a bridge in $G \setminus \{f_1, f_2\}$ and therefore $\{f_1, f_2, e_1\}$ is a critical cut in G. Thus $C \setminus (u_1, v_1) + e_1$ contains at least 2 protected edges, which implies $C \setminus (u_1, v_1)$ contains at least 1 protected edge. Since (u_1, v_1) is protected, C has at least 2 protected edges and is safe.

On the other hand, given solutions X_1 and X_2 of I_1 and I_2, respectively, we show that $X = X_1 \cup X_2 + e_1 + e_2$ is feasible for I, implying $c(X) = c(e_1) + c(e_2) + c(X_1) + c(X_2)$ and $\textsc{Opt}(I) \leq c(e_1) + c(e_2) + \textsc{Opt}(I_1) + \textsc{Opt}(I_2)$. Let C be any critical cut of G. Without loss of generality, we assume that C is inclusion-wise minimal, i.e., it does not contain any smaller critical cut. We distinguish the following three cases. First, assume that $C \subseteq G_1$. Then, C must also be an edge cut of $G_1 \cup (u_1, u_2)$ and therefore C is safe. The case $C \subseteq G_2$ is analogous. For the second case, we assume that the first case does not apply and further assume that $C \cap \{e_1, e_2\} = \emptyset$. Hence, either $|C \cap G_1| = 1$ or $|C \cap G_2| = 1$. Without loss of generality assume $|C \cap G_1| = 1$. Observe that $(C \cap G_2) + (u_2, v_2)$ is a critical edge cut in $G_2 + (u_2, v_2)$ and $(C \cap G_1) + (u_1, v_1)$ is a critical edge cut in $G_1 + (u_1, v_1)$. Further, by feasibility of X_1 and X_2, the only edge in $C \cap G_1$ and at least one of the two edges in $C \cap G_2$ must be protected, which implies C contains at least 2 protected edges and is safe. In the third and final case, we assume that none of the previous cases apply and further assume that C contains either e_1 or e_2. Any cut containing both e_1 and e_2 is safe, as both are protected in X. Without loss of generality assume $e_1 \in C$. We claim that either $C - e_1 \subseteq E(G_1)$ or $C - e_1 \subseteq E(G_2)$. Otherwise, C contains one edge e_3 in G_1 and one edge e_4 in G_2. Observe that $\{e_1, e_3\}$ is a 2-edge-cut of G, which contradicts the fact that C is inclusion-wise minimal. If $C - e_1 \subseteq E(G_1)$, then similar to the first part of this proof one can show that $C - e_1 + (u_1, v_1)$ is a cut in $G_1 + (u_1, v_1)$. Hence, $|X_1 \cap C| \geq 1$ and therefore, $|X \cap C| \geq 2$, as $e_1 \in X$. The case $C - e_1 \subseteq E(G_2)$ is analogous and hence this concludes the proof. ◀

By repeating the above process, we end up with a 3-edge-connected graph. The following tree representation gives us a clear structure about all the 3-edge-cuts and it can be computed in near-linear time [30].

▶ **Definition 13** (Tree representation of min cuts [20])**.** *Let $G = (V, E)$ be an undirected graph and suppose the capacity of its minimum cut is an odd number k. There is a polynomial-time algorithm that constructs a rooted tree $T = (U, F)$ together with a (not necessarily surjective) mapping $\phi : V \to U$. Further, there is a one-to-one correspondence between any k-edge-cut of G and $f \in F$ as follows. For any $f \in F$, let T_f be the subtree of T beneath f and let $V(T_f) = \{v \in V \mid \phi(v) \in T_f\}$. Then for any tree edge $f \in F$, $\delta_G(V(T_f))$ defines a k-edge-cut of G. For any k-edge-cut C of G, there is some tree edge $f \in F$ such that $C = \delta_G(V(T_f))$.*

Given this tree representation, we show now that our problem $(2, 2)$-Global-Connectivity Preservation reduces to the weighted multi-commodity flow problem on a tree. This problem is defined as follows: given a tree T, a set of paths $\mathcal{P}$ on the tree and a weight function $w : \mathcal{P} \to \mathbb{R}$, find a subset of pairwise edge-disjoint paths with maximum total weight.

▶ **Lemma 14.** *When the input graph G is 3-edge-connected, $(2, 2)$-Global-Connectivity Preservation reduces to the weighted multi-commodity flow problem on a tree.*

Proof. A solution $X \subseteq E$ is feasible if and only if it contains at least 2 edges in each 3-edge-cut. Equivalently, for each 3-edge-cut at most one edge is unprotected. We consider this complement problem in which we want to find a set of maximum-weight edges $\bar{X}$ such that each 3-edge-cut contains at most one edge of $\bar{X}$. We use the standard tree representation [20] of all the 3-edge-cuts of G, in which each 3-edge-cut of the original graph is represented by an edge in the tree. Given a tree representation $T = (U, F)$ and $e = (u, v) \in E$, define P_e as the path on T between $\phi(u)$ and $\phi(v)$ and let the weight of P_e be the cost of e. Observe that every 3-edge-cut containing e corresponds to a tree edge on P_e and vice versa. Therefore, a solution $X \subseteq E$ is feasible if and only if the set of paths $\{P_e \mid e \in \bar{X} = E \setminus X\}$ are pairwise edge-disjoint. Hence finding the optimal X reduces to finding a set of edge-disjoint paths on T, maximizing the total weight, which is the weighted multi-commodity flow problem. ◀

▶ Remark 15. We can prove that Lemma 14 holds more generally for any even p: If G is $(p+1)$-edge-connected, $(p,2)$-GCP reduces to the weighted multi-commodity flow problem on a tree for any even p. However, to solve $(p,2)$-GCP using this reduction, we need to reduce the problem to the case where G is $(p+1)$-edge-connected, which remains unclear for $p \geq 4$.

