Improved Approximations for Relative Survivable Network Design

Michael Dinitz[1], Ama Koranteng[1(✉)], Guy Kortsarz[2], and Zeev Nutov[3]

[1] Johns Hopkins University, Baltimore, USA
mdinitz@cs.jhu.edu, akorant1@jhu.edu
[2] Rutgers University, Camden, USA
guyk@camden.rutgers.edu
[3] The Open University, Ra'anana, Israel
nutov@openu.ac.il

Abstract. One of the most important and well-studied settings for network design is edge-connectivity requirements. This encompasses uniform demands (e.g. the Minimum k-Edge-Connected Spanning Subgraph problem), as well as nonuniform demands (e.g. the Survivable Network Design problem (SND)). In a recent paper [Dinitz, Koranteng, Kortsarz APPROX '22], the authors observed that a weakness of these formulations is that we cannot consider fault-tolerance in graphs that have small cuts but where *some* large fault sets can still be accommodated. To remedy this, they introduced new variants of these problems under the notion *relative* fault-tolerance. Informally, this requires not that two nodes are connected if there are a bounded number of faults (as in the classical setting), but that they are connected if there are a bounded number of faults *and the nodes are connected in the underlying graph post-faults.*

Due to difficulties introduced by this new notion of fault-tolerance, the results in [Dinitz, Koranteng, Kortsarz APPROX '22] are quite limited. For the Relative Survivable Network Design problem (RSND) with non-uniform demands, they are only able to give a nontrivial result when there is a single demand with connectivity requirement 3—a non-optimal 27/4-approximation. We strengthen this result in two significant ways: We give a 2-approximation for RSND when *all requirements are at most* 3, and a $2^{O(k^2)}$-approximation for RSND with a single demand of *arbitrary value* k. To achieve these results, we first use the "cactus representation" of minimum cuts to give a lossless reduction to normal SND. Second, we extend the techniques of [Dinitz, Koranteng, Kortsarz APPROX'22] to prove a generalized and more complex version of their structure theorem, which we then use to design a recursive approximation algorithm.

Keywords: Fault tolerance · Network design

See [17] for the full version.

M. Dinitz—Supported in part by NSF awards CCF-1909111 and CCF-2228995.
A. Koranteng—Supported in part by an NSF Graduate Research Fellowship and NSF award CCF-1909111.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
J. Byrka and A. Wiese (Eds.): WAOA 2023, LNCS 14297, pp. 190–204, 2023.
https://doi.org/10.1007/978-3-031-49815-2_14

1 Introduction

Fault-tolerance has been a central object of study in approximation algorithms, particularly for network design problems where the graphs we study represent physical objects which might fail (communication links, transportation links, etc.). In these settings it is natural to ask for whatever object we build to be fault-tolerant. The precise definition of "fault-tolerance" varies in different settings, but a common formulation is edge fault-tolerance, which typically takes the form of edge connectivity. Informally, these look like guarantees of the form "if up to k edges fail, then the nodes I want to be connected are still connected." For example, consider the following classical fault-tolerance problem.

Definition 1. *In the Survivable Network Design problem (SND, sometimes referred to as Generalized Steiner Network) we are given an edge-weighted graph G and demands $\{(s_i, t_i, k_i)\}_{i \in [\ell]}$, and we are supposed to find the minimum-weight subgraph H of G so that there are at least k_i edge-disjoint paths between s_i and t_i for every $i \in [\ell]$. In other words, for every $i \in [\ell]$, if fewer than k_i edges fail then s_i and t_i will still be connected in H even after failures.*

The Survivable Network Design problem is well-studied (see [15,23,25,30] for a sample); notably, Jain [25] gives a 2-approximation algorithm for the problem in a seminal paper. Beyond SND, edge fault-tolerance has been studied in many related network design contexts, with the k-Edge Connected Spanning Subgraph, Fault-Tolerant Group Steiner Tree, Fault-Tolerant Spanner, and Fault-Tolerant Shortest Paths problems being just a few examples (see [4,15,19,26]). These and other classical fault-tolerance problems, including the Survivable Network Design problem, are *absolute* fault-tolerance problems—if up to k objects fail, the remaining graph should function as desired. This differs from the stronger notion of fault-tolerance introduced in [16], called *relative* fault-tolerance. Relative fault-tolerance makes guarantees that rather than being absolute ("if at most k edges fail the network still functions") are relative to an underlying graph or system ("if at most k edges fail, the subgraph functions just as well as the original graph post-failures").

Relative fault-tolerance is therefore a natural generalization of absolute fault-tolerance: If the input graph has the desired connectivity, then the relative fault-tolerance and absolute fault-tolerance definitions are equivalent. However, if the input graph does not have the requested connectivity, then relative fault-tolerance allows us to return a solution with interesting and nontrivial guarantees while absolute fault-tolerance forces us to return nothing. In this way, relative fault-tolerance overcomes a significant weakness of absolute fault-tolerance.

This relative fault-tolerance definition was inspired by a recent line of work on relative notions of fault-tolerance for graph spanners and emulators [5–9,11,18,19]. In these settings, the goal is generally to find existential bounds and algorithms to achieve them, rather than to do optimization. In [16], by contrast, their approach takes the point of view of optimization and approximation algorithms. With this notion of fault-tolerance in network design, the authors of [16] define the relative version of the Survivable Network Design problem.

