Approximation Algorithms for Survivable Network Design

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Afrouz Jabal Ameli

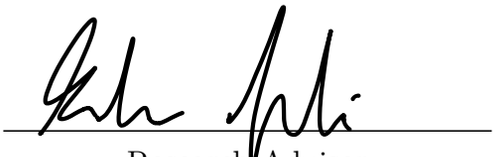
under the supervision of
Prof. Fabrizio Grandoni

April 2021

Dissertation Committee

Prof. Jarosław Byrka	University of Wrocław
Prof. Antonio Carzaniga	Università della Svizzera Italiana, Switzerland
Prof. Stefano Leonardi	Università "La Sapienza" Roma, Italy
Prof. Stefan Wolf	Università della Svizzera Italiana, Switzerland

Dissertation accepted on 19 April 2021



Research Advisor

Prof. Fabrizio Grandoni

PhD Program Director

Prof. Walter Binder / Prof. Silvia Santini

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Afrouz Jabal Ameli
Lugano, 19 April 2021

To my family

In the name of God.

Abstract

Many relevant discrete optimization problems are believed to be *hard to solve efficiently* (i.e. they cannot be solved in polynomial time unless P=NP). An *approximation algorithm* is one of the ways to tackle these hard optimization problems. These algorithms have polynomial running time and compute a feasible solution whose value is within a proven factor (*approximation factor*) of the optimal solution value. The field of approximation algorithms has grown quickly over the last few decades, leading to the development of several algorithmic and analytical techniques.

In this doctoral dissertation, we focus on *Survivable Network Design* problems, where the goal is to construct low-cost networks that are resilient to a few edge/node faults. More specifically, we consider a basic problem in this area, that is the Connectivity Augmentation problem (CAP). In this problem, we are given a k-edge-connected graph (namely, a graph in which removing any $k-1$ edges preserves the connectivity of the graph) and a collection of extra edges (*links*). Our goal is to identify a minimum cardinality subset of links whose addition to the graph makes it $(k+1)$-edge connected. This problem is NP-hard and has many interesting real-world applications; For this reason it has been studied through the lens of approximation algorithms in the past.

Despite the efforts of several researchers, no progress was made on this problem after the 2-approximation algorithm by Frederickson and JáJá [1981]. We remark that a 2 approximation is known even for wide generalizations of CAP. The main contribution of this thesis is breaching the 2 approximation barrier for CAP by presenting a 1.91 approximation algorithm. Our result is based on a non-trivial reduction to another fundamental problem, Steiner Tree. Along the way to this main achievement, we studied a special case of CAP, the Cycle Augmentation problem (CycAP) for which 2 was the best-known approximation factor. Here we are given a cycle plus additional links, and the goal is to find a subset of links with minimum size whose addition to G makes it 3-edge-connected. We show that CycAP is APX-hard, in particular it does not admit an approximation factor arbitrarily close to 1 (even the NP-

hardness of this problem was not known earlier). Furthermore, we present a $3/2 + \epsilon$ approximation algorithm for any constant $\epsilon > 0$.

Acknowledgements

.

I want to express my gratitude to my advisor, Fabrizio Grandoni. It has been a real privilege being a student of Fabrizio. His profound knowledge of approximation algorithms and combinatorial optimization is combined with inquisitive curiosity and a generous attitude. Besides several intriguing ideas that motivated all results of the thesis substantially, I could learn the importance of a deeper understanding of problems from him. Also, his uncompromising aesthetic standards concerning the clarity and simplicity of proofs and presentation have had a great impact on me. His constant search for intuitive and simple explanations, together with his thirst for knowledge and attention to detail, have set a role-model for me.

A Special thanks to Mohammad Pishnamaz, Mohammad Ali Aabaam, Alireza Alipour, Farhad Shahmohammadi and all the tutors of the Iranian Olympiad in Informatics, who initiated my interest in algorithms and discrete mathematics.

I am extremely grateful to my wife, because of her support during these years. We both suffered from being separated due to visa issues.

I am thankful to my family and friends from Iran, who constantly provided emotional support and encouragement to accomplish my projects, despite the physical distance separating us. Thanks a lot in particular to my mother Mahnaz, my father and my brother.

Thank you to all my colleagues and coauthors. It was a pleasure and honor to collaborate and share enjoyable time: Waldo Gàlvez, Kamyar Khodamoradi and Mohit Garg among others.

Special thanks to all the friends I met at IDSIA and USI, specially to my teachers and classmates at the sport groups. I would also like to thank Daniela, Cinzia and Elisa who helped me a lot to settle down in Lugano.

To all of you, my sincere gratitude.

Contents

Chapter 1

Introduction

Many of the problems that we need to solve in practice are discrete optimization problems. In these problems we need to identify, among a discrete set of feasible solution, a solution which optimizes a given objective function. This captures several applications as for example deciding inventory levels, assigning tasks, organizing data for efficient retrieval etc. Unfortunately, many natural discrete optimization problems are difficult to solve efficiently (i.e., in polynomial time in the input size). In particular, assuming that $P \neq NP$, it is impossible to provide an algorithm that is efficient and at the same time computes the optimal solution. Yet, for several reasons these problems require more progress; (1) First of all, these problems have industrial applications that can not be ignored. (2) Researchers are curious to know what can be achieved, if an efficient exact solution is impossible. (3) Also the recent advances in hardware systems inspire that someday even semi-efficient algorithm can have better performances. Thus, an extensive research has been devoted to techniques to deal with NP-hard problem while relaxing either optimality or efficiency. This has lead to the birth of several sub-fields of algorithms and computations.

One line of research, known as Parameterized algorithms, aims for algorithms whose running time depends in a super-polynomial way not on the entire input size, but only on the size of a proper part of the input called parameter.

Definition 1 (Cygan et al. [2015])**.** *A parameterized optimization problem $\mathscr{P}$ is called **Fixed-Parameter Tractable** (FPT) if there exists an algorithm $\mathscr{A}$, a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant c such that, given an instance (I, k), algorithm $\mathscr{A}$ correctly solves the problem in time bounded by $f(k) \cdot |I|^c$.*

Some commonly used parameters are the size of the solution, the dimension

for geometrical problems and the maximum degree or the treewidth of the input graph.

A different approach to attack these problems is to consider restricted instances. For instance, many graph optimization problems have been studied specifically for planar or sparse graphs. In many cases this approach led to improved algorithms for the general case or to a deeper understanding of hardness of the problem. Moreover, for many industrial application solving restricted instances is sufficient. For instance in most real-world applications the input graph is usually sparse.

Another approach is to use approximation algorithms (Knuth [1974]), which is our main focus in this doctoral dissertation. These are polynomial-time algorithms that compute feasible solutions of cost within a given factor (*approximation factor* or *approximation ratio*) from the value of the optimal solution.

1.1 Approximation algorithms

In this section, we introduce some fundamental definitions and concepts in the field of approximation algorithms.

Definition 2 (Knuth [1974])**.** *Given an optimization problem $\mathscr{P}$, an* α***-approximation algorithm*** *for $\mathscr{P}$ is an algorithm with polynomial running time that, for each instance of the problem, outputs a solution whose value is within a factor α of the value of an optimal solution.*

The **approximation ratio** $\alpha \geq 1$ of an approximation algorithm is defined as

$$\max_{I \in \mathscr{I}} \max \left\{ \frac{\mathrm{OPT}(I)}{APX(I)}, \frac{APX(I)}{\mathrm{OPT}(I)} \right\},$$

where $\mathscr{I}$ is the set of instances of the problem, $APX(I)$ is the value of the solution computed by the approximation algorithm when run on instance I and $\mathrm{OPT}(I)$ is the value of an optimal solution for instance I. Ideally, α is a constant (i.e not dependant on the size of the input) and in this case we say that our algorithm is a constant approximation. Indeed in terms of optimality, the goal is to minimize α. For an NP-hard optimization problem, the best kind of approximation algorithms one can hope to design are called *Polynomial Time Approximation Schemes* which we proceed to define:

Definition 3. *A* ***Polynomial Time Approximation Scheme*** *(PTAS) for an optimization problem $\mathscr{P}$ is a family of algorithms $\{\mathscr{A}_\varepsilon\}_{\varepsilon>0}$ such that, for*

each $\varepsilon > 0$, $\mathscr{A}_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm for $\mathscr{P}$. The running time of these algorithms is polynomial in the input size for any fixed ε.