Garg et al. [28] considered an unweighted version of multi-commodity flow problem on a tree and obtained an exact polynomial-time greedy algorithm. However, their arguments do not extend to the weighted case. We design a dynamic program for the weighted version.

▶ **Lemma 16.** *The weighted multi-commodity flow problem on a tree can be solved in polynomial time.*

Proof. We root the tree at an arbitrary vertex r. Without loss of generality, we assume there is no path that consists of only one vertex since the selected paths have to be only edge-disjoint. For any vertex v, let T_v be the subtree rooted at vertex v. For any tree edge $e = (u, v)$ where u is closer to the root, we use T_e to represent the subtree $T_u \setminus T_v$ for short. Define the subproblem $\mathcal{I}(T_v)$ in the subtree T_v as follows: From the set of paths completely contained in T_v, select a maximum-weight subset of paths that are pairwise edge-disjoint. Let $f(T_v)$ be the optimal value of $\mathcal{I}(T_v)$. We define $\mathcal{I}(T_e)$ and $f(T_e)$ for each $e \in E(T)$ analogously. We only show how to compute $f(T_v)$; the computation of $f(T_e)$ is similar.

Fix some vertex v and consider the subproblem $\mathcal{I}(T_v)$. If v is a leaf, then $f(T_v) = 0$. Otherwise, let $z_1, \dots, z_k$ be the children of v. Let $\mathcal{P}_v$ be the set of paths intersecting v. We say a path P occupies the subtree T_{z_i} if it intersects T_{z_i}. Our first observation is that each subtree T_{z_i} can be occupied by at most one selected path, as otherwise the edge (v, z_i) is contained in multiple selected paths, which is infeasible. Since a path in $\mathcal{P}_v$ can occupy either two subtrees T_{z_i}, T_{z_j} for some $1 \leq i < j \leq k$ or only one subtree T_{z_i} for some i, we reduce the problem $\mathcal{I}(T_v)$ to an instance $\mathcal{M}(T_v)$ of Maximum Weighted Matching. For $\mathcal{M}(T_v)$, we create an auxiliary graph $G(T_v)$ as follows. For each i with $1 \leq i \leq k$, we create a vertex u_i corresponding to the subtree T_{z_i} and a dummy vertex u'_i. For each path in $\mathcal{P}_v$, if it occupies T_{z_i} and T_{z_j} for some i, j, we create an edge between u_i and u_j. If it only occupies one subtree T_{z_i}, we create an edge between u_i and u'_i. We also create an extra edge between u_i and u'_i which represents the case where no selected path occupies T_{z_i}. It is not hard to see that there is a one-to-one correspondence between a feasible choice over $\mathcal{P}_v$ in $\mathcal{I}(T_v)$ and a matching on the auxiliary graph $G(T_v)$. It remains to properly set the weights ω of the edges of $G(T_v)$ such that a maximum-weight matching in $G(T_v)$ corresponds to an optimum solution for $\mathcal{I}(T_v)$. To do so, we observe that for a given fixed feasible choice over $\mathcal{P}_v$, it remains to solve a collection of subproblems represented by $\mathcal{I}(T'_v)$ or $\mathcal{I}(T'_e)$ for some $v', e' \in T_v$ and combine their optimal solutions. Formally, let $P = (v_1, \dots, v_\ell)$ be a path in $\mathcal{P}_v$ where $v = v_m$, $1 \leq m \leq \ell$. Assume P occupies two subtrees of v, say, $(v_1, \dots, v_{m-1}) \subseteq T_{z_i}$ and $(v_{m+1}, \dots, v_\ell) \subseteq T_{z_j}$. The case P only occupies one subtree of v follows analogously. Suppose we have selected P. Then it is still feasible to select paths completely contained in T_{z_i} and T_{z_j}, respectively, as long as they do not intersect P. This implies that the subproblems on $T_{z_i} \setminus E(P)$ and $T_{z_j} \setminus E(P)$ can be decomposed into $T_{v_1}, T_{(v_1,v_2)}, \dots, T_{(v_{m-2},v_{m-1})}$ and $T_{v_\ell}, T_{(v_\ell,v_{\ell-1})}, \dots, T_{(v_{m+2},v_{m+1})}$, respectively. See Figure 3 for an example. Hence, the gain of selecting P is the sum of the optimal values of these subproblems plus the weight of P itself. That is, we set $\omega((u_i, u_j)) := w(P) + f(T_{v_1}) + \sum_{k=1}^{m-2} f(T_{(v_k,v_{k+1})}) + f(T_{v_\ell}) + \sum_{k=m+1}^{\ell} f(T_{(v_k,v_{k+1})})$. For the extra edge between u_i and u'_i, which represents no selected path occupies T_{z_i}, we set its weight to $f(T_{z_i})$. It now easily follows that $f(T_v) = \textsc{Opt}(\mathcal{M}(T_v))$, which can be solved in polynomial time using algorithms for Maximum Weighted Matching [26]. ◀

Combining Lemmas 12, 14, and 16, we conclude with the theorem.