Definition 2. *In the Relative Survivable Network Design problem (RSND), we are given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$ and demands $\{(s_i, t_i, k_i)\}_{i \in [\ell]}$. A feasible solution is a subgraph H of G where for all $i \in [\ell]$ and $F \subseteq E$ with $|F| < k_i$, if there is a path in $G \setminus F$ from s_i to t_i then there is also a path in $H \setminus F$ from s_i to t_i. Our goal is to find the minimum weight feasible solution.*

Note that if s_i and t_i are k_i-connected in G for every $i \in [\ell]$, then RSND is exactly the same as SND. If in G there exists some $i \in [\ell]$ such that s_i and t_i are *not* k_i-connected, then although there is no solution for SND, there is a meaningful RSND solution.

There has been recent work on a related network design model introduced by Adjiashvili [1–3,10,13]. In this model, E is partitioned into "safe" and "unsafe" edges. Informally, in the Flex-SNDP problem we are given a graph $G = (V, E)$ with edge costs and with functions $p, q : V \times V \to \mathbb{Z}^+$. We must return a min cost subgraph such that for each vertex pair u, v, they are $p(u, v)$-connected after deleting any subset of up to $q(u, v)$ unsafe edges. Like RSND, Flex-SNDP is a natural generalization of SND. However, it is an absolute fault-tolerance problem since it does not consider the underlying connectivity of the input. No polynomial-time approximation algorithms are known for general Flex-SNDP, though there has been recent work on several special cases [3,10,12,13].

The Results of [16]. Although relative fault-tolerance is a natural and promising generalization of fault-tolerance, the results given in [16] for the RSND problem are quite limited. Outside of a 2-approximation algorithm for the special case in which all demands are identical, [16] is only able to give algorithms for some of the simplest RSND special cases. First, they give an extremely simple 2-approximation for the RSND special case where all demands are in $\{0, 1, 2\}$ (also known as the 2-RSND problem). The algorithm falls out of the observation that there is only a difference between a relative demand of 2 and an absolute demand of 2 when there is a cut of size one separating the vertex demand pair. Cuts of size one are very easy to handle, allowing for a simple and straightforward reduction to SND.

Cuts of size two or larger are significantly more difficult to reason about, and so the 2-RSND algorithm does not extend to larger demands. As a result of this more complex cut structure, [16] is only able to handle demands of value 3 (and reason about the size two cuts between them) when there is only *a single demand*, with value 3 (also known as the SD-3-RSND problem). Despite this being an extremely restricted special case of RSND, the algorithm and analysis given by [16] are quite complex, depending on a careful graph decomposition involving "important separators" (a concept from fixed-parameter tractability [28]). Moreover, this algorithm only achieved a 27/4-approximation for the problem, far from the 2-approximation (or even exact algorithm) that one might hope for.

The limited results of [16] show that while relative fault-tolerance is an attractive notion, applying it to the Survivable Network Design problem significantly changes the structure of the problem and makes it difficult to reason about and develop algorithms for. For example, while [16] only gives a 27/4-approximation

for SD-3-RSND, there is an exact polynomial-time algorithm for the SND equivalent (by a simple reduction to the Min-Cost Flow problem). So one might worry that relative fault-tolerance is simply too difficult of a definition, and the results of [16] are limited precisely because nothing is possible for even slightly more general settings.

1.1 Our Results and Techniques

In this paper, we seek to alleviate this worry by providing improved bounds for generalizations of the settings considered in [16]. In particular, we study two natural generalizations of the SD-3-RSND problem (which [16] provided a 27/4-approximation for). First, rather than only a single demand with value at most 3, can we handle an *arbitrary* number of demands that are at most 3? Secondly, in the single demand setting, instead of only handling a demand of at most 3, can we generalize to *arbitrary* values?

3-RSND. We begin with the setting where all demands are at most 3, but there can be an arbitrary number of such demands. We call this the 3-RSND problem. Note that, as discussed, there are no previous results for this setting, and the most related result is a 27/4-approximation if there is only a single such demand [16]. We prove the following theorem.

Theorem 1. *There is a polynomial-time 2-approximation for the 3-RSND problem.*

To obtain this theorem, we use entirely different techniques from those used in [16]. Most notably, we use the *cactus representation* of the global minimum cuts (which in this case are 2-cuts) of the input graph. The cactus representation of global minimum cuts is well studied and has been leveraged in a number of settings (see [20,21,24,27,29] for a sample). While it can be defined and constructed for more general connectivity values, for our setting we can construct the cactus representation by contracting components with certain connectivity properties. This results in a so-called *cactus graph*, which at a high level is a "tree of cycles": every pair of cycles intersects on at most one component in the construction. This cactus graph now has a simple enough structure that it allows us to reduce the original problem to a simpler problem in each of the contracted components. That is, we are able to show that certain parts of the cactus are essentially "forced", while other parts are not necessary, so the only question that remains is what to do "inside" of each cactus vertex, i.e., each component. This reduction makes the connectivity demands inside each component more complicated, but fortunately we are guaranteed 3-connectivity between special vertices inside the component. Hence we can use Jain's 2-approximation for SND [25] without worrying about the relative nature of the demands.