The class of problems that admit a PTAS and the class of problems that admit constant factor approximation are called PTAS and APX, respectively. Clearly, all problems that admit a PTAS are in APX, but the converse is not true if $P \neq NP$.

1.2 Lower bounds and Approximation Preserving Reductions

For an NP-hard optimization problem P, the main challenge is to find the best possible approximation factor (α^*). Any α-approximation algorithm for P, by definition, provides an *upper bound* on the best possible approximation factor (i.e this shows that $\alpha^* \leq \alpha$). A significant amount of research has been devoted to a dual kind of result, that is, proofs that certain approximation ratios are not possible (under the assumption that $P \neq NP$ or analogous assumptions). These results provide a lower bound on the approximation factor.

Definition 4. *A problem A is called **APX-hard** if it does not admit a PTAS unless $P = NP$.*

Thus, being APX-hard is considered as a strong evidence that the problem does not admit a PTAS. For many problems, stronger lower bounds are known that rely only on the assumption that $P \neq NP$. For instance, a $(\frac{3}{2} - \epsilon)$-approximation algorithm of Bin Packing Problem for any $\epsilon > 0$, can be used to solve the Partition problem in polynomial time. In the case of the Steiner Tree problem, it is NP-hard to find a solution of size less than $\frac{96}{95}$ of the optimal solution. Thus the existence of a $(\frac{96}{95} - \epsilon)$-approximation for any $\epsilon > 0$ seems impossible (Chlebík and Chlebíková [2008]).

Definition 5. *We say that α is a lower bound on the approximability of A, if for any constant $\epsilon > 0$ a polynomial time $(\alpha - \epsilon)$-approximation is impossible for A, unless $P = NP$.*

There are many optimization problems that are hard to approximate even with a constant approximation ratio. For instance, the set cover problem can not be approximated within a factor $o(\log n)$, where n is the size of the universe.

From the approximation algorithms perspective, the ultimate goal for any NP-Hard optimization problem is to find an approximation algorithm for it plus a matching hardness of approximation result.
In order to investigate the inapproximability of a problem A, a standard technique is to use some special types of *approximation-preserving* reductions from other problems. More specifically, an approximation-preserving reduction from a problem A to a problem B shows that if there exists an α-approximation for B, we can then get an $f(\alpha)$-approximation for A, where f is some function. Then we know that if A is hard to approximate within some factor x, then B is hard to approximate within a factor $f^{-1}(x)$.

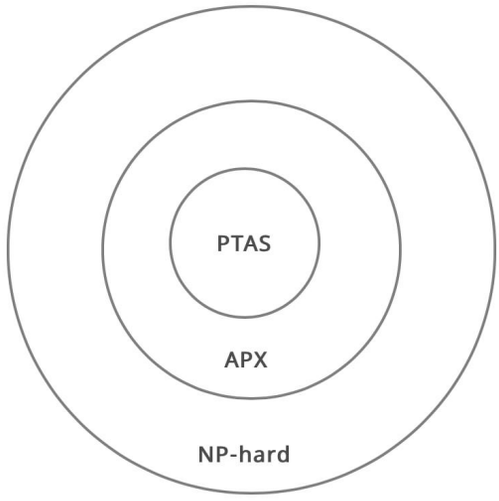


Figure 1.1. The diagram describing the relation of classification of NP-hard in terms of approximability.

There are several different types of approximation-preserving reductions. Here we define only those that are used in this thesis. Strict reduction is the simplest type of approximation-preserving reduction. In a strict reduction, $f(x) = x$ for any $x > 1$, and hence the approximation ratio obtained for problem A is at least as good as the approximation ratio of problem B. Strict reduction preserves membership in both PTAS and APX (See Lemma 14 for an example of strict reduction).

Another common type of approximation-preserving reduction are the S-reductions, for which the cost of the optimal solution for the corresponding

instances of A and B must have the same size. S-reduction is a very special case of strict reductions. In effect, the two problems A and B must be in near-perfect correspondence with each other. The existence of a S-reduction implies not only the existence of a strict reduction but every other approximation-preserving reduction.

In order to show that a problem B does not admit a PTAS using an APX-hard problem A, a special type of reductions known as L-reduction is required from A to B. with a L-reduction, an α-approximation for problem B is transformed to a $(1 + a(\alpha - 1))$-approximation for A, where $a > 0$ is a constant. Hence if A admits a PTAS, then so does B. (See section 3.3 for an example of such reductions).

For any NP-optimization the main challenge is to tighten the gap between the lower bound and upper bound of approximability. For many NP-hard problems this gap is already closed. For instance the Knapsack Problem is NP-hard and admits a PTAS. On the other hand there are many problems that still require substantial progress.

1.3 Survivable Network Design Problems

The goal of *Network Design* is to design cheap networks that support a given traffic. Among these problems, *Survivable Network Design* problems received a lot of attention in the last few years. The basic goal here is to construct low cost networks that preserve the connectivity between given sets of nodes despite the failure of a few edges/nodes (in the following we will focus on the edge failure case). This has many applications, e.g., in transportation and telecommunication networks. Several such problems are NP-hard, and in most cases the best known approximation factor is 2 due to Jain [2001].

An interesting family of survivable network design problems are the *Network Augmentation* problems. Here we are given an existing network plus a collection of extra edges (*links*) that can be added to the network at a given cost. Our goal is to identify a cheap subset of links whose addition to the network increases its connectivity in a desired manner. A well-studied classical problem in this area is the *Connectivity Augmentation* problem (CAP). Recall that a graph G is k-edge-connected (or k-connected for short), if G remains connected even after removing any subset of at most $k - 1$ edges. In CAP we are given a k-connected graph $G = (V, E)$, and an additional set L of edges (*links*). We need to find a subset L' of links such that, by adding them to G, the graph becomes $(k + 1)$-connected (i.e. $H = (V, E \cup L')$ is $(k + 1)$-connected). Our

goal is to minimize the cardinality of L'.

CAP has many applications in transportation systems and network services. For example in transportation systems, the cities and the roads can be viewed as a graph. Naturally roads might be blocked due to car accidents, reconstructions or natural events. However, it is essential that the transportation system guarantees the existence of paths between certain destinations even when a few roads are blocked. If the latter number is k, this means that the graph is k-connected. Increasing the value of k is possible by constructing new roads. However construction of extra roads are expensive, hence one wishes to increase the connectivity while minimizing the extra cost.

CAP and its special cases are among the best-studied survivable network design problems. Dinitz et al. [1976] (see also Cheriyan et al. [1999] and Khuller and Thurimella [1993]) presented an approximation-preserving reduction from this problem to the case $k = 1$ for odd k, and $k = 2$ for even k. This motivates a deeper understanding of the latter two special cases.

The case $k = 1$ is also known as the *Tree Augmentation* problem (TAP). The reason for this name is that any 2-connected component of the input graph G can be contracted, hence leading to an equivalent instance where the input graph is a tree. The case $k = 2$ is also known as the *Cactus Augmentation Problem* (CacAP), where for similar reasons we can assume that the input graph is a cactus[1].

TAP is a very well-studied problem. Cheriyan et al. [1999] showed that this problem is NP-hard, and later Kortsarz et al. [2004] showed that it is APX-hard. Several better than 2 approximation algorithms are known for TAP. The first algorithm beating the approximation guarantee of 2 is due to Nagamochi [2003], achieving an approximation factor of $1.815 + \epsilon$[2]. This factor was subsequently improved to 1.8 (Even et al. [2009]) and to 1.5 (Cheriyan et al. [2008]; Kortsarz and Nutov [2016b]). These results are combinatorial in nature, but LP-based results have been achieved as well. As an example, recently Nutov [2017] showed that the standard cut LP for TAP has an integrality gap of at most 28/15 while a lower bound of 3/2 was known Cheriyan et al. [2008]. An LP-based $\left(\frac{5}{3} + \epsilon\right)$-approximation was given by Adjiashvili [2017] and then refined by Fiorini et al. [2018] to obtain a $\left(\frac{3}{2} + \epsilon\right)$-approximation (see also Cheriyan and Gao [2018a]; Kortsarz and Nutov [2016a]). Both results are obtained by

[1]We recall that a *cactus* G is a connected undirected graph in which every edge belongs to exactly one cycle. For technical reasons it is convenient to allow length-2 cycles consisting of 2 parallel edges.

