



Improved Approximations for Flexible Network Design

Dylan Hyatt-Denesik  

Department of Mathematics and Computer Science, Eindhoven University of Technology,
The Netherlands

Afrouz Jabal-Ameli  

Department of Mathematics and Computer Science, Eindhoven University of Technology,
The Netherlands

Laura Sanità  

Department of Computing Sciences, Bocconi University, Milan, Italy

Abstract

Flexible network design deals with building a network that guarantees some connectivity requirements between its vertices, even when some of its elements (like vertices or edges) fail. In particular, the set of edges (resp. vertices) of a given graph are here partitioned into *safe* and *unsafe*. The goal is to identify a minimum size subgraph that is 2-edge-connected (resp. 2-vertex-connected), and stay so whenever any of the unsafe elements gets removed.

In this paper, we provide improved approximation algorithms for flexible network design problems, considering both edge-connectivity and vertex-connectivity, as well as connectivity values higher than 2. For the vertex-connectivity variant, in particular, our algorithm is the first with approximation factor strictly better than 2.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, Network Design, Flexible Connectivity

Digital Object Identifier 10.4230/LIPIcs.ESA.2024.74

Related Version *Full Version*: <https://arxiv.org/abs/2404.08972> [14]

Funding The second and the third authors are grateful for the support received from the NWO-VIDI grant VI.Vidi.193.087.

1 Introduction

Survivable network design is an important area of combinatorial optimization. In a classical setting, we are given a network represented as a graph, and the goal is to select the cheapest subset of edges that guarantee some connectivity requirements among its vertices, even when some of its elements (like vertices or edges) may fail.

Two fundamental problems in this area are the *2-edge-connected spanning subgraph* (2ECSS) and the *2-vertex-connected spanning subgraph* (2VCSS). These problems aim at building a network resilient to a possible failure of an edge or of a vertex, respectively. More in detail, in 2ECSS, we are given in input a graph $G = (V, E)$, and the goal is to select a subset $F \subseteq E$ of minimum-cardinality such that the graph (V, F) is 2-edge-connected: that is, (V, F) contains 2 edge-disjoint paths between any pair of vertices. In 2VCSS, the input is the same, but here we require that our selected set F is such that (V, F) is 2-vertex connected, i.e., it contains 2 vertex-disjoint paths between any pair of vertices. Both 2ECSS and 2VCSS have been extensively studied in the literature. They are APX-hard (see [11]) but admit constant-factor approximation algorithms (see e.g. [3, 13, 17, 20]). Currently, the



© Dylan Hyatt-Denesik, Afrouz Jabal-Ameli, and Laura Sanità;
licensed under Creative Commons License CC-BY 4.0

32nd Annual European Symposium on Algorithms (ESA 2024).

Editors: Timothy Chan, Johannes Fischer, John Iacono, and Grzegorz Herman; Article No. 74; pp. 74:1–74:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

best approximation for 2ECSS is $\frac{118}{89}$ [13]¹, while the best approximation for 2VCSS is $\frac{4}{3}$ [3]. The problems have been investigated also in the *weighted* setting. That is, when the input graph G comes equipped with an edge-weight function $w \in \mathbb{R}_+$, and the goal is to minimize the total weight of the selected set F , rather than its cardinality. For this more general case, nothing better than a 2-approximation is known [15, 16].

In recent years, an interesting generalization has been introduced by Adjiashvili et al. [1], which soon received a lot of attention in the network design community. This generalization is called *flexible* graph connectivity and is the focus of this paper. Specifically, the authors in [1] considered a scenario in which not all edges are subject to potential failures. The set of edges is partitioned into *safe* and *unsafe*, and the goal is to construct a network resilient to the failure of unsafe edges. A formal definition is given below.

► **Definition 1** (Flexible Graph Connectivity Problem (FGC)). *Given a graph $G = (V, E)$ and a partition of E into safe edges E_S and unsafe edges E_U , the goal is to find the smallest subset E' of E such that (1) $H = (V, E')$ is connected, (2) and for every edge $e \in E_U \cap E'$, the graph $(V, E' \setminus e)$ is connected.*

Next, we define the vertex-connectivity version of the problem, investigated first in [5]. Recall that a *cut-vertex* of a graph is a vertex u such that if we remove u and all its incident edges the number of connected components of the graph increases.

► **Definition 2** (Flexible Vertex Connectivity Problem (FVC)). *Given a graph $G = (V, E)$ and a partition of V into safe vertices V_S and unsafe vertices V_U , the goal is to find the smallest subset E' of E such that (1) $H = (V, E')$ is connected, (2) and for every vertex $u \in V_U$, u is not a cut-vertex of H .*

Note that, when $E_S = \emptyset$, FGC reduces to 2ECSS. Similarly, when $V_S = \emptyset$, FVC reduces to 2VCSS. Hence, these problems are at least as hard as 2ECSS and 2VCSS, respectively.

For FGC and some further generalizations, several approximation results have been developed in the past few years (see e.g. [1, 2, 4, 5, 6, 7, 8, 9, 19], and application of flexible graph based techniques). This list shows a growing interest in this problem and its variants in the literature.

The current best known approximation factor for FGC is given in [8], which is based on the best 2ECSS approximation ratio. At the moment, using the best approximation result available in the literature for 2ECSS, [13], this translates into a bound of $\frac{472}{385} \approx 1.45$.²

Of course, a first natural question is the following:

■ *Can the approximation factor for FGC be improved?*

Problems involving vertex-connectivity rather than edge-connectivity often turn out to be more challenging to address, and in fact approximation results on FVC are more scarce. The authors in [5] studied the problem in the *weighted* setting, and give a 2-approximation under the assumption to have at least one safe vertex (they target a more general setting of removing k unsafe vertices). Note that a 2-approximation is known for the weighted setting of FGC as well, however as mentioned before, for the standard FGC a significantly better approximation is known. This raises the question of whether a better-than-2 approximation is possible also for FVC.

■ *Is there an algorithm for FVC with approximation factor better than 2?*

¹ Recently a 1.3-approximation was presented for 2ECSS (see [18]) with the assumption that there exists a polynomial time algorithm for maximum triangle-free 2-matching problem.

² If one applies the aforementioned 1.3-approximation found in [18], then this approximation becomes $1.\bar{4}$.