SD-k-RSND. Our second improvement is orthogonal: rather than allowing for more demands of at most 3, we still restrict ourselves to a single demand but

allow it to be a general constant k rather than 3. We call this the SD-k-RSND problem. As with the 3-RSND problem discussed earlier, there are no known results for this problem. We prove the following theorem:

Theorem 2. *There is a polynomial-time $2^{O(k^2)}$-approximation for the SD-k-RSND problem.*

To prove this, we extend the technique used by [16] for the $k = 3$ case. They construct a "chain" of 2-separators (cuts of size 2 that are also important separators) so that in each component in the chain, there are no 2-cuts between the incoming separator and the outgoing separator. They are then able to use this structure to characterize the connectivity requirement of any feasible solution restricted to that component. To extend this technique, we use important separators of size up to $k - 1$ to carefully construct a *hierarchy* of chains. The hierarchy has $k - 1$ levels of nested components, so that for each component in the ith level of the hierarchy, there are no cuts of size at most i between the incoming and outgoing separators. There are multiple ways of constructing such a hierarchy, but we prove that a particular construction yields a hierarchy with a number of useful but delicate properties within a single level and between different levels of the hierarchy. With these properties, we can characterize the complex connectivity requirement of any feasible solution when restricted to a component in the hierarchy. Once we have this structure theorem, we approximate the optimal solution in each component of the hierarchy via a recursive algorithm.

Simplification of k-EFTS. The k-Edge Fault Tolerant Subgraph problem (k-EFTS) is the special case of RSND where all demands are identical: every two nodes have a demand of exactly k. A 2-approximation for k-EFTS was recently given in [16] via a somewhat complex proof; in particular, they defined a new property called *local weak supermodularity* and used it to show that Jain's iterative rounding still gave the same bounds in the relative setting. In the full version [17], we give a simplification of this proof. It turns out that local weak supermodularity is not actually needed, and a more classical notion of $\mathcal{F}$-supermodularity suffices. This allows us to reduce to previous work in a more black-box manner.

2 Preliminaries

We will consider the following special cases of RSND (Definition 2):

- The k-Relative Survivable Network Design problem (k-RSND) is the special case of RSND where $r(s,t) \leq k$ for all $s,t \in V$. In this paper we consider the case $k = 3$, namely, the 3-RSND problem.
- The Single Demand k-Relative Survivable Network Design problem (SD-k-RSND) is the special case of RSND where $r(s,t) = k$ for exactly one vertex pair $s,t \in V$ and there is no demand for any other vertex pairs (equivalently, all other demands are 0). We consider the full SD-k-RSND problem for arbitrary k.

– The k-Edge-Fault-Tolerant-Subgraph problem (k-EFTS) is the special case of RSND where $r(s,t) = k$ for all $s,t \in V$.

For each of the listed RSND problem variants, we will use the following notation and definitions throughout. Let $G = (V, E)$ be a (multi-)graph and H a spanning subgraph (or an edge subset) of G. For $A \subseteq V$, let $\delta_H(A)$ denote the set of edges in H with exactly one endpoint in A, and let $d_H(A) = |\delta_H(A)|$ be their number. Additionally, let $G[A]$ denote the subgraph of G induced by the vertex set A. Let $s,t \in V$. We say that A is an st-**set** if $s \in A$ and $t \notin A$, and that $\delta_G(A)$ (or $\delta_E(A)$) is an st-**cut** of G (induced by A). An st-cut $\delta_G(A)$ (or an st-set A) is G-**minimal** if $\delta_G(A)$ contains no other st-cut of G. Assuming G is connected, it is easy to see that $\delta_G(A)$ is G-minimal if and only if both $G[A]$ and $G[V \setminus A]$ are connected. One can also see that if an st-cut $X \subseteq E$ is G-minimal, then $X = \delta_G(A)$ for some $A \subseteq V$. Finally, let $\lambda_G(s,t)$ denote the size of a min st-cut in G.

By Theorem 17 of [16], we may assume without loss of generality that the input graph G of any RSND instance is 2-edge-connected (or "2-connected").

3 2-Approximation for 3-RSND (and SD-3-RSND)

Given an RSND instance, we say that a vertex pair s,t is a k-**demand** if $r(s,t) = k$. We call a k-demand **relative** if the minimum st-cut has size less than k; that is, if $\lambda_G(s,t) < k$. A k-demand is then **ordinary** if $\lambda_G(s,t) \geq k$. Recall that SD-3-RSND has only one demand st, and that it is a 3-demand. The edges of any size 2 st-cut, or 2-st-cut, belong to any feasible solution so we call them **forced edges**. As a result, we can assume without loss of generality that they have cost 0.

3.1 Overview

We first give an overview of the theorems and proofs in this section. In order to prove Theorem 1, we will show that we can replace a single *relative* 3-demand by an equivalent set of *ordinary* 3-demands. More formally, we will prove the following.