[2]Throughout this paper, when not specified otherwise, ε denotes an arbitrary but fixed positive constant.

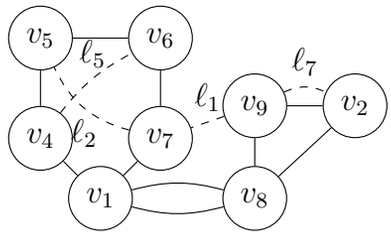


Figure 1.2. An instance of the Connectivity Augmentation Problem. The links are shown by the dashed edges.

adding a proper family of extra constraints to the standard cut LP. Recently, Grandoni et al. [2018] achieved a 1.458 approximation for TAP, which is smaller than the integrality gap of the standard cut LP.

No much progress was done specifically on CacAP until 2020. Here the best-known approximation factor was still 2, and this factor can be achieved with multiple approaches: Frederickson and JáJá [1981]; Goemans et al. [1994]; Jain [2001]; Khuller and Thurimella [1993]. Altogether this implied that the best known approximation factor for CAP is 2, i.e. the same factor that can be achieved with the much more general approach by Jain [2001]. In Byrka et al. [2020] we obtained the first approximation factor below 2 for CacAP by presenting a 1.91-approximation algorithm based on a method different from the recent advances for TAP.

Very recently, Cecchetto et al. [2021] made an amazing breakthrough regarding the approximability of CAP by achieving a 1.393-approximation. Notice that this even improves on the TAP approximation algorithm by Grandoni et al. [2018]. Interestingly the authors bridged the gap between TAP and CacAP by presenting techniques that allow for leveraging insights and methods from TAP to approach CAP.

1.4 Publications and Manuscripts

In this dissertation I will describe my results related to survivable network design. This led to the following two publications:

- **On the Cycle Augmentation Problem: Hardness and Approximation Algorithms** (Gálvez et al. [2021]). Joint work with Waldo Gálvez, Fabrizio Grandoni and Krzysztof Sornat. Published in *Journal of Computing Systems.* Preliminary version in WAOA 2019.

- **Breaching the 2-Approximation Barrier for Connectivity Augmentation: A Reduction to Steiner Tree** (Byrka et al. [2020]). Joint work with Jarosław Byrka and Fabrizio Grandoni. Published in *STOC 2020.*

During my Ph.D. I also worked on other problems which are not described in this thesis since they deviate too much from its main topic. In particular, I studied some geometric packing problems, where the general aim is to pack rectangles in a given region. In particular I worked on the Strip Packing problem and one of its variants. This led to the following publication and manuscript:

- **A Tight $\frac{3}{2} + \epsilon$ Approximation Algorithms for δ-skewed Strip Packing Problem.** (Gálvez et al. [2020]). Joint work with Waldo Gàlvez, Fabrizio Grandoni, Arindam Khan, Klaus Jansen and Malin Rau. Published in *APPROX/RANDOM 2020.*

- **On the Demand Strip Packing Problem.** Manuscript. Joint work with Waldo Gàlvez, Fabrizio Grandoni, and Kamyar Khodamoradi.

Finally, I worked on some problems independently from my research group at IDSIA. This led to the following publication and manuscript:

- **On the Triangle Enclosure Problem.** (Jabal Ameli et al. [2018]). Joint work with Hamid Zarrabi Zadeh. Published in *ICCG 2018.*

- **Chromatic Number and Dichromatic Polynomial of Digraphs.** Manuscript(Akbari et al. [2017]). Joint work with Saeed Akbari, Amir Ghodrati and Morteza Saghafian.

1.5 Our results and Outline of the Thesis

In Chapter 2 we provide the basic notation, definitions and basic results.

We focused on CacAP, with the aim of improving the approximation factor for this problem (hence for CAP). As a first step in this direction, we studied the *Cycle Augmentation* problem (CycAP), i.e. the special case of CacAP where the input cactus consists of a single cycle. We remark that this special case remains non-trivial. In Chapter 3 we describe our results on CycAP, namely presenting two approximation algorithms for the problem with approximation factor below 2, showing that this problem is APX-hard and presenting

a new LP-formulation for this problem with a better integrality gap than the previously known LP-formulations.

Our second, and most relevant result, is an improved 1.91 approximation for CacAP using a reduction to the Steiner Tree problem. This also implies the same approximation factor for CAP based on the mentioned reduction and the known results for TAP. We explain this result in Chapter 4.

Finally, in the Chapter 5 we conclude this thesis and discuss some open problems and future research directions. We defer to the Appendix some rather technical proofs and complementary results.

Chapter 2

Preliminaries

2.1 Edge-connectivity

In this section we discuss basic definitions and useful results related to edge-connectivity. Recall that a graph $G = (V, E)$ is k-connected if by removing any subset of size less than k from E, the graph remains connected.

A subset $E' \subseteq E$ of G is a called an edge-cut (or simply cut) if $H = (V, E \setminus E')$ is disconnected. Finding a cut with minimum size (or shortly a min-cut) is a well-studied classic problem. The size of the min-cut in a graph G is known as the *edge-connectivity* of the graph G and is denoted by $\kappa'(G)$ (i.e $\kappa'(G)$ is the smallest k such that G is k-connected). In many applications such as network systems having edge-connectivity up to some degree is a measure of the reliability and resilience of the system.

Observation 6. *Let G be a graph. Then G is k-edge-connected if and only if the size of the min-cut of G is at least k.*

Note that one can disconnect a graph G by removing all the edges incident to a node v of G, therefore:

Observation 7. *Let G be a k-connected graph, then for every vertex v of G,* $\deg_G(v) \geq k$.

Observation 7 suggest that for a graph G the minimum degree of the vertices of G ($\delta(G)$) is a lower bound on the edge-connectivity of G. This basic lower bound is very useful to lower bound the optimal solution in several connectivity related optimization problems. In fact, in Chapters 3 and 4 we are using this fact to analyse our approximation factor.

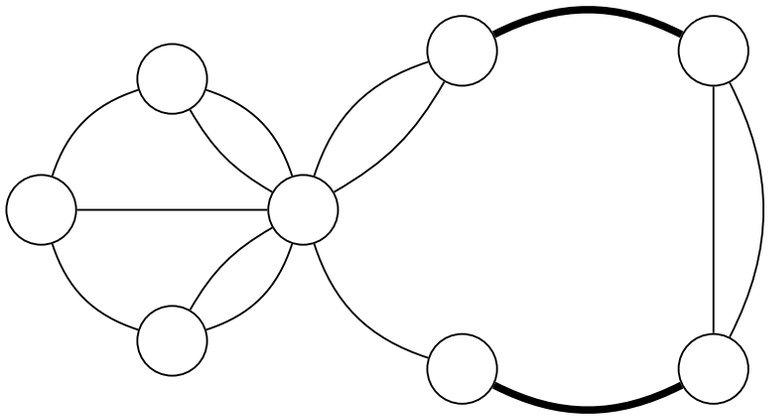


Figure 2.1. A graph G with edge-connectivity of 2. The bold black edges indicate an edge-cut of size 2.

An edge-cut E' is called a $s-t$ cut if there is no path from u to v in $H = (V, E \setminus E')$.

Many polynomial time algorithm exist for finding the min-cut of a graph and hence checking if a graph is k-edge connected can be done in polynomial time (see Cormen et al. [2009]).

Edge-connectivity is related to the number of edge-disjoint paths among pairs of vertices:

Theorem 8 (Menger's Theorem)**.** *The size of the minimum $s-t$ cut in a graph $G = (V, E)$ is k if and only if the maximum number of edge-disjoint paths from u to v is k.*

Corollary 9. *A graph $G = (V, E)$ is k-connected if and only if for every pair $\{u, v\} \subseteq V$, there exist k edge-disjoint paths from u to v.*

Note that similar definitions and theorems also hold for node-connectivity. To have a more detailed understanding in the subject of connectivity we refer the reader to Bondy and Murty [2008].

2.2 From CAP to TAP and CacAP

In this section we describe an insightful result regarding connectivity augmentation proposed by Dinitz et al. [1976], that influenced all the later works in this area.