Finally, we observe that both 2ECSS and 2VCSS have been studied in a generalized setting, called k ECSS and k VCSS respectively, where one requires k -edge-disjoint paths (resp. k -vertex-disjoint paths) between each pair of vertices. It turns out that for higher values of connectivity requirements, much better approximations can be developed. In particular, [10, 12] show an approximation factor of $1 + O(\frac{1}{k})$ for k ECSS and k VCSS. It is natural to ask what happens if we generalize FGC and FVC in a similar way. Specifically, we could aim at building graphs that are k -edge-connected (resp. k -vertex-connected) after the removal of any unsafe edge. The last question we are interested in is then the following:

- *Can we obtain similar approximation bounds as observed for k ECSS or k VCSS, if we consider FGC and FVC with a higher value of k as connectivity requirement?*

Our Results

In this paper, we focus on the above questions and answer them in a positive way.

In Section 3, we prove Theorem 3, which improves the best known approximation factor for FGC. The theorem is proved by giving a refined analysis of the algorithm developed in [8]. Their algorithm relies on an approximation algorithm for 2ECSS as a subroutine, whose analysis is used mainly when the size of the optimal solution is large enough compared to the number of vertices n . Our improvement stems from realizing that 2ECSS can be approximated better than the current best known factor whenever the optimal solution is large compared to n see Lemmas 30 and 31.

► **Theorem 3.** *There is a polynomial time $\frac{10}{7}$ -approximation algorithm for FGC.*

In Section 2 we prove Theorem 4, which yields the first approximation algorithm for FVC with an approximation factor strictly better than 2. Its proof constitutes the most technical part of our paper. It combines two different main algorithms, which rely on non-trivial combinatorial ingredients, like ear-decompositions and matroid intersection.

► **Theorem 4.** *There is a polynomial time $\frac{11}{7}$ -approximation algorithm for FVC.*

Finally, we answer the last question we raised, for FGC specifically. In particular, we consider the FGC problem for higher values of k , formally defined below.

► **Definition 5** (*k -Flexible Graph Connectivity Problem (k -FGC)*). *Given a graph $G = (V, E)$ and a partition of E into safe edges E_S and unsafe edges E_U , the goal is to find the smallest subset E' of E such that (1) $H = (V, E')$ is 1-edge-connected, (2) and for every k unsafe edges $\{e_1, \dots, e_k\} \subseteq E_U \cap E'$, the graph $(V, E' \setminus \{e_1, \dots, e_k\})$ is 1-edge-connected.*

Theorem 6 shows that also k -FGC becomes somewhat easier to approximate when k grows, as it happens for k ECSS.³ Due to space limitations, its proof can be found in the full version of this paper.

► **Theorem 6.** *There is a polynomial time $1 + O(\frac{1}{\sqrt{k}})$ -approximation algorithm for k -FGC.*

We were not able to extend our arguments to FVC with higher values of connectivity requirement k in a similar manner to k -FGC. We leave this as an open question.

³ We note that a (stronger) approximability of $1 + O(\frac{1}{k})$ for k -FGC was previously claimed by [1], but their proof is flawed as we explain in the full version of this paper

1.1 Preliminaries

Here, we begin with some preliminaries to define and construct some tools that will be helpful later on in the paper.

Given a graph $G = (V, E)$ and a subset of vertices $U \subseteq V$. We denote by G/U the (multi-)graph obtained by first replacing the vertices of U by a single vertex \hat{U} , and then for every edge $uv \in E$ such that $u \in U$ and $v \in V \setminus U$ we add an edge $\hat{U}v$ to E (Note that there can be multiple copies of the same edge $\hat{U}v$ in G/U , and there will be no loops on \hat{U}). We sometimes refer to this as *contracting* G by the vertices U . We can define a similar operation on a subset of edges $F \subseteq E$, by taking the vertex sets of connected components of (V, F) , and contracting G by these sets one by one. We denote this operation by G/F .

Given a graph $G = (V, E)$, and a subset of vertices $U \subseteq V$, we define $G[U] = (U, \{u_1u_2 \in E \mid u_1, u_2 \in U\})$, the *induced* subgraph of U . Similarly, we can define an induced subgraph on a subset of edges $F \subseteq E$, and by abuse of notation we use the same notation $G[F] = (\{v \in V \mid \exists u \in V, uv \in F\}, F)$. We also say a graph $H = (V', E')$ is a *spanning subgraph* of G if H is a subgraph of G and $V' = V$.

► **Definition 7 (Ear-Decomposition).** *Let $G = (V, E)$ be a graph. An ear-decomposition is a sequence P_1, \dots, P_k , where P_1 is a cycle of G , and for each $i \in \{2, \dots, k\}$, P_i is either:*

- *a path sharing exactly its two endpoints with $V(P_1) \cup \dots \cup V(P_{i-1})$, or;*
- *a cycle that shares exactly one vertex with $V(P_1) \cup \dots \cup V(P_{i-1})$.*

P_1, \dots, P_k are called *ears*. P_i is an *open ear* if it is a path. An ear-decomposition is *open* if for every $i \in \{2, \dots, k\}$, P_i is an open ear. We refer to $|E(P)|$ as the *length* of P .

Given an open ear-decomposition P_1, \dots, P_k , we say that P' is a *potential open ear* of P_1, \dots, P_k if P_1, \dots, P_k, P' is itself an open ear-decomposition. P_1, \dots, P_k is an *open ear-decomposition* of a graph $G = (V, E)$ if $(V(P_1) \cup \dots \cup V(P_k) = V$ and for each i , $E(P_i) \subseteq E$.

We will often abbreviate the adjective ‘2-vertex-connected’ with ‘2VC’. The following well known result on open ear-decompositions can be found in Chapter 4 of [21].

► **Lemma 8** ([21]). *A graph G is 2VC if and only if it has an open-ear decomposition.*

An important tool for the analysis of our algorithm in Section 2.2 is the following, which will let us characterize the vertex connectivity of a graph.

► **Definition 9 (Blocks).** *Let $G = (V, E)$ be a graph such that $|V(G)| \geq 2$. A *block* of G is a maximal connected subgraph of G that has at least one edge and has no cut vertex. Therefore if G has no self-loops, a block is either an induced connected subgraph on two vertices or it is a maximal 2VC subgraph on at least three vertices.*

We end this section by introducing some useful properties of blocks. For proofs and more details on these, we refer the reader to Chapter 4 of [21].

► **Lemma 10** ([21]). *Let $G = (V, E)$ be a graph, and let $\{B_1, \dots, B_k\}$ be the set of all blocks of G . The following properties hold*

1. *Two blocks share at most one vertex.*
2. *The blocks B_1, \dots, B_k of G partition E , that is $E = \cup_{i=1}^k E(B_i)$, and $E(B_i) \cap E(B_j) = \emptyset$, for $i \neq j$.*
3. *For two distinct edges e_1 and e_2 , e_1 and e_2 belong to the same block B_i if and only if there is a cycle in B_i that contains e_1 and e_2 .*
4. *If G is connected, G has at most $|V| - 1$ blocks.*

We also use the following useful Lemma, whose proof can be found in the full version of this paper.