Theorem 3. *Given an* SD-3-RSND *instance, there exists a polynomially computable set of ordinary* 3*-demands, D, such that for any $H \subseteq E$ that contains all forced edges, H is a feasible* SD-3-RSND *solution if and only if H satisfies all demands in D.*

This theorem reduces SD-3-RSND to the ordinary 3-SND problem (that is, the special case of SND where all demands are at most 3). In fact, this also gives us a lossless reduction from 3-RSND to 3-SND: Given a 3-RSND instance, we include the forced edges of all 3-demands into our solution, replace each relative 3-demand by an equivalent set of ordinary demands, and obtain an

equivalent ordinary 3-SND instance. Since SND admits approximation ratio 2, this reduction from 3-RSND to 3-SND implies Theorem 1.

We will also show that SD-3-RSND is approximation equivalent to certain instances of a special case of 3-SND. Before we define this special case, we must give a definition. A vertex subset R is a k-**edge-connected subset** in a graph H if $\lambda_H(u,v) \geq k$ for all vertex pairs $u, v \in R$. Since the relation $\{(u,v) \in V \times V : \text{no } (k-1)\text{-cut separates } u,v\}$ is transitive, this is equivalent to requiring that $\lambda_H(u,v) \geq k$ for pairs u, v that form a tree on R. We will prove that SD-3-RSND is approximation equivalent to special instances of the following problem:

4-Subset 3-EC
Input: A graph $J = (V', E')$ with edge costs, and a set $R \subseteq V'$ of at most 4 terminals.
Output: A min-cost subgraph H of J, such that R is 3-edge-connected in H.

More specifically, we will prove the following.

Theorem 4. *Let s and t be vertices in $J = (V', E')$, where J is the input graph to an instance of* 4-Subset 3-EC. SD-3-RSND *admits approximation ratio ρ if and only if* 4-Subset 3-EC *with the following properties* (A,B) *admits approximation ratio ρ:*

(A) $d_J(s) = d_J(t) = 2$ and R is the set of neighbors of s, t.
(B) If $d_J(A) = 2$ for some st-set A, then $A = \{s\}$ or $A = V' \setminus \{t\}$. Namely, if F is a set of 2 edges of J such that $J \setminus F$ has no st-path, then $F = \delta_J(s)$ or $F = \delta_J(t)$.

The general 4-Subset 3-EC problem admits approximation ratio 2, since it is a special case of SND. However, it is not actually known whether 4-Subset 3-EC is in P or is NP-hard (see [14,22] for results on a closely related problem). This 2-approximation is the best known for the 4-Subset 3-EC problem, so our 2-approximation for 3-RSND is the best we can hope for. In the rest of this section, we prove Theorems 4, 3, and 1. All missing proofs can be found in the full version [17].

3.2 Cactus Representation and Definitions

We first give some definitions and describe the cactus representation. The relation $\{(u,v) \in V \times V : \text{no } (k-1)\text{-cut separates } u,v\}$ is an equivalence, and we will call its equivalence classes k-**classes**. We construct a **cactus** $\mathcal{G}$ by shrinking every nontrivial 3-class (that is, every 3-class with at least 2 nodes) of the input graph G. Note that since G is 2-connected, $\mathcal{G}$ is a connected graph in which every two cycles have at most one node in common. Going forward, we will identify every 3-class with the corresponding node of $\mathcal{G}$. The edge pairs that belong to the same cycle of $\mathcal{G}$ are the 2-cuts of G.

We can assume that vertex pair st is a relative 3-demand. We say that the st-**chain of cycles** of $\mathcal{G}$ consists of all the cycles (and their nodes) in $\mathcal{G}$ that contain a 2-st-cut. We refer to the nodes, 3-classes, on these cycles as st-**relevant** nodes. Note that the set of edges in $\mathcal{G}$ that are in the st-chain of cycles are the forced edges. We also say that an st-relevant 3-class is **central** if it contains s or t, or belongs to two cycles of the st-chain. Additionally, the **attachment nodes** of an st-relevant 3-class are nodes in the 3-class that are either s or t, or are the ends of the edges (the **attachment edges**) that belong to some cycle in the st-chain of cycles. Since G is 2-connected, the number of attachment nodes in a non-central 3-class is exactly 2, while the number of attachment nodes in a central 3-class is between 2 and 4.

3.3 Proof of Theorems 4, 3, and 1

For the proof of Theorems 4 and 3, we associate with each st-relevant 3-class, C, a certain graph G_C which we call the **component of** C, obtained as follows:

- If C is a non-central 3-class then, in the graph obtained from G by removing the two attachment edges of C, G_C is the connected component that contains C.
- If C is a central 3-class, then removing the attachment edges of C results in at least one and at most two connected components that do not contain C – one contains s and the other contains t, if any. We obtain G_C from G by contracting the connected component that contains s into node s, and contracting the connected component that contains t into node t.

We now modify the central components G_C to satisfy properties (A,B) from Theorem 4. Consider some central 3-class C, and consider its component $J = G_C$. If J does not contain one of the original nodes s or t, then it has properties (A,B) and no modification is needed. If J contains the original node s, then we rename s to s', add a new node s, and connect new s by two zero cost edges to s'. The obtained J now has properties (A,B). A similar transformation applies if J contains the original node t.

The following lemma is about both the non-central components and these *modified* central components; in the lemma, we show that for H to be a feasible SD-3-RSND solution, it is necessary and sufficient to satisfy certain connectivity properties within each component.