In this section we sketch the ideas leading to this result. Consider an instance of CAP with input a k-connected graph G and a set L of links. Now consider the connected components formed by removing the min-cuts. Dinitz et al. [1976] showed that these components form a laminar family if k is odd. Furthermore, they showed that the minimum edge cuts of a graph G can be represented in polynomial time as the 2-edge-cuts of a graph G' such that $\kappa'(G) = 2$. This resulted in the following important theorem:

Theorem 10 (Dinitz et al. [1976])**.** *Any instance of the Connectivity Augmentation can be reduced in polynomial time to the case that $k = 1$ for odd k and to $k = 2$ for even k.*

Dinitz et al. showed that this is an S-reduction and this means that in terms of approximability we can restrict our attention only to the case that $k = 1$ or 2.

For the case that $k = 1$, if the input graph G contains a cycle one can reduce the size of the input G by contracting this cycle into a single node. By running this process repeatedly, the graph G becomes a tree. Hence:

Proposition 11. *There is a S-reduction from Connectivity Augmentation with $k = 1$ to TAP*

When $k = 2$, Dinitz et al. [1976] showed that one can contract a pair $\{u, v\}$ of vertices of G, such that there are at least three edge-disjoint paths from u to v (also known as 3-*edge-connected pair*). Note that this operation does not violate 2-edge-connectivity. After exhausting this process using the following lemma we show that the ultimate graph G^* is a cactus.

Lemma 12. *Let G be a 2-connected graph. Assume that for any pair of vertices of G such as $\{u, v\}$, there are at most two edge-disjoint paths from u to v. Then G is a cactus.*

Proof. Assume on the contrary that this is not true. Then, since G is 2-connected, any edge $e \in E(G)$ belongs to at least one cycle of G and since G is not a cactus there exists and edge $e_1 \in E(G)$ such that $e_1 = \{v_1, v_2\}$ belongs to more than one cycle. Let C_1 and C_2 be two distinct cycles of G that contain e_1. Now let $C_1 = v_1v_2...v_kv_{k+1}$ ($v_{k+1} = v_1$). Set $i > 1$, as the smallest index such that v_iv_{i+1}, is not an edge of C_2. Let v_j ($i < j \leq k + 1$) be the node that has the shortest path, P, to v_i in C_2. Note that since C_2 is a cycle that contains e_1 such a node must exist. Now P, $v_iv_{i+1}...v_j$ and $v_j...v_kv_1v_2....v_i$ are three edge-disjoint paths from v_i to v_j which is a contradiction. □

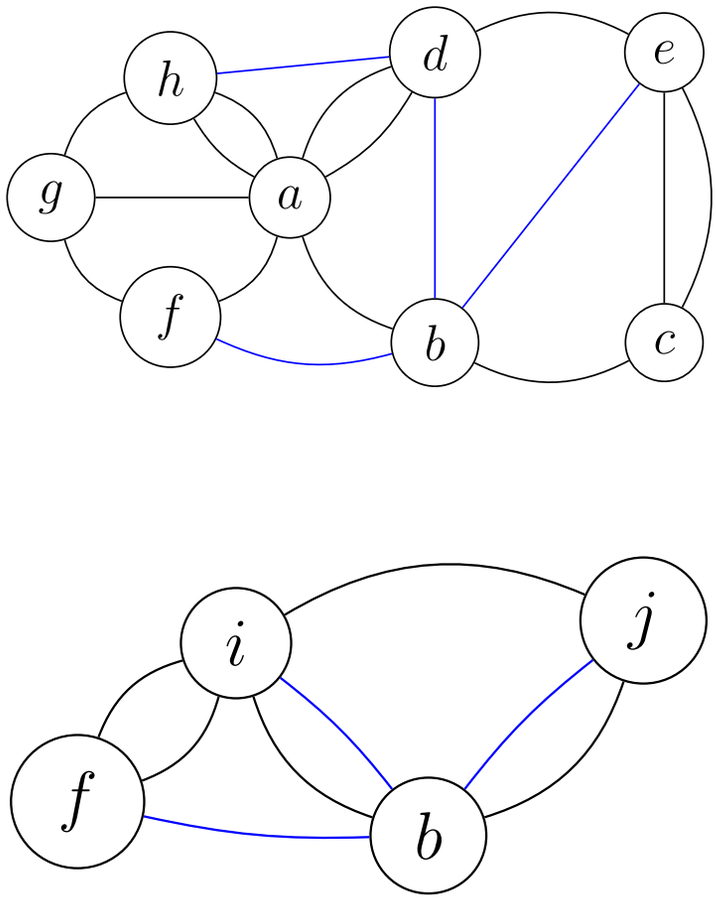


Figure 2.2. **Top**: An instance I of CAP with $k = 2$. **Bottom**: An instance J of CacAP obtained from I by contracting 3-edge-connected pairs of nodes. The node i is obtained from contracting g, h, a and d into one node and the node j is obtained from contracting c and e.

Proposition 13. *There is a S-reduction from Connectivity Augmentation with $k = 2$ to CacAP.*

The following lemma shows that CacAP is at least as hard to approximate as TAP:

Lemma 14. *There exists an strict reduction from TAP to CacAP.*

Proof. Consider an instance I of TAP with input tree $T = (V, E)$ and the set L of links. From this we obtain an instance of CacAP. Let $C = (V, E')$ be the cactus formed by duplicating the edges of T. Now assume J is the instance of CacAP formed by input cactus C and links L.
A critical observation is that the set of edge-cuts of size 2 in C are precisely the pair of edges $\{e, e'\}$ such that e' is the duplicate of e. This suggests that the components formed by removing the minimum edge-cuts in I and J are exactly the same. Therefore any subset $L' \subseteq L$ is a feasible solution for I if and only if it is a feasible solution for J. □

Using lemma 14 one can obtain the following statement which is a refined version of theorem 10:

Corollary 15. *There exists an S-reduction from CAP to CacAP.*

2.3 Related Problems

In this section we discuss some related Survivable Network Design problems. A very closely related problem to CAP is *the minimum spanning k-edge-connected subgraph (k-ECSS)*, which we proceed to define:

Definition 16 (Minimum Spanning K-edge-connected Subgraph Problem)**.** *In the Minimum Spanning K-edge-connected Subgraph Problem, given a graph G and an integer k, the goal is to find a k-edge-connected spanning subgraph of G with minimum number of edges.*

Note that when $k = 1$, the k-ECSS problem is equivalent to finding a spanning tree of the input graph G, hence we can solve this problem efficiently. However, this problem is NP-hard already for $k = 2$. Indeed, in this case the size of the optimal solution for this problem is equal to the number $|V(G)|$ of nodes if and only G has a Hamiltonian cycle. Moreover, it has been shown that this problem does not admit PTAS for any $k > 1$ even for subcubic[1] graphs.

[1] A graph G is subcubic if the degree of the vertices of G is at most three.

In terms of approximability 2-ECSS is a very well-studied problem. The first result beating the factor 2 is the $\frac{3}{2}$-approximation algorithm by Khuller and Vishkin [1994]. Cheriyan et al. [2001] improved the factor to $\frac{17}{12}$. A relatively recent paper by Sebö and Vygen [2014] provides a $\frac{4}{3}$-approximation algorithm given by using more elegant ear decompositions. Hunkenschröder et al. [2019] obtain the same approximation factor by via a drastically different approach.

Surprisingly as k grows, it is easier to approximate k-ECSS. In this line of research, an important result is by Gabow et al. [2009], in which using a LP rounding technique, authors achieved an approximation algorithm of factor $1 + \frac{2}{k}$ for even k and $1 + \frac{3}{k}$ for odd k.

A very interesting generalization of TAP and 2-ECSS is the *Forest Augmentation Problem* (FAP):

Definition 17 (Forest Augmentation Problem)**.** *In the Forest Augmentation Problem, as an input we are given a forest $G = (V, E)$ and a set of links L and the goal is to find a subset L' of L with minimum size such that $G = (V, E \cup L')$ is 2-connected.*

Note that in a point of view, 2-ECSS and TAP are the extreme cases of FAP, where in one problem the forest G contains no edge and in the other problem G has the maximum possible number of edges (i.e., it is a spanning tree). To the best of our knowledge, there is no approximation algorithm with approximation guarantee better than 2 for FAP. Another studied variant of FAP, is the *Matching Augmentation* problem (MAP), in which the forest G is simply a matching. MAP was first introduced in Cheriyan, Dippel, Grandoni, Khan and Narayan [2020], who present a $\frac{7}{4}$-approximation for this problem. The approximation factor was later improved to $\frac{5}{3}$ by Cheriyan, Cummings, Dippel and Zhu [2020].

All the mentioned problems are special cases of the following very general survivable network design problem.