► **Lemma 11.** *Let $G = (V, E)$ be a connected graph with $B(G)$ many blocks. Let H be a connected, spanning subgraph of G with $B(H) > B(G)$ many blocks. In polynomial time, we can find an edge $e \in E(G) \setminus E(H)$ such that $H' := (V(H), E(H) \cup \{e\})$ has fewer than $B(H)$ blocks.*

2 $\frac{11}{7}$ -Approximation for FVC

In this section we provide a $\frac{11}{7}$ -approximation algorithm for Flexible Vertex Connectivity Problem. We assume that our given graph $G = (V, E)$, is a simple graph, that is, G does not contain any loops or parallel edges (note that even if G were not simple, parallel edges and loops would not help in finding a feasible solution), and has $n := |V|$ vertices.

We begin this section by showing that in order to obtain a β -approximation one can assume that the input graph (i.e. G) has some additional properties, such as not containing specific subgraphs that we call *forbidden cycles*.

► **Definition 12 (Forbidden Cycle).** *We say that a 4-cycle C in G is a forbidden cycle if C has two vertices w and z such that $wz \notin E(C)$ and $\deg_G(w) = \deg_G(z) = 2$.*

The following Lemma allows us to assume without loss of generality that our input graph G is 2VC and does not contain a forbidden cycle. Its proof can be found the full version of this paper.

► **Lemma 13.** *If there is a β -approximation for FVC instances that are 2VC and do not contain forbidden cycles, then there is a β -approximation for FVC.*

Furthermore, using the next Lemma we assume throughout Section 2 that $OPT \geq n$ as otherwise we can solve the problem optimally in polynomial time. The proof of this Lemma is straightforward can be found in the full version of this paper for completeness sake.

► **Lemma 14.** *Let OPT be an optimal solution to FVC instance $G = (V, E)$. We have $|OPT| \geq n - 1$. Furthermore, if $|OPT| = n - 1$, then we can find such a solution in polynomial time.*

2.1 Approximation 1

By applying Lemma 13, we can assume without loss of generality that G is 2VC and does not contain a forbidden cycle. We also assume that G has at least 5 vertices, as we can handle smaller instances by enumeration. Our algorithm relies on the construction of a certain open ear-decomposition which we outline here. Start with D being a cycle of length at least four. We remark that such a cycle must exist, as $n \geq 4$, and G is 2VC (see [21]). Moreover one can find such a cycle in polynomial time. Now, until there exists a potential open ear of D that has a length of at least 4 for D , we find and add one such potential open ear to D . We observe by Lemma 8 that D is 2VC. To show that this procedure terminates in polynomial time we rely on the following simple claim, proven in the full version of this paper.

▷ **Claim 15.** Let D be an open-ear decomposition. If there exists a potential open ear of D with a length of at least 4, it can be detected in polynomial time.

The following Lemma, provides an upper bound on the number of edges in D . The proof of this lemma is very straightforward and is proven in the full version of this paper for completeness.

► **Lemma 16.** *When the algorithm terminates, we have $|E(D)| \leq \frac{4}{3}(|V(D)| - 1)$.*

With this open ear decomposition D , the following key Lemma, proven in the full version of this paper, shows that the edges of $G[V \setminus V(D)]$ have a useful structure, that will be critical to our approximation algorithms. Critically, the proof of this lemma relies on the assumption that G contains no forbidden cycles.

► **Lemma 17.** *The edges of $G[V \setminus V(D)]$ form a (not necessarily perfect) matching.*

Since the edges of $G[V \setminus V(D)]$ form a matching, each connected component of $G[V \setminus V(D)]$ is either a singleton or an edge. We will now partition the vertices of $V \setminus V(D)$ into the sets $K_{1,1}, K_{1,2}, K_{2,2}$, and $K_{2,3}$.

We define $K_{1,1} \subseteq V \setminus V(D)$ as the singletons of $G[V \setminus V(D)]$ that have a safe vertex neighbour in $V(D)$. We define $K_{1,2} \subseteq V \setminus V(D)$ as the singletons of $G[V \setminus V(D)]$ that do not have a safe vertex neighbour in $V(D)$. We define $K_{2,2} \subseteq V \setminus V(D)$ as the endpoints of edges uv in $G[V \setminus V(D)]$ that satisfy one of the following: u and v are both adjacent to a (possibly equal) safe vertex in $V(D)$, or one of u or v are safe, and that safe vertex is adjacent to a safe vertex in $V(D)$. Finally, we define $K_{2,3} = V \setminus (V(D) \cup K_{1,1} \cup K_{1,2} \cup K_{2,2})$.

Intuitively, we imagine the set $K_{i,j}$ to represent vertices of the components of $G[V \setminus V(D)]$ with i vertices, where any feasible FVC of solution of G must have at least j edges with endpoints in a component.

We now describe our first algorithm, which will compute a feasible solution APX_1 . Starting with $APX_1 := \emptyset$. We first add the edges of D to APX_1 . If $V \setminus V(D) = \emptyset$, then APX_1 is feasible since D is 2VC, and we are done. Otherwise, we buy edges in the following way to make APX_1 feasible.

First, for every $v \in K_{1,1}$, buy an edge $uv \in E$ where $u \in V(D)$ is safe (such a u exists, by definition of $K_{1,1}$). Second, for every $v \in K_{1,2}$, we buy arbitrary pair of edges $uv, u'v \in E$, where $u \neq u'$ (these edges exist since G is assumed to be 2VC). Third, for every edge uv of $G[K_{2,2}]$, if u and v are both adjacent to (potentially equal) safe vertices u' and v' in $V(D)$, then we buy the edges uu' and vv' . If at least one of u and v , is not adjacent to a safe vertex in $V(D)$, then by definition of $K_{2,2}$, (without loss of generality) v is safe and is adjacent to a safe vertex in $v' \in V(D)$; In that case, we buy edges uv, vv' . Lastly, for each edge uv in $G[K_{2,3}]$, we buy edge uv and arbitrary pair of edges uu' and vv' , where $u' \neq v'$. Observe that such edges must exist again as G is 2VC and $n \geq 4$.

We now state the following Lemma, which is proven in the full version of the paper.