Lemma 1. *Let H be a subgraph of G, and suppose that H contains all forced edges. Subgraph H is a feasible* SD-3-RSND *solution if and only if for every component J, the following holds.*

(i) *If J is a non-central component, then $H[J]$ contains two edge-disjoint uv-paths, where u and v are the two attachment nodes of J.*
(ii) *If J is a central component, then $H[J]$ is a feasible solution to the* SD-3-RSND *instance in J (with demand $r(s,t) = 3$).*

Suppose that for the special SD-3-RSND instances that arise in the central components we can achieve approximation ratio α. Then, we can achieve ratio α for general SD-3-RSND by picking into our solution H three types of edge sets.

1. The forced edges.
2. A min-cost set of 2 edge-disjoint paths between the attachment nodes of each st-relevant non-central component.
3. An α-approximate solution in each st-relevant central component.

Note that edges picked in steps 1,2 do not invoke any cost in the approximation ratio, since by Lemma 1 we actually pick parts of an optimal solution. Thus we get that the approximability of SD-3-RSND is equivalent to the approximability of the very special instances that arise in the central components. We will now show that these special instances from the central components are in fact instances of 4-Subset 3-EC with properties (A,B) from Theorem 4, thus proving Theorem 4. We will consider only central components with 4 attachment nodes; other cases with 3 or 2 attachment nodes are similar.

In what follows, let $\mathcal{I}$ be an SD-3-RSND instance on input graph J with properties (A,B) (just as in our central components). Let $R = \{x, y, z, w\}$ where x, y are the neighbors of s and z, w are the neighbors of t and let H be a subgraph of J that includes the four forced edges sx, sy, zt, and wt. We have the following.

Lemma 2. *Subgraph H is a feasible solution for instance $\mathcal{I}$ if and only if $R = \{x, y, z, w\}$ is a 3-edge-connected subset in H.*

By Lemma 2, H is a feasible solution for $\mathcal{I}$ if and only if H includes all forced edges and R is a 3-edge-connected subset—that is, R forms a feasible solution to 4-Subset 3-EC—in H. This, along with Lemma 1, implies that the approximability of SD-3-RSND is equivalent to that of 4-Subset 3-EC with properties (A,B), concluding the proof of Theorem 4.

Proof of Theorem 3. We will prove that a single relative 3-demand st can be replaced by an equivalent forest of ordinary 3-demands in polynomial time, where the trees in this forest span the sets of attachment nodes of the st-relevant 3-classes.

Recall that by Lemmas 1 and 2, subgraph H is a feasible SD-3-RSND solution for 3-demand st if and only if the following holds for every st-relevant 3-class C:

(i) If C is central, then the set R_C of attachment nodes of C is a 3-connected subset in H.
(ii) If C is non-central, then $H[C]$ contains 2 edge-disjoint uv-paths, where u and v are the attachment nodes of C.

The first condition is equivalent to satisfying a clique of 3-demands on R_C.[1] For the second condition, consider a non-central st-relevant 3-class C with attachment nodes u, v. One can see that if H contains all forced edges and satisfies

[1] Recall that since the relation $\{(u, v) \in V \times V :$ no 2-cut separates $u, v\}$ is transitive, this is equivalent to having a tree of 3-demands on R_C.

(i,ii) then the number of edge-disjoint uv-paths in H is larger by exactly 1 than their number in $H[C]$—the additional path (that exists in H but not in $H[C]$) goes along the cycle of the cactus that contains C, and there is exactly one such path. Thus, the demand $r(u,v) = 3$ is equivalent to requiring two edge-disjoint paths from u to v in R_C (in addition to including all forced edges).

We therefore obtain an equivalent 3-SND instance by replacing the single relative 3-demand st by a set D of 3-demands that form a clique (or, which is equivalent, a tree) on the set R_C of attachment nodes of every st-relevant 3-class C. These new demands can be computed in polynomial time, and they are ordinary 3-demands, since each R_C is a 3-edge-connected subset in G. This concludes the proof of Theorem 3.

Proof of Theorem 1. We can now describe a 2-approximation for 3-RSND. We treat each demand in the 3-RSND instance as its own instance of SD-3-RSND, solve each SD-3-RSND instance, and include the edges of each solution in our output.

4 SD-k-RSND

We give a recursive $2^{O(k^2)}$-approximation algorithm for SD-k-RSND for arbitrary constant k (Theorem 2). The algorithm is a generalization of the SD-3-RSND algorithm from [16]. At a high level, the main idea is to partition the input graph using a hierarchy of important separators, prove a structure theorem that characterizes the required connectivity guarantees within each component of the hierarchy, and then achieve these guarantees using a variety of subroutines.

4.1 Hierarchical Chain Decomposition

We first define important separators and describe how to use them to construct a hierarchical k-chain decomposition of G.