Definition 18. *(The Steiner Network Problem) In the Steiner Network problem as an input we are given:*

- *A graph $G = (V, E)$ with non-negative weights on the edges.*
- *And for each pair of vertices $\{u, v\} \subseteq V$, a number $f_{u,v}$.*

The goal is to find a subgraph H of G with minimum total weight such that for every pair of vertices $\{u, v\} \subseteq V$, there are $f_{u,v}$ edge-disjoint paths from u to v in H (i.e the minimum $u - v$-cut has at least $f_{u,v}$ edges).

For this problem Jain [2001] achieved a 2-approximation using an interesting iterative LP rounding technique. It is not hard to show that approximating Steiner Network is at least as hard as approximating CAP and k-ECSS.

Lemma 19. *There exists an strict reduction from CAP to the Steiner Network problem.*

Proof. Consider an instance I of the Connectivity Augmentation with the k-connected graph $G = (V, E)$ and the set of links L. We transform this to an instance J of the Steiner Network in the following manner. We set $G' = (V, E \cup L)$ and we assign weight zero and one to E and L, respectively. For each pair of vertices of G also set the values of $f_{u,v}$ to $k + 1$. Now using Theorem 8, any feasible solution of size l for I is a feasible solution of cost l for J and vice-versa. □

Lemma 20. *There exists an strict reduction from k-ECSS to the Steiner Network problem.*

Proof. Consider an instance I of k-ECSS with the input graph G and k. From this we obtain an instance J of the Steiner network design in the following manner. We set $G' = (V, E)$ and we assign weight one to E. For each pair of vertices of G also set the values of $f_{u,v}$ to k. Now using Theorem 8, a subset of edges E' is a feasible solution for I if and only if E' is a feasible solution for J. □

A simple variant of CAP is when there is a link for every pair of vertices. Unlike the previously mentioned problems, there is a polynomial-time exact algorithm for this problem (Watanabe and Nakamura [1987]).

One can consider also a weighted version of CAP, the *Weighted Connectivity Augmentation* problem (WCAP), where links have a positive weight and our goal is to minimize the total weight of selected links L'. The same reduction as mentioned before also works in the weighted case. In particular, one can reduce WCAP to weighted version WTAP of TAP for odd k, and to the weighted version WCacAP of CacAP for even k. The best-known approximation factor even for WTAP (hence for WCAP) is 2. This factor was first established by Frederickson and JáJá [1981]. Their algorithm was later simplified by Khuller and Thurimella [1993]. A 2-approximation can also be achieved by various other techniques developed later on, including a primal-dual approach (Goemans et al. [1994]) and iterative rounding (Jain [2001]). Improvements on the factor 2 have only been obtained for restricted cases, including bounded diameter trees (Cohen and Nutov [2011]) and bounded weights (Adjiashvili [2017];

Fiorini et al. [2018]; Grandoni et al. [2018]; Nutov [2017]). Very recently Traub and Zenklusen [2021] published a paper on arXiv in which they provide a $(1 + \ln 2 + \epsilon)$-approximation for WTAP.

Another related problem is the node-connectivity augmentation in which given a k-node-connected graph G and a set of links L, the goal is to find a minimum size subset L' of L such that $G = (V, E \cup L')$ is $(k+1)$-node-connected. One can observe that increasing connectivity by one already poses significant challenges and in general the node-connectivity versions of these problems seem to be more difficult than their edge-connectivity counterparts.

The best-studied version is when $k = 1$, known as *Block-Tree Augmentation*:

Definition 21 (Block-Tree Augmentation Problem)**.** *Given a tree $T = (V, E)$ and a set of links L on V, the goal is to find a minimum size subset L' of L such that $G = (V, E \cup L')$ is 2-node-connected.*

Frederickson and JáJá [1981] and Khuller and Thurimella [1993] achieved a 2-approximation for Block-Tree Augmentation. However, unlike the edge-connectivity counterpart (TAP), an approximation algorithm with ratio better than 2 was achieved only very recently by Nutov [2020]. Nutov exploits our 1.91-approximation algorithm for CacAP (Byrka et al. [2020]) as a black box.

Chapter 3

The Cycle Augmentation Problem

Our first attempt to attack the 2-approximation barrier for the Cactus Augmentation problem (CacAP) was to investigate some interesting special cases. In particular, we introduced the *Cycle Augmentation* problem (CycAP), which is the special case of CacAP where the input cactus is a cycle. Figure 3.1 illustrates an instance of CycAP.

The best known approximation factor for this special case was still 2, like for the general case, and CycAP was even not know to be NP-hard. We also consider the natural weighted version WCycAP of CycAP. We obtained the following main results:

- We present a fast and simple $\frac{5}{3}$-approximation for CycAP.
- We present a slower (still polynomial-time) $(\frac{3}{2} + \epsilon)$-approximation for CycAP.
- We show that CycAP is APX-Hard.
- We show that WCycAP is as hard to approximate as WCacAP.
- We introduce a new LP-formulation for CycAP with integrality gap of $(\frac{3}{2} + \epsilon)$.

Our first result is obtained via a greedy algorithm. The second result is based on a procedure to solve the problem exactly and efficiently when there are no *long* links. More precisely, we show that CycAP is FPT on the maximum distance of the endpoints of the links. These first two results are described in Section 3.2.

Our third result (see Section 3.3) shows that CycAP does not admit a PTAS. We remark that prior to our work this special case was not even known

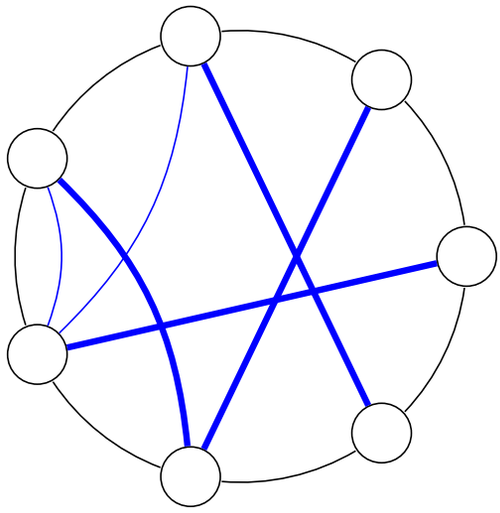


Figure 3.1. An instance of CycAP. The blue edges are the input links. The bold links form an optimal solution.

to be NP-hard. NP-hardness was not obvious at all. In particular, if we replace the cycle with a path the problem can be solved exactly in polynomial time.

Our fourth result (see Section 3.4) shows that in the weighted case there is no substantial difference between WCycAP and WCacAP. Hence unfortunately focusing on the cycle case cannot help much as a first step towards addressing the general case. On the positive side, researchers might focus on the cycle case which is easier to state.

The recent literature on TAP approximation (Adjiashvili [2017]; Fiorini et al. [2018]; Grandoni et al. [2018]) shows that finding strong LP relaxations for the problem can be very helpful to design improved approximation algorithms. In the same spirit, we tried to address the problem of finding LP relaxations for CycAP with small integrality gap. For both TAP and CacAP (hence CycAP) one can define a natural and simple standard cut LP (more details later).

While for TAP it was recently shown that the standard cut LP has integrality gap smaller than 2 (Nutov [2017]), interestingly for CycAP (hence for CacAP) the standard cut LP has integrality gap 2. As the final result we present a stronger LP that, for any $\varepsilon > 0$, has integrality gap at most $\frac{3}{2} + \varepsilon$ (hence matching the approximation ratio of our algorithm). In our opinion this could be useful for future work on CacAP approximation.

3.1 Definitions and Notations

For a set X and element y, we use the shortcut $X \setminus y$ for $X \setminus \{y\}$, and similarly for other set operations.

Given a graph $G = (V, E)$, we let $V(G) = V$ and $E(G) = E$. Recall that in WCacAP we are given a cactus $G = (V, E)$, a set of links $L \subseteq \binom{V}{2}$ and a non-negative weight function $c : L \to \mathbb{R}_{\geq 0}$. The task is to compute a subset of links $A \subseteq L$ such that the graph $(V, E \cup A)$ is 3-connected while minimizing $c(A) := \sum_{\ell \in A} c(\ell)$. The special case where G is a cycle is called WCycAP, and the unweighted versions of the above problems are called CacAP and CycAP respectively. By n we will denote the number of nodes of the considered instance of the problem.