► **Lemma 18.** *Our algorithm computes a feasible FVC solution, APX_1 , in polynomial time.*

By our construction of APX_1 , $|APX_1| \leq |E(D)| + |K_{1,1}| + 2|K_{1,2}| + |K_{2,2}| + \frac{3}{2}|K_{2,3}|$. The following Lemma is clear since $|E(D)| \leq \frac{4}{3}(|V(D)| - 1)$, by Lemma 16.

► **Lemma 19.** $|APX_1| \leq \frac{4}{3}(|V(D)| - 1) + |K_{1,1}| + 2|K_{1,2}| + |K_{2,2}| + \frac{3}{2}|K_{2,3}|$.

We fix an optimal solution OPT to the instance $G = (V, E)$. The following Lemma provides a lower bound on $|OPT|$. The proof can be found in the full version of the paper.

► **Lemma 20.** $|OPT| \geq \max\{n, |K_{1,1}| + 2|K_{1,2}| + |K_{2,2}| + \frac{3}{2}|K_{2,3}|\}$.

In the next Lemma show that even with the tools we have already developed, we have a $\frac{5}{3}$ -approximation, answering the question if there exists an approximation factor less than 2 in the affirmative. We spend the remainder of the section improving on this easier approximation. The proof can be found in the full version of the paper.

► **Lemma 21.** *APX_1 is a $\frac{5}{3}$ -approximate solution for FVC. Furthermore, if $|K_{1,2}| + |K_{2,3}| \leq 2$, then APX_1 is a $\frac{4}{3}$ -approximate solution.*

2.2 Approximation 2

In this section, we will provide a second algorithm that relies on the vertex sets defined in Section 2.1, that were computed by our first algorithm. Namely, we are interested in $V(D)$, $K_{1,1}$, $K_{1,2}$, $K_{2,2}$, and $K_{2,3}$. This algorithm, when combined with the algorithm of Section 2.1 will achieve a $\frac{11}{7}$ -approximation. Applying Lemma 21, we assume that at least one of $K_{1,2}$ or $K_{2,3}$ is non-empty, as otherwise we immediately have a $\frac{4}{3}$ -approximation algorithm.

As in the previous section, we will rely on the fact that D is constructed as a 2VC subgraph of G , and the fact that any feasible solution must take edges incident to vertices in $K_{1,1}$, $K_{1,2}$, $K_{2,2}$, and $K_{2,3}$. However this time we start by buying a minimal set of edges E' incident to $K_{1,1}$, $K_{1,2}$, $K_{2,2}$, and $K_{2,3}$ that are required for feasibility and then we complement these edges with a subset of edges of $G[D]$ to obtain a feasible solution APX_2 . One important ingredient for our second algorithm is to use the following well-studied problem.

► **Definition 22** (Maximum Rainbow Connection Problem). *Given a (multi-)graph G and a coloring $c : E \rightarrow \mathbb{N}$ of the edges, find a spanning subgraph of G that minimizes the number of components, while choosing exactly one edge from each colour.*

This problem can be solved to optimality using matroid intersection between the graphic matroid, and the partition matroid on the colour classes. Furthermore, we remark that this problem has been studied in the area of Survivable Network Design [13, 20]. In the following Lemma, which is proven the full version of the paper, we in fact consider a slight generalization of this problem where the number of isolated vertices is also minimized. This additional property (minimizing the number of isolated vertices) is a key part of our algorithm analysis.

► **Lemma 23.** *Given an instance of the Maximum Rainbow Connection Problem with (multi-)graph $G = (V, E)$, with coloring $c : E \rightarrow \mathbb{N}$. We can find in polynomial time, an optimal solution P such that the number of isolated vertices (vertices of degree 0) in (V, P) is minimal with respect to replacing an edge with another edge of the same colour class.*

Our goal with the Maximum Rainbow Connection problem is to more cleverly find a minimal set E' than simply taking an *arbitrary* minimal set of edges incident on at least one vertex of $V \setminus V(D)$. Our goal is to select such an E' that also minimizes the number of connected components of (V, E') . To do this we use the edges of E that are incident on $K_{1,1}$, $K_{1,2}$, $K_{2,2}$, and $K_{2,3}$ to create an instance of Maximum Rainbow Connection Problem (and solve it with Lemma 23).

We define a set of so-called *pseudo-edges* \tilde{E} with endpoints in $V(D)$ (named pseudo-edges in order to distinguish them from the “real” edges, E), and assign to each pseudo-edge a unique colour indexed by vertices in $K_{1,2}$ and pairs of adjacent vertices in $K_{2,3}$. For every $u \in K_{1,2}$, and every distinct pair of edges uv_1 and $uv_2 \in E$, we add pseudo-edge v_1v_2 to \tilde{E} . Assign v_1v_2 the colour c_u . Intuitively, a pseudo-edge xy with colour c_v (for example) corresponds to a path in E from x to v to y .

For every ordered pair $(u, v) \in K_{2,3} \times K_{2,3}$ such that $uv \in E$, we add pseudo-edges to \tilde{E} in the following way: (1) for every pair of edges uu' and vv' with $u' \neq v'$, we add pseudo-edge $u'v'$ to \tilde{E} . Assign $u'v'$ the colour c_{uv} . (2) If u is adjacent to safe vertex $u' \in V(D)$, and for any two neighbours $v'_1, v'_2 \in V(D)$ of v (if v has at least two neighbours in $V(D)$), we add pseudo-edge $v'_1v'_2$ to \tilde{E} . We assign $v'_1v'_2$ the colours c_{uv} (note v is not adjacent to a safe vertex since $u, v \notin K_{2,2}$). (3) If u is safe, then for every distinct pair of edges uv_1 and uv_2 , we add pseudo-edge v_1v_2 to \tilde{E} . Assign v_1v_2 the colour c_{uv} .

Again, a pseudo-edge xy with colour c_{uv} corresponds to a path from x to y created by a minimum selection of the edges incident on x and y of a feasible FVC solution.

Notice that with $(V(D), \tilde{E})$ (along with the corresponding edge colours we provide) describe an instance of the Maximum Rainbow Connection problem. We use Lemma 23 to compute a solution to the Maximum Rainbow Connection problem, which we denote by P . Say that P has α many components, and α_{large} many components with at least two vertices. Then we use the following three algorithms one by one to obtain a feasible solution. We have one last tool we need to provide before we can define for the next step of our algorithm, which is inspired by techniques employed in [3]. This tool will be useful for letting us decide which edges of \tilde{E} to buy.