Definition 3. *Let X and Y be vertex sets of a graph G. An (X,Y)-separator of G is a set of edges S such that there is no path between any vertex $x \in X$ and any vertex $y \in Y$ in $G \setminus S$. An (X,Y)-separator S is minimal if no subset $S' \subset S$ is also an (X,Y)-separator. If $X = \{x\}$ and $Y = \{y\}$, we say that S is an (x,y)-separator.*

Definition 4 (Definition 20 of [16]). *Let S be an (X,Y)-separator of graph G, and let R be the vertices reachable from X in $G \setminus S$. Then S is an important (X,Y)-separator if S is minimal and there is no (X,Y)-separator S' such that $|S'| \leq |S|$ and $R' \subset R$, where R' is the set of vertices reachable from X in $G \setminus S'$.*

In Sect. 4.1 of [16], the authors describe how to construct the "$s-t$ 2-chain" of a graph G.[2] Here, we define the (X,Y) *h-chain* of G similarly, where X and Y are vertex sets and $h > 0$ is an integer.

[2] Note that all separator-based chain definitions given in this section are unrelated to the cactus-based chain definitions in Sect. 3.

First, if there are no important (X, Y)-separators of size h in G, then the (X, Y) h-chain of G is simply G and we're done (the chain is a single component, G, with no separators). If such an important separator exists, then we first find an important (X, Y)-separator S_0^h of size h in G, and we let R_0^h be the set of vertices reachable from any vertex $x \in X$ in $G \setminus S_0^h$. We let $V_{(0,r)}^h$ be the vertices in R_0^h incident on S_0^h, and let $V_{(1,\ell)}^h$ be the nodes in $V \setminus R_0^h$ incident on S_0^h. We then proceed inductively. Given $V_{(i,\ell)}^h$, if there is no important $(V_{(i,\ell)}^h, Y)$-separator of size h in $G \setminus (\cup_{j=0}^{i-1} R_j^h)$ then the chain is finished. Otherwise, let S_i^h be such a separator, let R_i^h be the nodes reachable from $V_{(i,\ell)}^h$ in $(G \setminus (\cup_{j=0}^{i-1} R_j^h)) \setminus S_i^h$, let $V_{(i,r)}^h$ be the nodes in R_i^h incident on S_i^h, and let $V_{(i+1,\ell)}^h$ be the nodes in $V \setminus (\cup_{j=0}^{i} R_j^h)$ incident on S_i^h. After this process completes we have our (X, Y) h-chain, consisting of components $R_0^h, \ldots, R_p^h$ along with important separators $S_0^h, \ldots, S_{p-1}^h$ between the components.

Next we note that by Lemma 21 of [16], we can find an important (X, Y)-separator of size h in polynomial time as long as h is a constant.

Lemma 3 Lemma 21 of [16]). *Let $d \geq 0$. An important (X, Y)-separator of size d can be found in time $4^d \cdot n^{O(1)}$ (if one exists), where $n = |V|$.*

Constructing the Hierarchical k-chain Decomposition. Now we describe how to construct the *hierarchical k-chain decomposition* of G. We start by creating the (s, t) 2-chain of G. We say that each component of the (s, t) 2-chain is a *2-component* of G in the hierarchical chain decomposition.

We then proceed inductively. Let R_i^h be an h-component of the hierarchical k-chain decomposition. If $h = k-1$, then the decomposition is finished. Otherwise, build the $(V_{(i,\ell)}^h, V_{(i,r)}^h)$ $(h+1)$-chain of R_i^h. The $(h+1)$-chain consists of $(h+1)$-components. Component R_i^h is the parent of these $(h+1)$-components. After this process completes we have our hierarchical k-chain decomposition of G.

The set of all h-components can be ordered as follows: The h-component that contains s is the first component while the h-component that contains t is last. All other h-components are adjacent via a left important separator and a right important separator to a left neighbor h-component and a right neighbor h-component, respectively.

4.2 Structure Theorem

Preliminaries. We say a subgraph H satisfies the RSND demand (X, Y, d) on input graph G if the following is true: If there is a path from at least one vertex in X to at least one vertex in Y in $G \setminus F$, where F is a set of at most $d-1$ edges, then there is a path from at least one vertex in X to at least one vertex in Y in $H \setminus F$. Going forward, if $V_{(i,\ell)}^h = \{s\}$, then we consider S_{i-1}^h to be the empty set. Similarly, if $V_{(i,r)}^h = \{t\}$, then S_i^h is the empty set.

Fix an h-component R_i^h and let X be a vertex set such that $X \subseteq V_{(i,\ell)}^h$. We say that S_X is the set of edges in S_{i-1}^h incident on vertices in X. Similarly, if

Y is a vertex set such that $Y \subseteq V^h_{(i,r)}$, we say that S_Y is the set of edges in S^h_i incident on vertices in Y. We will also use S to denote the set of all edges in an important separator in the decomposition. Let H be a subgraph of G. We will also say that $G^h_i = G[R^h_i]$ is the subgraph of G induced by the vertex set R^h_i, and that $H^h_i = H[R^h_i]$ is the subgraph of H induced by R^h_i.

We can now use the hierarchical chain construction to give a structure lemma that characterizes feasible solutions. The lemma states that a subgraph H of G is a feasible solution to SD-k-RSND if and only if in the hierarchical k-chain decomposition of G, all edges in S are in H, and certain connectivity requirements between groups of vertices in $V^h_{(i,\ell)}$ and in $V^h_{(i,r)}$ are met in H^h_i for each component R^h_i in the decomposition.