Notice that, given an instance (G, L) of CacAP, we can check in polynomial time if the graph $(V(G), E(G) \cup L)$ is 3-connected by exhaustively checking if the removal of any pair of elements from $E(G) \cup L$ disconnects the graph. Hence we will assume along this work that the instance always admits a feasible solution.

Observation 22. *The 2-edge cuts of a cactus G are identified by pairs $S = \{e, e'\}$ of distinct edges belonging to the same cycle, and consist of the node sets $(U, V \setminus U)$ of the two connected components obtained by removing S from G. A necessary and sufficient condition for a subset of links A to be a feasible solution for WCacAP is that, for any such cut S, there is at least one $\ell =< u, v > \in A$ that $u \in U$ and $v \in (V \setminus U)$. (in which case ℓ **satisfies** the $\{e, e'\}$-cut).*

Note that in the case of CycAP, Observation 22 implies that any feasible solution must be an edge cover as 2-edge cuts defined by neighboring edges of the cycle must be satisfied. Given a 2-edge cut $S = \{e, e'\}$, let L_S be the subset of links satisfying S. The standard cut LP for CycAP is as follows:

$$\begin{aligned} \min \quad & \sum_{\ell \in L} x_\ell && \text{(standard cut LP)} \\ s.t. \quad & \sum_{\ell \in L_S} x_\ell \geq 1 && \forall S : S \text{ is a 2-edge cut} \\ & 0 \leq x_\ell \leq 1 && \forall \ell \in L \end{aligned}$$

Now we proceed to define a standard building block for our algorithms, the *contraction* of a link.

Definition 23. *Contracting a subset of nodes W consists of the following operations: (i) remove the nodes in W and all edges/links incident to them; (ii) add a new node w and, for each original edge/link of type (y, x), $x \in W, y \notin W$,*

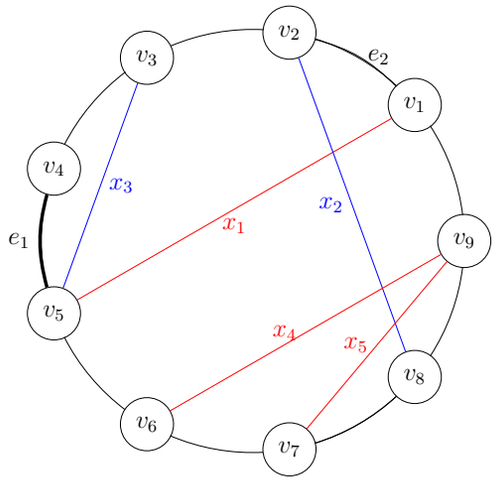


Figure 3.2. Depiction of the Standard cut LP. The blue links are the set of all the links that satisfy the cut $\{e_1, e_2\}$. Therefore this corresponds to the following constraint: $x_2 + x_3 \geq 1$.

add the edge/link (y, w) (of the same weight for the case of links). Note that we do not create loops this way but may introduce parallel links. We say that (y, w) is the image of (y, x) and (y, x) is the preimage of (y, w).

We will sometimes slightly abuse notation and use the same label to denote a link and its image: the meaning will be clear from the context.

For a link $\ell = (u, v)$, we define a sequence $w_0, \ldots, w_q$ of boundary nodes $B(\ell)$ as follows. Consider a simple path from u to v in the cactus, and let $C_1, C_2, \ldots, C_q$ be the ordered sequence of cycles visited by this path (possibly $q = 1$). Note that a path visits a cycle iff it includes an edge from the cycle. We define w_i, $i = 1, \ldots, q-1$ as the unique common node between C_i and C_{i+1}, and set $w_0 = u$ and $w_q = v$.

Definition 24. *Contracting a link ℓ is the operation of contracting its boundary nodes $B(\ell)$. We denote by $G|\ell$ the graph obtained by this operation. Contracting a set of links A is the operation of contracting any $\ell \in A$, and then continue recursively on $G|\ell$ and on the image of $A \setminus \ell$ until A becomes empty.*

Note that contracting a link in a cactus yields again a cactus. We will extensively use the following standard fact.

Lemma 25. *Let (G, L) be a CacAP instance, $A \subseteq L$, and $\ell \in A$. Then A is a feasible solution for (G, L) iff the image of $A \setminus \ell$ is a feasible solution for $(G|\ell, L \setminus \ell)$.*

We require some further notation before proving the lemma. The *internal projections* $S(\ell)$ of ℓ are the links (w_i, w_{i+1}), $i = 0, \ldots, q-1$. In terms of feasibility, ℓ and $S(\ell)$ are equivalent as the following proposition states.

Proposition 26. *Let (G, L) be a CacAP instance and $\ell \in L$. Then ℓ satisfies precisely the same 2-edge cuts as $S(\ell)$.*

Proof. Let $B(\ell) = (w_0, \ldots, w_q)$ and $C_1, \ldots, C_q$ be the corresponding sequence of cycles visited by a simple path between the endpoints of ℓ. Notice that pairs (w_i, w_{i+1}), $i = 0, \ldots, q-1$, subdivide each C_i into two paths next denoted as C_i' and C_i''. Trivially ℓ satisfies only cuts belonging to the cycles $C_1, \ldots, C_q$, and the same holds for $S(\ell)$. Consider any pair (e_1, e_2) belonging to some C_i. Link ℓ satisfies the corresponding cut if and only if precisely one such edge e_j belongs to C_i'. The same holds for (w_i, w_{i+1}), hence for $S(\ell)$. □

In order to prove Lemma 25, let us first consider the simpler case where G is a cycle.

Lemma 27. *Let $(G = (V, E), L)$ be a CycAP instance, $A \subseteq L$, and $\ell = (u, v) \in A$. Then A is a feasible solution for (G, L) iff the image of $A \setminus \ell$ is a feasible solution for the CacAP instance $(G|\ell, L \setminus \ell)$.*

Proof. Let C_1 and C_2 be the two cycles in $G|\ell$, with common node w.

Suppose first that the image of $A \setminus \ell$ is a feasible solution for $(G|\ell, L \setminus \ell)$. Consider a pair of edges $\{e_1, e_2\}$ belonging to a common cycle C_i, and the corresponding cut (S', S'') in $G|\ell$ with $w \in S''$. There must be a link $\ell' \in A \setminus \ell$ satisfying this cut in $G|\ell$. The preimage of ℓ' has one endpoint in S' and the other in $V \setminus S' = (S'' \setminus \{w\}) \cup \{u, v\}$, hence it satisfies the $\{e_1, e_2\}$-cut in G. The remaining pairs of edges $\{e_1, e_2\}$ of G satisfy $e_1 \in C_1$ and $e_2 \in C_2$, modulo symmetries. Those cuts are satisfied by ℓ in G.

Suppose now that A is feasible for (G, L). Consider a pair of edges $\{e_1, e_2\}$ belonging to a common cycle C_i. Let (S', S'') be the corresponding cut in $G|\ell$ with $w \in S''$. Since ℓ does not satisfy that cut in G, this means that there is some other link $\ell' \in A \setminus \ell$ satisfying it. The image of ℓ' has one endpoint in S' and the other in S'', hence it satisfies the $\{e_1, e_2\}$-cut. □

Now we can proceed with the proof of Lemma 25.

Proof of Lemma 25. By Proposition 26, we obtain an equivalent statement of the lemma by replacing A with the set $S(A)$ of the internal projections of links in A and replacing ℓ with its internal projection $S(\ell)$.

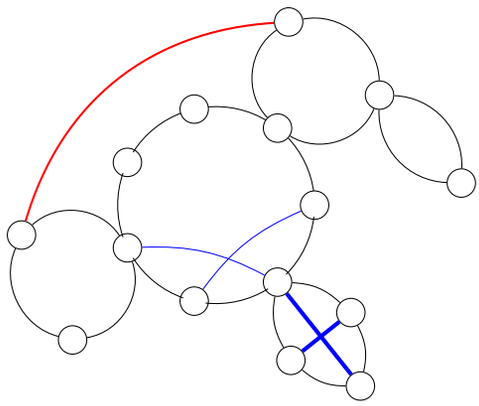


Figure 3.3. The graph G is introduced by the black edges, the internal links are shown by the blue edges and the red edge is an external link. The bold blue edges are a pair of crossing internal links.