► **Definition 24** (Good Cycles). *Let $\Pi = (V_1, \dots, V_k)$, $k \geq 2$, be a partition of the vertex-set of a 2VC simple graph G .*

A good cycle C of Π is a subset of edges with endpoints in distinct subsets of Π such that: (1) C is a cycle of length at least 2 in the graph obtained from G by contracting each V_i into a single vertex one by one (that is, $G/V_1/V_2/\dots/V_k$); (2) given any two edges of C incident to some V_i , these edges are incident to distinct vertices of V_i unless $|V_i| = 1$; (3) C has an edge incident to at least one V_i with $|V_i| \geq 2$, and; (4) $|C| = 2$ only if both V_i and V_j incident to C have $|V_i|, |V_j| \geq 2$.

The following Lemma allows us to compute good cycles in polynomial time. Its proof can be found in the full version of this paper.

► **Lemma 25.** *Let $\Pi = \{V_1, \dots, V_k\}$, $k \geq 2$, be a partition of the vertex-set of a 2VC simple graph G with the following conditions: $G[V_i]$ is connected for all $i = 1, \dots, k$, with at least one set of size at least 2. Furthermore, if there is exactly one $V_i \in \Pi$ with $|V_i| \geq 2$, then there are at least two singletons in Π that are adjacent in G .*

Then in polynomial time, one can compute a good cycle C of Π .

In our algorithm we will distinguish between connected components that have one vertex (i.e. singletons) and connected components with at least two vertices. Given a graph H , we say that a connected component is *large* if it has at least two vertices. Now we have all the ingredients required to finalize the description of our approximation algorithm. After computation of P , we initialize the set that will eventually be our solution as $APX_2 \leftarrow \emptyset$.

Our plan is to gradually build APX_2 . We have three steps, Algorithm 1, 2 and 3, which we apply one after another. These algorithms return edge sets S_1 , S_2 and S_3 , respectively that will be part of our solution APX_2 . We now take time to describe these algorithms in more details.

First we use Algorithm 1, which takes the pseudo-edges P , and find the large components and a subset X_1 of singletons of $(V(D), P)$, buying a subset of edges, S_1 , to connect them into a single component A in $(V(D), S_1 \cup P)$ using Lemma 25. Upon termination of this algorithm, we will obtain the additional property that $V(D) \setminus V(A)$ is an independent set in G .

Algorithm 1 Buying Good Cycles.

```

1 Input: pseudo-edges  $P$ 
2 Let  $Y$  denote the singletons of  $(V(D), P)$ 
3  $S_1 \leftarrow \emptyset$ 
4 while There is a good cycle  $C$  in  $G[V(D)]$  with connected components of
    $(V(D), P \cup S_1)$  being the vertex partitioning do
5    $S_1 \leftarrow S_1 \cup E(C)$ 
6  $A \leftarrow$  unique large component of  $(V(D), S_1 \cup P)$ 
7 Let  $X_1 \leftarrow Y \cap V(A)$ 
8 Return  $(X_1, Y, S_1, A)$ 

```

In Algorithm 2, our goal is to buy a minimal set S_2 of edges between $V(A)$ and compute a subset of $V(D) \setminus V(A)$ (which we denote by X_2), such that $(V(A) \cup X_2, P \cup S_1 \cup S_2)$ has one block. We remark that after termination of this algorithm any vertex of $X_3 := V(D) \setminus V(A) \cup X_2$ is a singleton in $(V(D), P \cup S_1 \cup S_2)$

Algorithm 2 Making Large Component 2VC.

```

1 Input: Edges  $S_1 \subseteq E$ , pseudo-edges  $P$ , large component  $A$ 
2  $X_2, S_2 \leftarrow \emptyset$ 
3 while  $G[V(A) \cup X_2] \cup P$  has more than one block do
4   Find  $v \in V(D) \setminus (V(A) \cup X_2)$  such that  $G[V(A) \cup X_2 \cup \{v\}] \cup P$  has fewer blocks
   than  $G[V(A) \cup X_2] \cup P$ 
5   Find edges  $e_1 = vu, e_2 = vw, v \neq w \in V(A)$ , such that
    $(V(A) \cup X_2 \cup \{v\}, P \cup S_1 \cup S_2 \cup \{e_1, e_2\})$  has fewer blocks than
    $(V(A) \cup X_2, P \cup S_1 \cup S_2)$ 
6    $X_2 \leftarrow X_2 \cup \{v\}$ 
7    $S_2 \leftarrow S_2 \cup \{e_1, e_2\}$ 
8 while There exists an edge  $e_3 = uz \in E \setminus S$  such that  $(V(A) \cup X_2, P \cup S_1 \cup S_2 \cup \{e_3\})$ 
   has fewer blocks than  $(V(A) \cup X_2, P \cup S_1 \cup S_2)$  do
9    $S_2 \leftarrow S_2 \cup \{e_3\}$ 
10 Return  $(X_2, S_2)$ 

```

In Algorithm 3, the goal is to buy a subset S_3 of edges such that $(V(D), S_1 \cup S_2 \cup S_3 \cup P)$ is a feasible FVC solution on $V(D)$ (i.e. $(V(D), S_1 \cup S_2 \cup S_3 \cup P)$ is connected and has no unsafe cut-vertices). For every vertex in $v \in X_3 := V(D) \setminus (V(A) \cup X_2)$ that has a safe neighbour in $V(A) \cup X_2$, we buy one edge from v to one of its safe neighbour in $V(A) \cup X_2$. For any other vertex in X_3 we buy two distinct edges from it to $V(A) \cup X_2$. We define α'_1 as the number of vertices of X_3 that have a safe neighbour, and α'_2 is the number of vertices of X_3 that do not have a safe neighbour. Thus $|X_3| = \alpha'_1 + \alpha'_2$. To maintain a consistent notation for number of components we define $\alpha' := |X_3| = \alpha'_1 + \alpha'_2$. Note that this implies, $\alpha = \alpha_{large} + |X_1| + |X_2| + |X_3| = \alpha_{large} + |X_1| + |X_2| + \alpha'$.

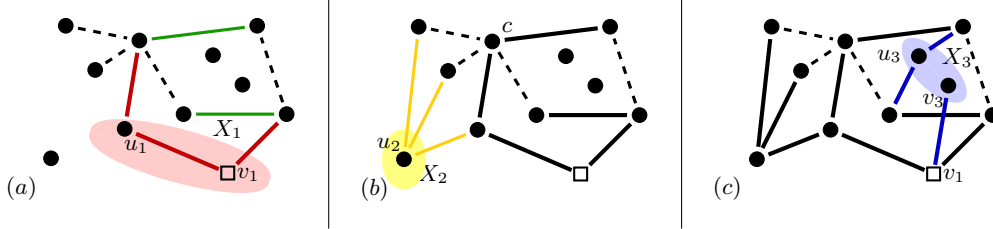
We finalize our solution for the instance by computing S_P , edges with endpoints in $K_{1,1}, K_{1,2}, K_{2,2}$, and $K_{2,3}$, by considering each pseudo-edge $\tilde{e} = v_1v_2 \in P$. If \tilde{e} has colour c_u , then $u \in K_{1,2}$. We add edges uv_1 and uv_2 to S_P .