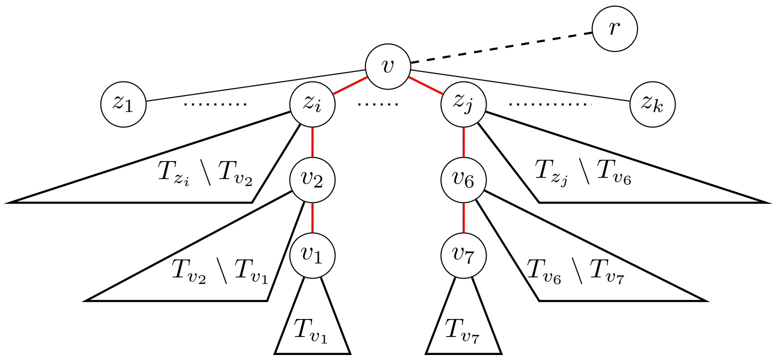

r
v
z_1
z_i
z_j
z_k
$T_{z_i} \setminus T_{v_2}$
v_2
v_6
$T_{z_j} \setminus T_{v_6}$
$T_{v_2} \setminus T_{v_1}$
v_1
v_7
$T_{v_6} \setminus T_{v_7}$
T_{v_1}
T_{v_7}

Figure 3 Illustration of subproblems: consider the red path $(v_1, v_2, v_3 = z_i, v_4 = v, v_5 = z_j, v_6, v_7)$ through v. If we select the red path, the subtrees T_{z_i} and T_{z_j} break into $T_{v_1}, T_{(v_1,v_2)} = T_{v_2} \setminus T_{v_1}, T_{(z_i,v_2)} = T_{z_i} \setminus T_{v_2}$ and $T_{(z_j,v_6)} = T_{z_j} \setminus T_{v_6}, T_{(v_6,v_7)} = T_{v_6} \setminus T_{v_7}, T_{v_7}$, respectively. They define independent subproblems and their optimal solutions have been computed before we compute $f(T_v)$.

▶ **Theorem 17.** *There is a polynomial-time exact algorithm for* $(2, 2)$*-Global-Connectivity Preservation.*

4 Approximation Algorithms for large p or q

In this section we provide approximation algorithms and hardness results for large p and q.

4.1 Approximation for the cases with $p = 1$

We present a primal-dual algorithm for $(1, q)$-Steiner-Connectivity Preservation. Consider the corresponding linear programming relaxation of (CutIP) and its dual:

$$\begin{array}{llll} \min & \sum_{e\in E} c_e x_e & & \\ \text{s.t.} & \sum_{e\in\delta(S)} x_e \geq 1 & \forall S \in \mathcal{S} \\ & x_e \geq 0 & \forall e \in E \end{array} \qquad \begin{array}{llll} \max & \sum_{S\in\mathcal{S}} y_S & & \\ \text{s.t.} & \sum_{S:S\in\mathcal{S}, e\in\delta(S)} y_S \leq c_e & \forall e \in E \\ & y_S \geq 0 & \forall S \in \mathcal{S} \end{array}$$

Algorithm 3. We start from a dual solution $\{y_S = 0 \mid S \in \mathcal{S}\}$ and maintain a partial solution $X \subseteq E$ which is the current protected edge set. At the beginning, $X := \emptyset$. We increase the dual variables iteratively and add edges to X whose corresponding dual constraints $\sum_{S:S\in\mathcal{S},e\in\delta(S)} y_S \leq c_e$ become tight. In each iteration, we pick some $S \in \mathcal{S}$ with $\delta(S) \cap X = \emptyset$ and increase y_S. Such a vertex set S can be found by enumerating terminal pairs (s, t) and checking whether there is an s-t-cut of value less than $q + 1$ with respect to the following capacity function: set the capacity of e to $q + 1$ if $e \in X$ and to 1, otherwise. We increase y_S until for some edge $e \in \delta(S)$, the dual constraint $\sum_{S:S\in\mathcal{S},e\in\delta(S)} y_S \leq c_e$ is tight. Then we add e to X and move to the next iteration until X is a feasible solution, which is the case if any terminal-separating cut has a capacity of at least $q + 1$ w.r.t. the above capacity function.

To bound the cost of X, we have

$$\sum_{e\in X} c_e = \sum_{e\in X} \sum_{S:S\in\mathcal{S},e\in\delta(S)} y_S = \sum_{S\in\mathcal{S}} y_S|\delta(S)\cap X| \le \sum_{S\in\mathcal{S}} y_S|\delta(S)| \le q\sum_{S\in\mathcal{S}} y_S \le q\cdot \textsc{Opt}.$$

▶ **Theorem 18.** *Algorithm 3 is a polynomial-time q-approximation algorithm for $(1,q)$-Steiner-Connectivity Preservation.*

The *global* connectivity variant $(1,q)$-GCP has more symmetry since we do not need to distinguish whether an edge cut is terminal-separating. By exploiting the special structure of the family $\mathcal{S} = \{S \subset V \mid |\delta(S)| \le p+q-1\}$, Bansal et al. [8] obtained a primal-dual 16-approximation algorithm for the *Augmenting Small Cuts* problem, which generalizes $(1,q)$-GCP. Recently, the factor has been reduced to 10 [37] and 5 [6] via refined analysis.

▶ **Theorem 19** (follows from [6, 8]). *There is a polynomial-time 5-approximation algorithm for $(1,q)$-Global-Connectivity Preservation.*

Finally, we consider $(1,q)$-s-t-Connectivity Preservation. We show that this problem is equivalent to the undirected *Minimum Shared Edge* problem: We are given a graph with edge weights and two specified vertices s,t. The task is to find k s-t paths with the minimum total weight of shared edges. Here, an edge is shared if it is contained in at least 2 paths.

▶ **Proposition 20.** *An edge set X is a feasible solution to $(1,q)$-stCP if and only if there are $(q+1)$ s-t-paths such that any edge shared by at least two paths belongs to X.*

Proof. To show necessity, we construct a graph $G=(V,E)$ with a capacity function u on the edges, where the capacity $u(e)$ of any edge e is $q+1$ if $e\in X$, and 1 otherwise. Since X is a feasible solution, by Proposition 5 the capacity of any s-t cut is at least $q+1$. Thus, there exist $q+1$ s-t-paths such that edges shared by at least two paths belong to X.