Theorem 5 (Structure Theorem). *Subgraph H is a feasible solution to SD-k-RSND if and only if all edges in S are included in H, and for each h-component R^h_i in the hierarchical k-chain decomposition of G, subgraph H^h_i satisfies the following:*

1 H^h_i is a feasible solution to RSND on subgraph G^h_i with demands

$$\Big\{(X,Y,d) \;:\; X \subseteq V^h_{(i,\ell)},\; Y \subseteq V^h_{(i,r)},\; (X,Y) \neq \big(V^h_{(i,\ell)}, V^h_{(i,r)}\big),\\ d = \max(0, k + |S_X| + |S_Y| - |S^h_{i-1}| - |S^h_i|)\Big\}.$$

2 H^h_i is a feasible solution to RSND on subgraph G^h_i with demand

$$\left(V^h_{(i,\ell)}, V^h_{(i,r)}, h+1\right).$$

3 H^h_i is a feasible solution to RSND on subgraph G^h_i with demand

$$\left(V^h_{(i,\ell)}, V^h_{(i,r)}, k-1\right).$$

The proof of this structure theorem is long and involved; due to space constraints it can be found in the full version [17].

4.3 Algorithm and Analysis

Algorithm. We can now use this Structure Theorem to give a $2^{O(k^2)}$-approximation for SD-k-RSND. We first create the hierarchical k-chain decomposition of G in polynomial time, as described in Sect. 4.1. Within each component we run a set of algorithms to satisfy the RSND demands stated in Theorem 5. Our solution, H, includes the outputs of each of these algorithms along with S, the set of all edges in the separators of the hierarchical k-chain decomposition. We now describe the set of algorithms run on each component in the hierarchical k-chain decomposition. Fix a component R^h_i of the decomposition and let $X \subseteq V^h_{(i,\ell)}$, $Y \subseteq V^h_{(i,r)}$, and $d = \max(0, k + |S_X| + |S_Y| - |S_{i-1}| - |S_i|)$:

- **Base Case (Shortest st Path).** For each X, Y pair such that $d = 1$, contract the vertices in X and contract the vertices in Y to create super nodes x and y, respectively. We first check in polynomial time if x and y are connected in $G_i^h = G[R_i^h]$. If they are connected, then we create an instance of the Weighted st Shortest-Path problem on G_i^h (in polynomial time), using x and y as our source and destination nodes. For each edge $e \in E(R_i^h)$, set the weight of e to $w(e)$. Run a polynomial-time Weighted st Shortest-Path algorithm on this instance (e.g. Dijkstra's algorithm), and add to H all edges in the output of the algorithm.
- **Recursive Step.** For each X, Y pair such that $1 < d < k$, we create an instance of SD-d-RSND on G_i^h. Contract the vertices in X, and in Y, to create super nodes x and y, respectively. For each $e \in E(R_i^h)$, set the cost of e to $w(e)$. The set of RSND demands is just $\{(x, y, d)\}$. Run the recursive polynomial-time SD-d-RSND algorithm on this instance, where $d < k$. Add to H all edges in the output of the algorithm.
- **Final Recursive Step.** We create an SD-$(k-1)$-RSND instance on G_i^h. Contract the vertices in $V_{(i,\ell)}^h$, and in $V_{(i,r)}^h$, to create super nodes v_ℓ and v_r, respectively. For each $e \in E(R_i^h)$, set the cost of e to $w(e)$. The set of RSND demands is just $\{(v_\ell, v_r, k-1)\}$. Run the recursive SD-$(k-1)$-RSND algorithm on this instance. Add to H all edges in the output of the algorithm.
- **Min-Cost Flow.** We create an instance of the Min-Cost Flow problem on G_i^h. Contract the vertices in $V_{(i,\ell)}^h$, and in $V_{(i,r)}^h$, to create super nodes v_ℓ and v_r, respectively. Let v_ℓ be the source and v_r the sink. For each $e \in E(R_i^h)$, set the capacity of e to 1 and the cost of e to $w(e)$. Require a minimum flow of $h + 1$, and run a poly-time Min-Cost Flow algorithm on this instance. Since all capacities are integer the algorithm will return an integral flow, so we add to H all edges with non-zero flow.

Analysis. All missing proofs from this section can be found in the full version [17]. The following lemma is essentially directly from Theorem 5 and the description of the algorithm.

Lemma 4. *Let H be the output of the algorithm given in Sect. 4.3. Subgraph H is a feasible solution to the SD-k-RSND problem.*

Let H^* denote the optimal solution, and for any set of edges $A \subseteq E$, let $w(A) = \sum_{e \in A} w(e)$. The next lemma follows from combining the approximation ratios of each of the subroutines used in the algorithm and solving the recurrence.

Lemma 5. $w(H) \leq 2^{O(k^2)} * w(H^*)$.

Theorem 2 is directly implied by Lemmas 4 and 5 together with the observation that the algorithm runs in polynomial time.