■ **Algorithm 3** Making Solution Feasible.

```

1 Input: Singletons  $X_2$ , large component  $A$ 
2  $S_3 \leftarrow \emptyset$ 
3  $X_3 \leftarrow V(D) \setminus (V(A) \cup X_2)$ 
4  $\alpha'_1 \leftarrow 0, \alpha'_2 \leftarrow 0$ 
5 for every  $v \in X_3$  do
6   if  $v$  is adjacent to safe vertex  $u \in V(A) \cup X_2$  then
7      $S_3 \leftarrow S_3 \cup \{uv\}, \alpha'_1 \leftarrow \alpha'_1 + 1$ 
8   else
9     Find  $u, w \in V(A) \cup X_2$  adjacent to  $v$ 
10     $S_3 \leftarrow S_3 \cup \{uv, uw\}, \alpha'_2 \leftarrow \alpha'_2 + 1$ 
11 Return  $(X_3, S_3, \alpha'_1, \alpha'_2)$ 

```



■ **Figure 1** A depiction of the edges and vertex sets found by Algorithms 1, 2, and 3 in $V(D)$. Here the unsafe vertices are depicted by black circles. In this example there is only one safe vertex, v_1 in the set $V(D)$ that is shown by a square.

(a) The dashed edges are pseudo-edges P found by Lemma 23. Algorithm 1 first computes good cycle on green edges that merges two large components of pseudo-edges, then it finds the red cycle that merges the new large component and 2 singletons. $X_1 = \{u_1, v_1\}$ (b) The yellow edges of the second figure are found by Algorithm 2 which cover the cut-vertex c in the component. The interior vertex is $X_2 = \{u_2\}$. (c) The blue edges of the third figure are the edges found by Algorithm 3, which add edges to the solution that bring u_3 and v_3 into $V(D)$ form a feasible FVC solution. The vertex v_1 is a safe vertex so we only add one edge (x_3v_1) incident on v_3 .

If \tilde{e} has colour c_{uv} , then $u, v \in K_{2,3}$. We add edges to S_P in exactly one of the following ways (breaking ties in an arbitrary but fixed manner): (1) we add uv_1, vv_2 and uv to S_P if $uv_1, vv_2 \in E$; (2) we add uv_2, vv_1 and uv to S_P if $uv_2, vv_1 \in E$; (3) we add uv_1, uv_2 , and uv to S_P if u is safe, and $uv_1, uv_2, uv \in E$; (4) we add vv_1, vv_2 , and uv to S_P if v is safe, and $vv_1, vv_2, uv \in E$; (5) we add vv_1, vv_2 , and uu_1 to S_P if there exist a safe vertex $u_1 \in V(D)$ such that $uu_1 \in E$, and $vv_1, vv_2 \in E$, and; (6) we add uv_1, uv_2 , and vu_1 to S_P if there exist a safe vertex $u_1 \in V(D)$ such that $vu_1 \in E$, and $uv_1, uv_2 \in E$.

For every $v \in K_{1,1}$, buy an edge $uv \in E$ where $u \in V(D)$ is safe. By definition of $K_{1,1}$, u exists. Also, for every $uv \in E(K_{2,2})$, if u and v are both adjacent to safe vertices u' and v' in $V(D)$, then we buy the edges uu' and vv' . If at least one of u and v , (wlog say u) is not adjacent to a safe vertex in $V(D)$, then by definition of $K_{1,2}$, v must be safe, and v must be adjacent to a safe vertex $v' \in V(D)$ in G . In this case we buy edges vv', uv . Observe that by construction, $|S_P| = |K_{1,1}| + 2|K_{1,2}| + 2|K_{2,2}| + \frac{3}{2}|K_{2,3}|$. The output of our algorithm is $APX_2 := S_P \cup S_1 \cup S_2 \cup S_3$. The following Lemma shows that APX_2 is feasible, can be computed in polynomial time, as well as an upper bound on APX_2 . Its proof can be found in the full version of this paper.

► **Lemma 26.** *Our algorithm computes a feasible solution $APX_2 = S_P \cup S_1 \cup S_2 \cup S_3$, in polynomial time and $|APX_2| \leq |V(D)| - 1 + |S_P| + \alpha - 1 - (\frac{\alpha - \alpha'}{2} + \frac{\alpha_{large}}{2} + \alpha'_1)$.*

2.3 Approximation Factor

We fix an optimal solution, OPT , for the instance $G = (V, E)$. The following Lemma, proven in the full version of this paper, finds a set of lower bounds on $|OPT|$ that depend on terms found by our algorithm, in particular $K_{1,1}, K_{1,2}, K_{2,2}, K_{2,3}, \alpha, \alpha_{large}, \alpha'_1$, and α'_2 . We Recall that $|S_P| = |K_{1,1}| + 2|K_{1,2}| + 2|K_{2,2}| + \frac{3}{2}|K_{2,3}|$.

► **Lemma 27.** $|OPT| \geq \max\{|S_P| + \alpha - 1, 2K_{1,2} - 2\alpha_{large} + \alpha'_1 + 2\alpha'_2, n\}$.

With Lemma 27, Lemma 19, and Lemma 26 we have the tools necessary to prove Theorem 4.

Proof of Theorem 4. Given instance of (1,1)-FVC, $G = (V_S \cup V_U, E)$. We apply Lemma 13 to assume without loss of generality that G does not contain any forbidden cycles and G is 2VC. We first find solution APX_1 , and vertex sets $D, K_{1,1}, K_{1,2}, K_{2,2}$, and $K_{2,3}$. By Lemma 18, APX_1 is a feasible solution that we obtain in polynomial time. By Lemma 19, we have $|APX_1| \leq \frac{4}{3}(|V(D)| - 1) + |K_{1,1}| + 2|K_{1,2}| + |K_{2,2}| + \frac{3}{2}|K_{2,3}| = \frac{4}{3}(|V(D)| - 1) + |S_P|$.