As for the sufficiency, suppose we have $q+1$ s-t-paths that only share edges in X. We claim that the shared edges form a feasible solution of $(1,q)$-stCP. For each cut of size at most q, at least one edge must be shared by two paths and this edge is in X. Thus, every cut $\delta(S)$ with $|\delta(S)|\le q$ satisfies $\delta(S)\cap X\ne\emptyset$. By Proposition 5, X is a feasible solution. ◀

▶ **Lemma 21.** *An edge set X is an inclusion-wise minimal solution to $(1,q)$-s-t-Connectivity Preservation if and only if there are $(q+1)$ s-t-paths such that the shared edges are exactly X.*

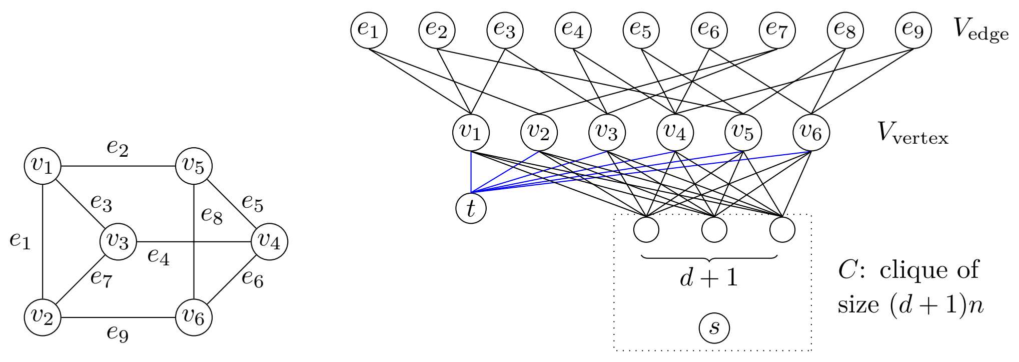
We conclude that $(1,q)$-stCP is equivalent to the Minimum Shared Edge Problem. Hence, Lemma 21 and the results of [5, 23, 38] imply the following.

▶ **Theorem 22.** *When parameterized by q, $(1,q)$-stCP admits an FPT algorithm for undirected graphs and an XP algorithm for directed graphs. Furthermore, $(1,q)$-stCP on directed graphs admits no $\mathcal{O}(2^{\log^{1-\epsilon}\max\{q,n\}})$-approximation, unless $\mathsf{NP}\subseteq\mathsf{DTIME}(n^{polylog(n)})$.*

4.2 Extension for larger p

Before presenting algorithms for more general cases, we argue that (p,q)-SCP is quite hopeless when both p and q are part of the input. Indeed, if this is the case, there is no polynomial-time algorithm that verifies feasibility of any given solution unless P=NP.

By Proposition 5, a given protected edge set X is infeasible if and only if there is a terminal-separating cut $\delta(S)$ such that $|\delta(S)|\le p+q-1$ and $|\delta(S)\cap X|\le p-1$. We define and study the complexity of the following (A,B)-bicriteria s-t-cut problem: Given an undirected graph with two specified vertices s,t and a subset of edges protected, decide



(a) A 3-Clique instance, $d = 3$. **(b)** The constructed (A, B)-bicriteria s-t-cut instance.

Figure 4 The reduction: the blue edges are protected edges and the black edges are unprotected.

whether there is an s-t-cut such that the number of protected edges in the cut is at most A $(= p - 1)$ and the total number of edges in the cut is at most B $(= p + q - 1)$ and. Recall that in the (p, q)-s-t-Flexible Network Design problem, verifying the feasibility of a solution can be formulated as follows. Given an undirected graph with safe and unsafe edges, decide whether there are p edge-disjoint paths between s and t after at most q failures of unsafe edges. Hence verifying the feasibility is equivalent to the (A, B)-bicriteria s-t-cut problem. We show that the (A, B)-bicriteria s-t-cut problem is NP-complete, which implies that there is no polynomial-time approximation algorithm for (p, q)-s-t-Connectivity Preservation or (p, q)-s-t-Flexible Network Design, unless P = NP.

▶ **Theorem 2.** *When both p and q are part of the input, verifying the feasibility of a solution to (p, q)-s-t-Connectivity Preservation or (p, q)-s-t-Flexible Network Design is* NP*-complete, even in perfect graphs. Hence, they do not admit an α-approximation for any α unless* P = NP.

Proof. We use a reduction similar to [24] from k-Clique on d-regular graphs, which is NP-complete [27]. See Figure 4 for an illustration. In k-Clique, we are given an undirected graph and we need to decide whether there is a clique of size k. Given an instance of k-clique with graph $G = (V, E)$, where G is d-regular, we construct an instance of the (A, B)-bicriteria s-t-cut problem (G', s, t, A, B) as follows. Let $n := |V|, m := |E|$. We create n vertices V_{vertex} corresponding to V and m vertices V_{edge} corresponding to E. In the following, when we say we connect two vertices, then they are connected by an *unprotected* edge by default. For each $e = (u, v) \in E$, we connect its corresponding vertex to the two vertices corresponding to u and v. Then we create a vertex t and connect it to each vertex in V_{vertex} with *protected* edges. Create an auxiliary clique C of size $n(d + 1)$ and fix an arbitrary vertex in the clique as s. Fix $d + 1$ vertices in the clique other than s and fully connect them to each vertex in V_{vertex}, which results in a complete bipartite subgraph $K_{n,d+1}$. Let $A := k, B := (d + 1)n - k(k - 1)$. We claim that G has a clique of size k if and only if the protected edges form an *infeasible* solution to the instance of the (A, B)-bicriteria s-t-cut problem.