References

1. Adjiashvili, D., Hommelsheim, F., Mühlenthaler, M.: Flexible graph connectivity: approximating network design problems between 1- and 2-connectivity (2020)
2. Adjiashvili, D., Hommelsheim, F., Mühlenthaler, M., Schaudt, O.: Fault-tolerant edge-disjoint paths - beyond uniform faults (2020)
3. Bansal, I., Cheriyan, J., Grout, L., Ibrahimpur, S.: Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions (2022)
4. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Multiple-edge-fault-tolerant approximate shortest-path trees (2016). https://doi.org/10.48550/ARXIV.1601.04169
5. Bodwin, G., Dinitz, M., Nazari, Y.: Vertex fault-tolerant emulators. In: Braverman, M. (ed.) 13th Innovations in Theoretical Computer Science Conference, ITCS 2022. LIPIcs, vol. 215, pp. 25:1–25:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ITCS.2022.25
6. Bodwin, G., Dinitz, M., Nazari, Y.: Epic fail: emulators can tolerate polynomially many edge faults for free. In: 14th Innovations in Theoretical Computer Science Conference, ITCS 2023 (2023)
7. Bodwin, G., Dinitz, M., Parter, M., Williams, V.V.: Optimal vertex fault tolerant spanners (for fixed stretch). In: Czumaj, A. (ed.) Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7–10 January 2018, pp. 1884–1900. SIAM (2018)
8. Bodwin, G., Dinitz, M., Robelle, C.: Optimal vertex fault-tolerant spanners in polynomial time. In: Naor, J.S., Buchbinder, N. (eds.) Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, pp. 2924–2938. SIAM (2022). https://doi.org/10.1137/1.9781611976465.174
9. Bodwin, G., Patel, S.: A trivial yet optimal solution to vertex fault tolerant spanners. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, pp. 541–543. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3293611.3331588
10. Boyd, S., Cheriyan, J., Haddadan, A., Ibrahimpur, S.: Approximation algorithms for flexible graph connectivity (2022)
11. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: Fault tolerant spanners for general graphs. SIAM J. Comput. **39**(7), 3403–3423 (2010)
12. Chekuri, C., Jain, R.: Approximating flexible graph connectivity via räcke tree based rounding (2022)
13. Chekuri, C., Jain, R.: Augmentation based approximation algorithms for flexible network design (2022)
14. Cheriyan, J., Laekhanukit, B., Naves, G., Vetta, A.: Approximating rooted Steiner networks. ACM Trans. Algorithms **11**(2), 8:1–8:22 (2014)
15. Cheriyan, J., Thurimella, R.: Approximating minimum-size k-connected spanning subgraphs via matching. SIAM J. Comput. **30**(2), 528–560 (2000). https://doi.org/10.1137/S009753979833920X
16. Dinitz, M., Koranteng, A., Kortsarz, G.: Relative survivable network design. In: APPROX-RANDOM, vol. 245, pp. 41:1–41:19 (2022)
17. Dinitz, M., Koranteng, A., Kortsarz, G., Nutov, Z.: Improved approximations for relative survivable network design (2023). https://doi.org/10.48550/arXiv.2304.06656
18. Dinitz, M., Krauthgamer, R.: Fault-tolerant spanners: better and simpler. In: Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, 6–8 June 2011, pp. 169–178 (2011)

19. Dinitz, M., Robelle, C.: Efficient and simple algorithms for fault-tolerant spanners. In: Emek, Y., Cachin, C. (eds.) ACM Symposium on Principles of Distributed Computing, PODC 2020, pp. 493–500. ACM (2020). https://doi.org/10.1145/3382734.3405735
20. Dinitz, Y., Westbrook, J.: Maintaining the classes of 4-edge-connectivity in a graph on-line. Algorithmica **20**, 242–276 (1998)
21. Dinitz, Y., Nutov, Z.: A 2-level cactus model for the system of minimum and minimum+ 1 edge-cuts in a graph and its incremental maintenance. In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, pp. 509–518 (1995)
22. Feldmann, A.E., Mukherjee, A., van Leeuwen, E.J.: The parameterized complexity of the survivable network design problem. In: SOSA, pp. 37–56 (2022)
23. Gabow, H.N., Goemans, M.X., Tardos, É., Williamson, D.P.: Approximating the smallest k-edge connected spanning subgraph by LP-rounding. Networks **53**(4), 345–357 (2009)
24. Henzinger, M.R.: A static 2-approximation algorithm for vertex connectivity and incremental approximation algorithms for edge and vertex connectivity. J. Algorithms **24**(1), 194–220 (1997)
25. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner network problem. Combinatorica **21**(1), 39–60 (2001). https://doi.org/10.1007/s004930170004
26. Khandekar, R., Kortsarz, G., Nutov, Z.: Approximating fault-tolerant Group-Steiner problems. Theor. Comput. Sci. **416**, 55–64 (2012)
27. Lo, O.S., Schmidt, J.M., Thorup, M.: Compact cactus representations of all non-trivial min-cuts. Discret. Appl. Math. **303**, 296–304 (2021)
28. Marx, D.: Important separators and parameterized algorithms. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 5–10. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25870-1_2
29. Poutre, J.A.L.: Maintenance of 2- and 3-edge-connected components of graphs II. SIAM J. Comput. **29**(5), 1521–1549 (2000)
30. Williamson, D.P., Goemans, M.X., Mihail, M., Vazirani, V.V.: A primal-dual approximation algorithm for generalized Steiner network problems. Combinatorica **15**(3), 435–454 (1995). https://doi.org/10.1007/BF01299747