Let $B(\ell) = (w_0, \ldots, w_q)$ and $C_1, \ldots, C_q$ be the corresponding sequence of cycles visited by a simple path between the endpoints of ℓ. Consider any cycle C not in the above list. Then trivially any pair of edges in C is covered by links in $S(A) \setminus S(\ell)$. Therefore it is sufficient to consider pairs of edges e_1, e_2 belonging to the same cycle C_i. Let $\ell_i = (w_i, w_{i+1})$ be the internal projection of ℓ with both endpoints in C_i, and define similarly $S_i(A)$ w.r.t. $S(A)$. Then it is sufficient to show that $S_i(A)$ is a feasible solution for the CycAP instance induced by C_i if and only if $S_i(A) \setminus \ell_i$ is a feasible solution for the CycAP instance induced by $C_i|\ell_i$, which follows from Lemma 27. □

3.2 Approximation algorithms

In this section, we present our approximation algorithms for CycAP. We start with a simple $\frac{5}{3}$-approximation to show our main intuition and illustrate the main ideas. Then we present a slightly more complex $\left(\frac{3}{2} + \epsilon\right)$-approximation. Both algorithms are based on greedy choices of links. The approach we will follow in both cases is as follows: in the first phase, we iteratively add a properly chosen subset of a few links to the solution under construction, and then contract them. Notice that, after the first contraction, the cycle structure may be lost and we obtain a CacAP instance instead. These choices are designed so that, at the end of the first phase, the remaining CacAP instance can be solved efficiently, which is done in a second phase with an ad-hoc algorithm. We remark that the running times of the presented algorithms are not analyzed in detail, and indeed such a task may require to devise carefully crafted data structures.

We next describe a simple greedy algorithm that provides a $\frac{5}{3}$-approximation for CycAP. We need the following definitions.

Definition 28. *A link $\ell = (u, v)$ of a CacAP instance is **internal** if both its endpoints belong to a common cycle, and* external *otherwise.*

Definition 29. *Given a CacAP instance, a pair of internal links $\{(u_1, v_1), (u_2, v_2)\}$ of a cycle C is **crossing** if they are node disjoint and deleting u_2 and v_2 disconnects u_1 from v_1 in C.*

In the first stage of the algorithm we only add external links plus crossing pairs of links. More in detail, the algorithm has two main stages. The first stage consists of a set of rounds, where in each round we first check if there exists an external link ℓ, in which case we add it to our solution, contract it and proceed to the next round. Otherwise, if there exists a pair of (internal) crossing links ℓ' and ℓ'', we add them to our solution, contract them and proceed to the next round. If none of the two cases above applies, we are left with a CacAP instance without any external link or crossing pairs of links which we address in the second stage of the algorithm. We refer to this algorithm as crossing-first. As the following lemma states, in the second stage we can efficiently compute the optimal solution.

Lemma 30. *Consider an instance $(G = (V, E), L)$ of CacAP. If there are no external links and no crossing pairs of links, then every minimal solution has size exactly $|V| - 1$ and induces a spanning tree over V.*

Proof. We prove the first part of the claim by induction on $n = |V|$. The base case $n = 2$ is trivial since in this case the instance is just a cycle consisting of two parallel edges and any link must be incident to the two nodes of G (hence defining a feasible solution). For the inductive case, assume the claim is true up to instances having $n - 1$ nodes, and consider an instance of the problem defined by a cactus G having n nodes with optimal solution OPT. If G is not a cycle of length n, then it is defined by a set of cycles of length at most $n - 1$ where every link is internal, so we can apply the inductive hypothesis to each cycle independently. If G is a cycle of n nodes, then let $\ell = (u, v) \in \mathsf{OPT}$. Contracting ℓ leads to a CacAP instance on two cycles C_1 and C_2 sharing a common node w, with $|V(C_1)| + |V(C_2)| = n$. Let OPT' be the optimal solution for the new instance. By Lemma 25, $|\mathsf{OPT}| = |\mathsf{OPT}'| + 1$. Observe that any remaining link ℓ' must have both endpoints in the same C_i (otherwise ℓ and ℓ' would be crossing). Thus by the inductive hypothesis the optimum solution for the problem induced by C_i has size $|V(C_i)| - 1$. It then follows

that $|\mathsf{OPT}'| = |V(C_1)| - 1 + |V(C_2)| - 1 = n - 2$. Hence $|\mathsf{OPT}| = n - 1$ as desired.

For the second part of the claim, it is sufficient to show that a minimal solution does not induce a cycle. By contradiction, consider a minimal solution containing a simple cycle L', and consider now a solution where we remove precisely one arbitrary link $\ell = (u, v)$ from L'. Consider any pair of edges e_1, e_2 belonging to the same cycle such that ℓ satisfies the $\{e_1, e_2\}$-cut. Since $L' \setminus \ell$ induces a simple u-v path, then some $\ell' \in L' \setminus \ell$ must satisfy the cut. Thus $L' \setminus \ell$ is a feasible solution, contradicting the minimality of L'. □

By showing that the number of links in the second stage is a lower bound on the size of the optimal solution we obtained the following result:

Theorem 31. *The* crossing-first *algorithm is a* $\frac{5}{3}$*-approximation for CycAP.*

Proof. Let OPT be the optimal solution and APX the computed solution. Let also n'' be the number of nodes remaining at the end of the first stage, and APX' (resp. APX'') be the set of links added to the solution during the first (resp. second) stage. Since contracting an external link decreases the number of nodes by at least 2 and contracting any pair of crossing links decreases the number of nodes by at least 3, we have that $|\mathsf{APX}'| \leq \frac{2}{3}(n - n'')$.

By Lemma 30, $|\mathsf{APX}''| = n'' - 1$, and hence $|\mathsf{APX}| \leq \frac{2}{3}(n - n'') + n'' - 1 = \frac{2n+n''-3}{3}$. On the other hand, since any feasible solution must be an edge cover, we have that $|\mathsf{OPT}| \geq n/2$. Observe also that $|\mathsf{OPT}| \geq n'' - 1$ since by Lemma 25 contracting links cannot increase the cost of the optimum solution. Thus $|\mathsf{OPT}| \geq \max\{n/2, n'' - 1\}$. We can conclude that $\frac{|\mathsf{APX}|}{|\mathsf{OPT}|} \leq \frac{(2n+n''-3)/3}{\max\{n/2, n''-1\}} \leq \frac{5}{3}$, being $n'' - 1 = n/2$ the worst case. □

Finally we designed an input showing that the approximation ratio of crossing-first is no better that $\frac{5}{3}$ (see Figure 3.4), which shows that our analysis in the proof of theorem 31 was tight.

Lemma 32. *The approximation ratio of the* crossing-first *algorithm is not better than* $\frac{5}{3}$*.*

Proof. Consider the following construction: for each $k \geq 2$ consider an instance (G_k, L_k) of CycAP defined by a cycle of $n = 6k$ nodes (assume that the cycle is defined by the order of the nodes $v_1, v_2, \ldots, v_{6k}$) and the following set of links (see Figure 3.4 (Left)):

- $(v_1, v_{\frac{n}{2}+1}) \in L_k$;

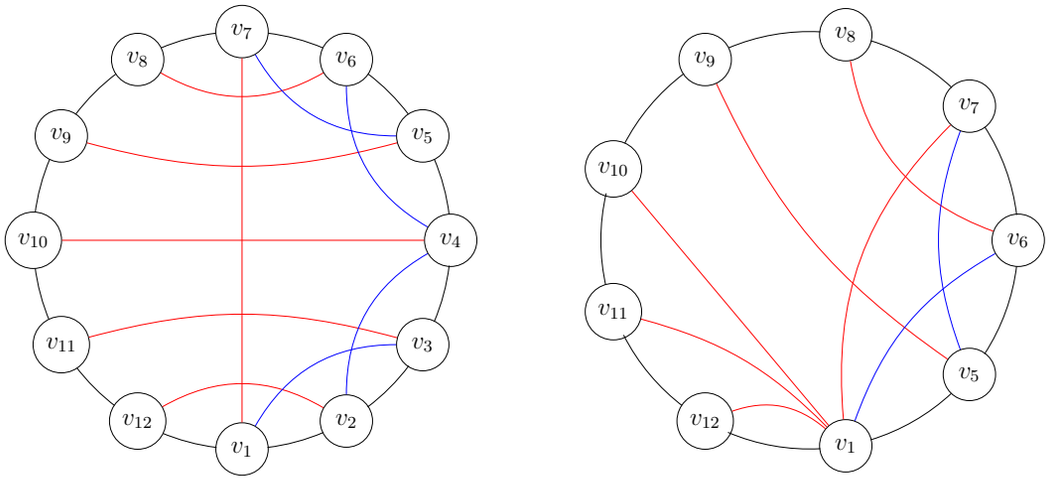


Figure 3.4. **Left**: Instance (G_2, L_2) from the lower bound construction in Lemma 32. Red links define an optimal solution. **Right**: If the algorithm in the first phase picks and contracts the crossing links $\{(v_1, v_3), (v_2, v_4)\}$, this is the obtained CacAP instance.