For the first direction, suppose G has a clique $CL = (V_{\text{CL}}, E_{\text{CL}})$ of size k. Let S include all the vertices in V_{vertex} and V_{edge} corresponding to V_{CL}, E_{CL}, and the auxiliary clique C. We show $\delta(S)$ defines an (A, B)-bicriteria s-t-cut. In $\delta(S)$, the only protected edges are those edges between t and the vertices corresponding to V_{CL}. Hence there are exactly $k = A$ protected edges. As for $|\delta(S)|$, consider the edges between V_{vertex} and $t \cup C$. Each vertex in $V_{\text{vertex}} \setminus S$ contributes $d+1$ and each vertex in $V_{\text{vertex}} \cap S$ contributes 1. Now consider the edges between V_{edge} and V_{vertex}. There are $d \cdot k$ edges incident to $V_{\text{vertex}} \cap S$, among which $k(k - 1)$ do not contribute to $|\delta(S)|$. Hence, $|\delta(S)| = (d+1)(n-k)+k+dk-k(k-1) = (d+1)n-k(k-1) = B$ and $\delta(S)$ is an (A, B)-bicriteria s-t-cut.

For the other direction, suppose there is an (A, B)-bicriteria s-t-cut $\delta(S)$ with $|\delta(S)| \leq B$ such that $\delta(S)$ contains at most A protected edges. We show that $S \cap V_{\text{vertex}}$ corresponds to the vertices of a clique of size k in G and $S \cap V_{\text{edge}}$ contains the edges of this clique. Observe that $|S \cap V_{\text{vertex}}| \leq k$ since there are at most $A = k$ protected edges in $\delta(S)$. We will show that $|S \cap V_{\text{vertex}}| = k$ and $|S \cap V_{\text{edge}}| \geq k(k-1)/2$. Furthermore, these $k(k-1)/2$ vertices in $S \cap V_{\text{edge}}$ have both their neighbors in $S \cap V_{\text{vertex}}$. Hence $S \cap V_{\text{vertex}}$ defines a clique of size k in G.

Observe that S must include the whole auxiliary clique C, otherwise $|\delta(S)|$ would exceed B. Let $S' = S \setminus V_{\text{edge}}$ and note that $C \subseteq S'$. We prove that $|\delta(S')| = (d+1)n > B$ by considering the following process. Starting from $Y := C$, we add the vertices in $S' \setminus C$ one by one to Y. During the process, $|\delta(Y)|$ does not change since each vertex in $S' \setminus C$ is connected to exactly $d+1$ vertices in C, d vertices in V_{edge} and t. Hence $|\delta(S')| = |\delta(C)| = (d+1)n$. Now starting from $Y = S'$, we add the vertices in $S \setminus S'$ one by one to Y. During the process, the only case that adding a vertex decreases $|\delta(Y)|$ (by 2) is when both its neighbors are in $S \cap V_{\text{vertex}}$. Therefore, we have at least $k(k-1)/2$ vertices in $S \setminus S'$, each having both their neighbors in $S \cap V_{\text{vertex}}$, since $|\delta(S')| - |\delta(S)| \geq (d+1)n - B = k(k-1)$. Hence, $|S \cap V_{\text{vertex}}| \geq k$, and with the above inequality of $|S \cap V_{\text{vertex}}| \leq k$ we have $|S \cap V_{\text{vertex}}| = k$. Further, for any two vertices in $S \cap V_{\text{vertex}}$, there is some vertex in $S \cap V_{\text{edge}}$ connected to both of them, implying that $S \cap V_{\text{vertex}}$ corresponds to the vertices of a clique in G. Hence, G contains a clique of size k. ◀

On the positive side, if q is a constant, we can enumerate those edge sets F with $|F| \leq q$ such that some terminal pair in $(V, E \setminus F)$ is not p-edge-connected. For each of those sets, we need to protect at least one edge in the set, which reduces to the hitting set problem and admits a q-approximation where q is the largest size of the sets to be hit [10, 31].

In the following, we extend algorithm for $(1, q)$-SCP to (p, q)-SCP with p being a constant. The idea is to start from an empty solution and augment the current solution by iteratively increasing the number of protected edges in each critical cut. Our algorithm consists of p phases. In phase i, we are given a partial solution X_{i-1} satisfying that each critical cut contains at least $i-1$ edges in X_{i-1}. We then (approximately) solve the following augmentation problem $\mathcal{P}_i$: Add to X_{i-1} a minimum-cost set of edges $Y_i \subseteq E \setminus X_{i-1}$ such that $X_i := X_{i-1} \cup Y_i$ includes at least i edges of each critical cut. That is, find a set Y_i that includes at least one edge from each critical cut with exactly $i-1$ protected edges in X_{i-1}. The augmentation problem is solved similarly to the primal-dual framework for $(1, q)$-SCP.

Formally, let $\mathcal{S}_0 = \mathcal{S}, X_0 = \emptyset$. In phase i with $1 \leq i \leq p$, we define $\mathcal{S}_i = \{S \in \mathcal{S} \mid |\delta(S) \cap X_{i-1}| = i-1\}$, i.e., the critical cuts with exactly $i-1$ protected edges. Next, we solve the following problem $\mathcal{P}_i$: find a minimum-cost edge set $Y_i \subseteq E \setminus X_{i-1}$ such that $Y_i \cap \delta(S) \neq \emptyset$ for any $S \in \mathcal{S}_i$. Then we set $X_i := X_{i-1} \cup Y_i$ and go on to the next phase. To solve $\mathcal{P}_i$, we use a primal-dual algorithm based on the following LP to compute a $(p+q-1)$-approximation solution to $\mathcal{P}_i$ which is essentially the same as $(1, q)$-Steiner-Connectivity Preservation. The approximation ratio is bounded by $p+q-1$ as the size of a critical cut is at most $p+q-1$.

$$\begin{array}{llll}
\min & \displaystyle\sum_{e \in E \setminus X_{i-1}} c_e x_e & & \\
\text{s.t.} & \displaystyle\sum_{e \in \delta(S) \setminus X_{i-1}} x_e \geq 1 & \forall S \in \mathcal{S}_i & \\
& x_e \geq 0 & \forall e \in E \setminus X_{i-1} &
\end{array}
\qquad
\begin{array}{llll}
\max & \displaystyle\sum_{S \in \mathcal{S}_i} y_S & & \\
\text{s.t.} & \displaystyle\sum_{S : S \in \mathcal{S}_i, e \in \delta(S)} y_S \leq c_e & \forall e \in E \setminus X_{i-1} & \\
& y_S \geq 0 & \forall S \in \mathcal{S}_i &
\end{array}$$

The only difference is the process of finding a violating set $S \in \mathcal{S}_i$ with respect to some partial solution X. However, finding such a violating set is non-trivial. We are only aware of a solution when p is a constant, which we present in the following lemma.