Using sets $V(D), K_{1,1}, K_{1,2}, K_{2,2}$, and $K_{2,3}$ we apply Lemma 23 to compute a set of pseudo-edges P on $V(D)$ with α many components and α_{large} many large components (at least 2 vertices). We then apply Algorithms 1, 2, and 3 as described in Section 2.2 to compute edge sets S_1, S_2 , and S_3 , as well as α'_1 and α'_2 , where $\alpha' = \alpha'_1 + \alpha'_2$. We then find edge set S_P by replacing pseudo-edges with corresponding edges, and let $APX_2 = S_P \cup S_1 \cup S_2 \cup S_3$. By Lemma 26 computing APX_2 in this way takes polynomial time and $|APX_2| \leq |V(D)| - 1 + |S_P| + \alpha - 1 - (\frac{\alpha - \alpha'}{2} + \frac{\alpha_{large}}{2} + \alpha'_1)$.

By Lemma 27, we have $|OPT| \geq \max\{|S_P| + \alpha - 1, 2K_{1,2} - 2\alpha_{large} + \alpha'_1 + 2\alpha'_2, n\}$. Therefore, we have $\frac{\min\{|APX_1|, |APX_2|\}}{|OPT|}$ is at most:

$$\frac{\min\{\frac{4}{3}(|V(D)| - 1) + |S_P|, |V(D)| - 2 + |S_P| + \alpha - (\frac{\alpha - \alpha'}{2} + \frac{\alpha_{large}}{2} + \alpha'_1)\}}{\max\{|S_P| + \alpha - 1, 2K_{1,2} - 2\alpha_{large} + \alpha'_1 + 2\alpha'_2, n\}} \leq \frac{11}{7}.$$

Where the last inequality will be proven in the full version of this paper. ◀

3 FGC Improvement

The goal of this section is to prove Theorem 3. We use the algorithm described by [8], and provide a tighter analysis. In particular, their algorithm combines two different algorithms, and takes the best solution output among the two. Our improvement is based on a better analysis of the second one.

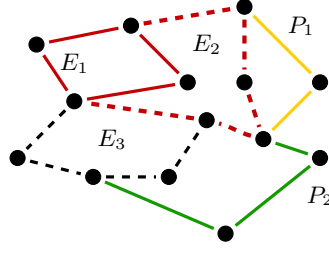
The lemma below comes from [8], in particular using their first algorithm. The authors prove the following (see Claim 6.4 of [8]).

► **Lemma 28** ([8]). *Given a FGC instance with optimal solution OPT , one can compute in polynomial time a solution F_1 with $|F_1| \leq |OPT \cap E_S| + \frac{3}{2}|OPT \cap E_U|$.*

The second algorithm used in [8] needs to be described fully, as we modify its analysis.

Algorithm 2. Given a FGC instance defined on a graph G , consider the graph G'' obtained from G by duplicating every safe edge in E . Run a β -approximation algorithm for the 2ECSS problem on G'' , and let F_2 be the output. Drop extra copies of safe edges from F_2 .

The authors prove the following claim (see Claim 6.5 of [8]).



■ **Figure 2** Here we have an example of a “nice” ear decomposition, consisting of ears E_1, E_2, E_3, P_1 and P_2 , each represented with a different colour or edge shape. Note that the only short ears P_1 and P_2 are open and “pendant”. That is, the internal vertices are *only* on their respective ears.

▶ **Lemma 29** ([8]). *We have $|E_2| \leq 2\beta|OPT \cap E_S| + \beta|OPT \cap E_U|$.*

In order to improve this algorithm we show that one can find a better approximation for 2ECSS if the size of the optimal solution is far enough from n , the number of vertices in G :

▶ **Lemma 30.** *Let G be a 2ECSS instance. One can find in polynomial time a solution APX of size $\frac{2}{3}n + \frac{2}{3}|OPT| - \frac{2}{3}$.*

Instead of proving Lemma 30, we here prove the following lemma, which assumes the instance is 2VC. Lemma 30 is proved the full version of this paper by considering the blocks of G .

▶ **Lemma 31.** *Let G be a 2ECSS instance that is 2VC. One can find in polynomial time a solution APX of size $\frac{2}{3}n + \frac{2}{3}|OPT| - \frac{2}{3}$.*

Proof. We provide a refined analysis of the algorithm provided in [20].

The authors in [20] first find what they call a “nice” ear decomposition. To define it, let’s introduce some terminology. The minimum number of ears of 1 one are called trivial, while ears of length 2 and 3 are called short. A vertex is pendant if it is not an endpoint of any non-trivial ear, and an ear is pendant if it is non-trivial and all its internal vertices are pendant. A nice ear decomposition is an ear decomposition with minimum number of even ears, in which there are no trivial ears, all short ears are pendant, and internal vertices of distinct short ears are non-adjacent. See Figure 2 for an example of a nice ear decomposition, and a clarification of short ears. Now consider the nice ear decomposition found in [20] and let π denote the number of short ears, π_i denote the number of ears of length i , and $\phi(G)$ denote the number of even length ears.

The authors define two algorithms and take the minimum output of the two. The first algorithm (see Section 5.3 of [20]) outputs a solution ALG_1 which satisfies $|ALG_1| \leq \frac{3}{2}|OPT| - \pi$. The second algorithm simply returns as a solution a nice ear decomposition (which they show exists for a 2VC graph, and can be computed efficiently). Let us call this solution ALG_2 . We now provide a bound on the size of ALG_2 .

$$\begin{aligned}
 |ALG_2| &= \sum_{i \geq 2} i\pi_i = 2\pi_2 + 3\pi_3 + 4\pi_4 + \sum_{i \geq 5} i\pi_i \\
 &\leq \left(\frac{5}{4} + \frac{3}{4}\right)\pi_2 + \left(\frac{5}{4} \cdot 2 + \frac{1}{2}\right)\pi_3 + \left(\frac{5}{4} \cdot 3 + \frac{1}{4}\right)\pi_4 + \sum_{i \geq 5} \frac{5}{4}(i-1)\pi_i \\
 &\leq \frac{5}{4}(n-1) + \frac{3}{4}\pi_2 + \frac{1}{2}\pi_3 + \frac{1}{4}\pi_4 = \frac{5}{4}(n-1) + \frac{1}{4}(\pi_2 + \pi_4) + \frac{1}{2}(\pi_3 + \pi_2) \\
 &\leq \frac{5}{4}(n-1) + \frac{1}{4}\phi(G) + \frac{1}{2}\pi.
 \end{aligned}$$

The second to last inequality follows since an ear of length i has $i - 1$ internal vertices, and every vertex of the graph but 1 is an internal vertex of exactly one ear. So the algorithm in [20] returns a solution of size $\leq \min\{|ALG_1|, |ALG_2|\} \leq \frac{1}{3}|ALG_1| + \frac{2}{3}|ALG_2|$ which is

$$\leq \frac{1}{3} \left(\frac{3}{2}OPT - \pi \right) + \frac{2}{3} \left(\frac{5}{4}(n-1) + \frac{1}{4}\phi(G) + \frac{1}{2}\pi \right) = \frac{1}{2}OPT + \frac{5}{6}(n-1) + \frac{1}{6}\phi(G).$$