▶ **Lemma 23.** *Given an edge set $X \supseteq X_{i-1}$, there is a polynomial-time algorithm that computes a set $S \in \mathcal{S}_i$ such that $\delta(S) \cap X = \emptyset$ when p is a constant.*

Proof. Since $X \supseteq X_{i-1}$, we have $|\delta(S) \cap X| \geq i-1$ for any $S \in \mathcal{S}$. It suffices to find some $S \in \mathcal{S}$ with $|\delta(S) \cap X| = i-1 \leq p$. To this end, we guess the edge set $X' = \delta(S) \cap X$. Note that $|X'| < p$. Further, for each edge $e = (u, v)$ in X', we guess whether $u \in S, v \notin S$ or $u \notin S, v \in S$. Thus the number of possibilities is at most $\binom{m}{p} \cdot 2^p$, which is polynomial when p is constant. For the edges in X', let the set of endpoints in S be A, and the other endpoints be B. It reduces to finding some $S \in \mathcal{S}$ with $A \subseteq S$, $B \notin S$ and $\delta(S) \cap X = X'$. This can be achieved by identifying the vertices in A and B by a new vertex v_A and v_B, respectively, contracting edges in $X \setminus X'$, and computing a minimum v_A-v_B cut in the resulting graph. If the cut has size less than $p + q$, then this cut belongs to $\mathcal{S}_i$ and has $i-1$ edges in X. ◀

To bound the total cost of the p phases of our algorithm, we use the following LP relaxation and its dual for the analysis. The constraints $x_e \leq 1$ cannot be omitted as we do for $p = 1$. Otherwise, an edge may be "protected" multiple times, which is not allowed.

$$\begin{array}{llll} \min & \sum_{e \in E} c_e x_e & & \\ \text{s.t.} & \sum_{e \in \delta(S)} x_e \geq p & \forall S \in \mathcal{S} \\ & 0 \leq x_e \leq 1 & \forall e \in E \end{array} \qquad \begin{array}{llll} \max & \sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e & \\ \text{s.t.} & \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e \leq c_e & \forall e \in E \\ & y_S, z_e \geq 0 & \forall S \in \mathcal{S}, \forall e \in E \end{array}$$

In the following lemma, we compare the optimal cost of the augmentation problem $\mathcal{P}_i$ to the optimal cost of (p, q)-Steiner-Connectivity Preservation.

▶ **Lemma 24.** *Given a feasible dual solution $y^{(i)}$ of $\mathcal{P}_i$, we can construct a feasible dual solution y of (p, q)-Steiner-Connectivity Preservation such that*

$$\sum_{S \in \mathcal{S}_i} y_S^{(i)} \leq \frac{1}{p-i+1} \Big(\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e \Big) .$$

Proof. Let $y_S = y_S^{(i)}$ for $S \in \mathcal{S}_i$ and $y_S = 0$ for $S \in \mathcal{S} \setminus \mathcal{S}_i$. Let $z_e = 0$ for $e \in E \setminus X_{i-1}$ and $z_e = \sum_{S: S \in \mathcal{S}_i, e \in \delta(S)} y_S^{(i)}$ for $e \in X_{i-1}$. We claim that (y, z) forms a feasible dual solution. For any $e \in X_{i-1}$, $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e = 0$ by definition. For $e \in E \setminus X_{i-1}$, $\sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S - z_e = \sum_{S: S \in \mathcal{S}, e \in \delta(S)} y_S^{(i)} \leq c_e$. Next, we compare the dual objective values of y and $y^{(i)}$. We have

$$\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e = \sum_{S \in \mathcal{S}_i} p \cdot y_S^{(i)} - \sum_{e \in X_{i-1}} \sum_{S: S \in \mathcal{S}_i, e \in \delta(S)} y_S^{(i)} = \sum_{S \in \mathcal{S}_i} y_S^{(i)} (p - |\delta(S) \cap X_{i-1}|).$$

By definition of $\mathcal{S}_i$, for any $S \in \mathcal{S}_i$, we have $|\delta(S) \cap X_{i-1}| = i-1$. Thus, we conclude:

$$\sum_{S \in \mathcal{S}} p \cdot y_S - \sum_{e \in E} z_e = (p-i+1) \sum_{S \in \mathcal{S}_i} y_S^{(i)}.$$ ◀

▶ **Theorem 3.** *There is a polynomial-time $\mathcal{O}((p+q)\log p)$-approximation algorithm for (p, q)-Steiner-Connectivity Preservation when p is a constant.*

Proof. The algorithm consists of p phases. In phase i, we apply Lemma 23 and the primal-dual framework for $(1, q)$-SCP to find a $(p+q-1)$-approximation solution for the augmentation problem $\mathcal{P}_i$. By Lemma 24, the cost of Y_i in phase i is at most $(p+q-1)\cdot\textsc{Opt}(\mathcal{P}_i) \leq \frac{p+q-1}{p-i+1}\textsc{Opt}$. Thus the total cost is at most $\sum_{i=1}^{p} c(Y_i) \leq \sum_{i=1}^{p} \frac{p+q-1}{p-i+1}\textsc{Opt} \leq H_p \cdot (p+q-1)\cdot\textsc{Opt}$, where H_p is the p-th harmonic number. Using $H_p \leq \log(p) + 1$, we obtain the theorem. ◀

For (p, q)-Global-Connectivity Preservation, we can approximately solve the augmentation problem without requiring p to be a constant. Indeed, we reduce finding the critical cuts to finding certain 2-approximate minimum cuts in G, where each edge e has a capacity of $\frac{p+q}{i-1}$ if $e \in X_{i-1}$ and 1 otherwise. These cuts can be enumerated in polynomial time [33, 36].

Algorithm 4. In phase 1, we apply the 5-approximation algorithm from [6]. That is, we compute X_1 such that for any $S \in \mathcal{S}_0 = \mathcal{S}$, $X_1 \cap \delta(S) \neq \emptyset$. For phase i with $2 \leq i \leq p$, we approximately solve the augmentation problem $\mathcal{P}_i$ by reducing it to Set Cover. Here, we view a set $S \in \mathcal{S}_i$ as an element in the Set Cover instance and view an edge $e \in E \setminus X_{i-1}$ as a set in the Set Cover instance. We use either the $\mathcal{O}(\log N)$-approximation [19] where N is the number of elements to be covered, or the f-approximation [10, 31] where f is the maximum number of sets in which an element is contained. Note that applying Lemma 24 requires a dual feasible solution, which is fortunately a byproduct of these Set Cover algorithms.

▶ **Theorem 4.** *There is a polynomial-time $\mathcal{O}(\log p\cdot\min\{p+q, \log n\})$-approximation algorithm for (p, q)-Global-Connectivity Preservation.*

Proof. The cost of phase 1 is no more than $5 \cdot \textsc{Opt}$. For phase i with $2 \leq i \leq p$, we apply Set Cover algorithms explicitly. We show that the number of elements to be covered is $|\mathcal{S}_i| = \mathcal{O}(|V|^4)$ and we can construct the Set Cover instance in polynomial time. To this end, we assign different capacities to edges in X_{i-1} and other edges such that for any $S \in \mathcal{S}_i$, $\delta(S)$ is a 2-approximate minimum cut with respect to the capacity function. By Karger's bound [33], the number of 2-approximate minimum cuts is $\mathcal{O}(|V|^4)$ and we can enumerate them in polynomial time [36]. Formally, let the capacity of edges in X_{i-1} be $\frac{p+q}{i-1}$ and the capacity of edges in $E \setminus X_{i-1}$ be 1. Given any cut C, the capacity of C is at least $p + q$ since it either contains at least $p + q$ edges or contains at least $i - 1$ edges in X_{i-1}. For any $S \in \mathcal{S}_i$, the capacity of $\delta(S)$ is at most $(i-1)\frac{p+q}{i-1} + p + q - 1 < 2(p + q)$. Thus $\delta(S)$ defines a 2-approximate minimum cut and we can find all the sets in $\mathcal{S}_i$ in polynomial time.

Further, in the constructed Set Cover instance, an element is contained in at most $p + q - 1$ sets since $|\delta(S)| \leq p + q - 1$ for any $S \in \mathcal{S}_i$. Thus, we can compute an augmenting edge set $X_i \setminus X_{i-1}$ whose cost is $\mathcal{O}(\min\{\log n, p + q\} \cdot \sum_{S\in\mathcal{S}_i} y_S^{(i)})$ where $y^{(i)}$ is the dual feasible solution of $\mathcal{P}_i$. Combining it with Lemma 24, we conclude that the algorithm is an $\mathcal{O}(\log p \cdot \min\{\log n, p + q\})$-approximation. ◀

5 Conclusion

We examine Connectivity Preservation from two perspectives. For small values of p and q, we focus on polynomial-time exact algorithms. For large values of p and q, we show hardness and devise approximation algorithms. Nonetheless, there remain some gaps between cases solvable in polynomial time and NP-hard ones. In particular, it remains open whether $(1, q)$-GCP admits any polynomial-time exact algorithm for constant $q \geq 3$. Another interesting problem is $(1, q)$-GCP with q being the capacity of the minimum cuts, i.e., finding a minimum-cost edge set that intersects with all the minimum cuts. Note that for the s-t-connectivity variant, this can be tackled via Min-cost Flow techniques.

References

1 Waseem Abbas, Aron Laszka, Yevgeniy Vorobeychik, and Xenofon Koutsoukos. Improving network connectivity using trusted nodes and edges. In *2017 American Control Conference (ACC)*, pages 328–333. IEEE, 2017. doi:10.23919/ACC.2017.7962974.

2 David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity. *Math. Program.*, 192(1):409–441, 2022. doi:10.1007/S10107-021-01664-9.

3 David Adjiashvili, Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. Fault-tolerant edge-disjoint s-t paths - beyond uniform faults. In *SWAT*, volume 227 of *LIPIcs*, pages 5:1–5:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.SWAT.2022.5.

4 David Adjiashvili, Sebastian Stiller, and Rico Zenklusen. Bulk-robust combinatorial optimization. *Math. Program.*, 149(1-2):361–390, 2015. doi:10.1007/S10107-014-0760-6.

5 Sepehr Assadi, Ehsan Emamjomeh-Zadeh, Ashkan Norouzi-Fard, Sadra Yazdanbod, and Hamid Zarrabi-Zadeh. The minimum vulnerability problem. *Algorithmica*, 70(4):718–731, 2014. doi:10.1007/S00453-014-9927-Z.

6 Ishan Bansal. A global analysis of the primal-dual method for pliable families, 2024. arXiv:2308.15714.

7 Ishan Bansal, Joe Cheriyan, Logan Grout, and Sharat Ibrahimpur. Algorithms for 2-connected network design and flexible steiner trees with a constant number of terminals. In *APPROX/RANDOM*, volume 275 of *LIPIcs*, pages 14:1–14:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.APPROX/RANDOM.2023.14.