To prove our claim it is enough to show that the latter term is bounded by $\frac{4}{3}n + \frac{2}{3}(x-1)$. For this, we need to employ Theorem 5 of [20] which states that $n + \phi(G) - 1 \leq |OPT|$, and hence $\phi(G) - 1 \leq x$. We then get

$$\begin{aligned} & \frac{4}{3}n + \frac{2}{3}(x-1) - \left(\frac{1}{2}OPT + \frac{5}{6}(n-1) + \frac{1}{6}\phi(G) \right) \\ &= \frac{4}{3}n + \frac{2}{3}(x-1) - \left(\frac{4}{3}n + \frac{1}{2}x + \frac{1}{6}\phi(G) - \frac{5}{6} \right) = \frac{1}{6}(x - \phi(G) + 1) \geq 0. \quad \blacktriangleleft \end{aligned}$$

We are now ready to prove Theorem 3.

Proof of Theorem 3. By Lemma 28, we have that $ALG_1 \leq |OPT \cap E_S| + \frac{3}{2}|OPT \cap E_U|$.

Let $opt(G'')$ be the size of an optimal solution for the 2ECSS instance G'' considered by Algorithm 2. It is important to observe that $opt(G'') \leq 2|OPT \cap E_S| + |OPT \cap E_U|$. Let us denote $OPT_S := OPT \cap E_S$ and $OPT_U := OPT \cap E_U$. Using Lemma 30, we can see that

$$\begin{aligned} |ALG_2| &\leq \frac{4}{3}n + \frac{2}{3}(opt(G'') - n - 1) \leq \frac{4}{3}n + \frac{2}{3}(2|OPT_S| + |OPT_U| - n - 1) \\ &= \frac{2}{3}(n-1) + \frac{4}{3}|OPT_S| + \frac{2}{3}|OPT_U| \leq \left(\frac{4}{3} + \frac{2}{3} \right) |OPT_S| + \frac{4}{3}|OPT_U|. \end{aligned}$$

Thus our algorithm provides a solutions of size at most $\min\{|ALG_1|, |ALG_2|\}$ which is

$$\begin{aligned} &\leq \min \left\{ |OPT_S| + \frac{3}{2}|OPT_U|, \left(\frac{4}{3} + \frac{2}{3} \right) |OPT_S| + \frac{4}{3}|OPT_U| \right\} \\ &\leq \frac{3}{7}(2|OPT_S| + \frac{4}{3}|OPT_U|) + \frac{4}{7}(|OPT_S| + \frac{3}{2}|OPT_U|) = \frac{10}{7}|OPT_S| + \frac{10}{7}|OPT_U|. \quad \blacktriangleleft \end{aligned}$$

References

- 1 David Adjiashvili, Felix Hommelsheim, and Moritz Mühlenthaler. Flexible graph connectivity: Approximating network design problems between 1-and 2-connectivity. *Mathematical Programming*, 192(1-2):409–441, 2022.
- 2 David Adjiashvili, Felix Hommelsheim, Moritz Mühlenthaler, and Oliver Schaudt. Fault-Tolerant Edge-Disjoint s-t Paths - Beyond Uniform Faults. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*, volume 227 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.SWAT.2022.5.
- 3 Afrouz Jabal Ameli, Fabrizio Grandoni, and Miguel Calvo-Bosch. A 4/3 approximation for 2-vertex-connectivity. In *ICALP*, 2023.
- 4 Ishan Bansal. A constant factor approximation for the $(p, 3)$ -flexible graph connectivity problem. *arXiv preprint arXiv:2308.15714*, 2023.
- 5 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Extensions of the (p, q) -flexible-graph-connectivity model. *arXiv preprint arXiv:2211.09747*, 2022.

- 6 Ishan Bansal, Joseph Cheriyan, Logan Grout, and Sharat Ibrahimpur. Improved approximation algorithms by generalizing the primal-dual method beyond uncrossable functions. In *50th International Colloquium on Automata, Languages, and Programming (ICALP 2023)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.
- 7 Matthias Bentert, Jannik Schestag, and Frank Sommer. On the complexity of finding a sparse connected spanning subgraph in a non-uniform failure model. *arXiv preprint arXiv:2308.04575*, 2023.
- 8 Sylvia Boyd, Joseph Cheriyan, Arash Haddadan, and Sharat Ibrahimpur. Approximation algorithms for flexible graph connectivity. *Mathematical Programming*, pages 1–24, 2023.
- 9 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.
- 10 Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k -connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000.
- 11 Artur Czumaj and Andrzej Lingas. On approximability of the minimum-cost k -connected spanning subgraph problem. In *SODA*, volume 99, pages 281–290. Citeseer, 1999.
- 12 Harold N Gabow, Michel X Goemans, Éva Tardos, and David P Williamson. Approximating the smallest k -edge connected spanning subgraph by lp-rounding. *Networks: An International Journal*, 53(4):345–357, 2009.
- 13 Mohit Garg, Fabrizio Grandoni, and Afrouz Jabal Ameli. Improved approximation for two-edge-connectivity. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2368–2410. SIAM, 2023.
- 14 Dylan Hyatt-Denesik, Afrouz Jabal Ameli, and Laura Sanita. Improved approximations for flexible network design, 2024. [arXiv:2404.08972](https://arxiv.org/abs/2404.08972).
- 15 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001.
- 16 Samir Khuller and Balaji Raghavachari. Improved approximation algorithms for uniform connectivity problems. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 1–10, 1995.
- 17 Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994. doi:10.1145/174652.174654.
- 18 Yusuke Kobayashi and Takashi Noguchi. An approximation algorithm for two-edge-connected subgraph problem via triangle-free two-edge-cover. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation, ISAAC 2023, December 3-6, 2023, Kyoto, Japan*, volume 283 of *LIPICs*, pages 49:1–49:10. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.ISAAC.2023.49.
- 19 Zeev Nutov. Improved approximation algorithms for some capacitated k edge connectivity problems. *arXiv preprint arXiv:2307.01650*, 2023.
- 20 András Sebő and Jens Vygen. Shorter tours by nicer ears: $7/5$ -approximation for the graph-tsp, $3/2$ for the path version, and $4/3$ for two-edge-connected subgraphs. *Combinatorica*, pages 1–34, 2014.
- 21 Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.