8 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions. *Algorithmica*, 86(8):2575–2604, 2024. doi:10.1007/S00453-024-01235-2.

9 Ishan Bansal, Joseph Cheriyan, Sanjeev Khanna, and Miles Simmons. Improved approximation algorithms for flexible graph connectivity and capacitated network design, 2024. doi:10.48550/arXiv.2411.18809.

10 Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *J. Algorithms*, 2(2):198–203, 1981. doi:10.1016/0196-6774(81)90020-1.

11 Marshall W. Bern and Paul E. Plassmann. The steiner problem with edge lengths 1 and 2. *Inf. Process. Lett.*, 32(4):171–176, 1989. doi:10.1016/0020-0190(89)90039-2.

12 Daniel Bienstock and Nicole Diaz. Blocking small cuts in a network, and related problems. *SIAM J. Comput.*, 22(3):482–499, 1993. doi:10.1137/0222034.

13 Sylvia C. Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Math. Program.*, 204(1):493–516, 2024. doi:10.1007/S10107-023-01961-5.

14 Federica Cecchetto, Vera Traub, and Rico Zenklusen. Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *STOC*, pages 370–383. ACM, 2021. doi:10.1145/3406325.3451086.

15 Ruoxu Cen, Jason Li, and Debmalya Panigrahi. Edge connectivity augmentation in near-linear time. In *STOC*, pages 137–150. ACM, 2022. doi:10.1145/3519935.3520038.

16 Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. *Algorithmica*, 72(2):493–514, 2015. doi:10.1007/S00453-013-9862-4.

17 Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

18 Chandra Chekuri and Rhea Jain. Approximation algorithms for network design in non-uniform fault models, 2024. arXiv:2403.15547, doi:10.48550/arXiv.2403.15547.

19 Vasek Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

20 Efim A. Dinits, Alexander V. Karzanov, and Micael V. Lomonosov. On the structure of a family of minimum weighted cuts in a graph. *Studies in Discrete Optimization*, pages 209–306, 1976.

21 Gabriel L Duarte, Hiroshi Eto, Tesshu Hanaka, Yasuaki Kobayashi, Yusuke Kobayashi, Daniel Lokshtanov, Lehilton LC Pedrosa, Rafael CS Schouery, and Uéverton S Souza. Computing the largest bond and the maximum connected cut of a graph. *Algorithmica*, 83:1421–1458, 2021. doi:10.1007/S00453-020-00789-1.

22 Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976. doi:10.1137/0205044.

23 Till Fluschnik, Stefan Kratsch, Rolf Niedermeier, and Manuel Sorge. The parameterized complexity of the minimum shared edges problem. *J. Comput. Syst. Sci.*, 106:23–48, 2019. doi:10.1016/J.JCSS.2018.12.002.

24 Fedor V. Fomin, Petr A. Golovach, and Janne H. Korhonen. On the parameterized complexity of cutting a few vertices from a graph. In *MFCS*, volume 8087 of *Lecture Notes in Computer Science*, pages 421–432. Springer, 2013. doi:10.1007/978-3-642-40313-2_38.

25 Greg N. Frederickson and Joseph F. JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981. doi:10.1137/0210019.

26 Harold N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *SODA*, pages 434–443. SIAM, 1990. URL: http://dl.acm.org/citation.cfm?id=320176.320229.

27 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

28 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997. doi:10.1007/BF02523685.

29 Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232. ACM/SIAM, 1994. URL: http://dl.acm.org/citation.cfm?id=314464.314497.

30 Zhongtian He, Shang-En Huang, and Thatchaphol Saranurak. Cactus representation of minimum cuts: Derandomize and speed up. In *SODA*, pages 1503–1541. SIAM, 2024. doi:10.1137/1.9781611977912.61.

31 Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM J. Comput.*, 11(3):555–556, 1982. doi:10.1137/0211045.

32 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Comb.*, 21(1):39–60, 2001. doi:10.1007/S004930170004.

33 David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313605.

34 Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004. doi:10.1137/S0097539702416736.

35 Hiroshi Nagamochi and Toshihide Ibaraki. *Algorithmic Aspects of Graph Connectivity*, volume 123 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 2008. doi:10.1017/CBO9780511721649.

36 Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing all small cuts in an undirected network. *SIAM J. Discret. Math.*, 10(3):469–481, 1997. doi:10.1137/S0895480194271323.

37 Zeev Nutov. Improved approximation ratio for covering pliable set families, 2024. arXiv:2404.00683, doi:10.48550/arXiv.2404.00683.

38 Masoud T. Omran, Jörg-Rüdiger Sack, and Hamid Zarrabi-Zadeh. Finding paths with minimum shared edges. *J. Comb. Optim.*, 26(4):709–722, 2013. doi:10.1007/S10878-012-9462-2.

39 David Pritchard. k-edge-connectivity: Approximation and LP relaxation. In *WAOA*, volume 6534 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 2010. doi:10.1007/978-3-642-18318-8_20.

40 Vera Traub and Rico Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *FOCS*, pages 1–12. IEEE, 2021. doi:10.1109/FOCS52979.2021.00010.

41 Vera Traub and Rico Zenklusen. Local search for weighted tree augmentation and steiner tree. In *SODA*, pages 3253–3272. SIAM, 2022. doi:10.1137/1.9781611977073.128.

42 Vera Traub and Rico Zenklusen. A $(1.5+\epsilon)$-approximation algorithm for weighted connectivity augmentation. In *STOC*, pages 1820–1833. ACM, 2023. doi:10.1145/3564246.3585122.

43 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Comb.*, 15(3):435–454, 1995. doi:10.1007/BF01299747.

44 Jianan Zhang, Eytan Modiano, and David Hay. Enhancing network robustness via shielding. *IEEE/ACM Transactions on Networking*, 25(4):2209–2222, 2017. doi:10.1109/TNET.2017.2689019.