- For each $i = 1, \ldots, \frac{n}{2} - 1$, $(v_{i+1}, v_{n+1-i}) \in L_k$;
- For each $i = 1, \ldots, \frac{n}{6}$, $(v_{3(i-1)+1}, v_{3(i-1)+3}) \in L_k$ and $(v_{3(i-1)+2}, v_{3(i-1)+4}) \in L_k$;

Notice that the first and second set of links define a feasible solution of size $\frac{n}{2}$, hence being optimal: if we remove any two edges of the cycle, then we are either satisfying the corresponding cut via $(v_1, v_{\frac{n}{2}+1})$, or one side of the partition is contained in either $\{v_2, \ldots, v_{\frac{n}{2}}\}$ or in $\{v_{\frac{n}{2}+2}, \ldots, v_n\}$ but the links selected form a matching between those sets.

We will now prove that there exists a sequence of choices performed by our algorithm that outputs a solution of size $\frac{5n}{6} - 1$, which implies that the approximation ratio is at least $\frac{5}{3} - \frac{2}{n}$ and this value approaches $\frac{5}{3}$ as k goes to infinity. Notice first that the pair of links $\{(v_1, v_3), (v_2, v_4)\} \subseteq L_k$ is crossing, and hence the algorithm can include them in the solution in the first round (and finish the round). Furthermore, after these links are contracted no link becomes external as the new cactus instance consists of a cycle of length $n-3$, and also the links with endpoints v_n, v_{n-1} and v_{n-2} are not part of any pair of crossing links (see Figure 3.4 (Right)). If we now iteratively pick all the pairs of crossing links $\{(v_{3(i-1)+1}, v_{3(i-1)+3}), (v_{3(i-1)+2}, v_{3(i-1)+4})\} \subseteq L_k$, $i = 2, \ldots, \frac{n}{6}$, after $\frac{n}{6}$ rounds we end up with a cycle of length $\frac{n}{2}$ without crossing links, and the algorithm must now take the remaining $\frac{n}{2} - 1$ links to complete the solution. Thus, the size of the computed solution is $2 \cdot \frac{n}{6} + \frac{n}{2} - 1 = \frac{5}{6}n - 1$, proving the

claim. □

Now we describe our more advanced $(\frac{3}{2}+\epsilon)$-approximation algorithm. Here we introduce a new classification of the links, namely a classification by size.

Definition 33. *The **length** of an internal link (u, v) is the length of the shortest path between u and v in the corresponding cycle. For a given parameter $0 < \epsilon < 1$, an internal link is called **long** if its length is at least $\frac{1}{\varepsilon}$, and **short** otherwise.*

Our algorithm consists of the following two main phases. In the first phase, we iteratively check if there exists a long (internal) link ℓ. Otherwise, we check if there exists an external link ℓ. In both cases, we add ℓ to the solution under construction and contract it. Observe that contracting links does not create new long links, hence we will first select a set L_{long} of long links, and then a set L_{ext} of external links.

After exhausting the previous choices, we move to the second phase. Here we are left with an instance where all links are short and internal. This allows us to solve independently the sub-instance induced by each cycle. We refer to this algorithm as long-first. This second stage can be solved efficiently, due to the lack of long links, by means of the following lemma[1].

Lemma 34. *Given a CycAP instance, there exists an algorithm based on dynamic programming that returns the optimal solution in time poly$(n) \cdot 2^{O(h_{\max}^2)}$, where $h_{\max}$ is the maximum length among the links.*

Let L_{short} be the collection of edges obtained in the second stage. The final solution is $L_{\text{long}} \cup L_{\text{ext}} \cup L_{\text{short}}$. The intuition is that $|L_{\text{long}}|$ is very small compared to the size of the optimal solution, external edges are greedily good choices and L_{short} is an exactly optimal solution for the final stage and is a lower-bound for the size of the optimal solution.

Theorem 35. *The long-first algorithm is a $(\frac{3}{2}+\varepsilon)$-approximation algorithm for CycAP.*

Proof. The running time of the algorithm is upper-bounded by poly$(n)2^{O(1/\varepsilon^2)}$. Consider next the approximation factor. Note first that $|L_{\text{long}}| \leq \varepsilon n$. Indeed, contracting a long link always increases the number of cycles in the cactus by one without decreasing the number of edges, and all these cycles always have

[1]This lemma implies that CycAP is FPT with parameter $h_{\max}$.

size at least $1/\varepsilon$, so there are at most εn of them. Similarly to Theorem 31, we have that $|\mathsf{OPT}| \geq |L_{\mathsf{short}}|$ and $|\mathsf{OPT}| \geq \frac{n}{2}$.

If $|L_{\mathsf{long}}|+|L_{\mathsf{ext}}|+|L_{\mathsf{short}}| \leq \frac{(3+2\varepsilon)n}{4}$ then we already have a $\left(\frac{3}{2}+\varepsilon\right)$-approximation as $|\mathsf{OPT}| \geq \frac{n}{2}$. Otherwise, since the contraction of each external link reduces the number of nodes by at least 2 and the contraction of any other link reduces the number of nodes by at least 1, we have that $|L_{\mathsf{long}}|+2|L_{\mathsf{ext}}|+|L_{\mathsf{short}}| \leq n$. So $|L_{\mathsf{ext}}| \leq n - \frac{(3+2\varepsilon)n}{4} = \frac{(1-2\varepsilon)n}{4}$ and hence $|L_{\mathsf{ext}}|+|L_{\mathsf{long}}| \leq \frac{n+2\varepsilon n}{4} \leq \left(\frac{1}{2}+\varepsilon\right)|\mathsf{OPT}|$. Since $|\mathsf{OPT}| \geq |L_{\mathsf{short}}|$, we have that in this case the size of the solution is also at most $(\frac{3}{2}+\varepsilon)|\mathsf{OPT}|$, concluding the proof. □

Remark 36. *By replacing ε with $1/\sqrt{\log n}$ in the above construction, we can obtain a slightly improved approximation factor of $3/2 + o(1)$ which still runs in polynomial time.*

It remains to prove Lemma 34. To do this, we need some more notations. Given a link $\ell = (u, v)$, we say that the edges of the shortest path between u and v in the cycle are *covered* by ℓ (in case of multiple shortest paths we choose the one going from u to v in counter-clockwise order along the cycle). Given an edge e of the cycle, we define the *cut-neighborhood* of e, namely $\mathscr{N}(e)$, as the $2h_{\max} - 1$ edges that are closest to e, e included. We also define $\mathscr{N}_L(e)$ as the set of links in L covering at least one edge from $\mathscr{N}(e)$.

Notice that in any feasible solution to a CycAP instance, at most one edge of the cycle is not covered: if it is not the case, then the cut defined by two uncovered edges is not satisfied as any link satisfying the cut would cover one of these two edges. We can use this observation to characterize the feasibility of a solution in terms of the cut-neighborhoods.

Lemma 37. *Consider a CycAP instance and let A be a set of links such that every edge of the cycle is covered by some link in A. A is feasible iff for each edge e, all the $\{e, e'\}$-cuts, where $e' \in \mathscr{N}(e)$, are satisfied.*

Proof. If A is feasible then the required properties are clearly satisfied since every cut is satisfied. On the other hand, suppose that A satisfies that every edge is covered by some link in A and the $\{e, e'\}$-cuts are satisfied for every edge e and $e' \in \mathscr{N}(e)$. Consider a pair of edges $\{e, e'\}$ such that $e' \notin \mathscr{N}(e)$. By definition of $\mathscr{N}(e)$ there is no link in A covering both edges at the same time, and as e is covered by some link, this link satisfies the $\{e, e'\}$-cut. This implies that A is feasible as every cut is satisfied. □

This lemma is useful as it implies that, given an edge e and a set of links S, we can optimally complete S in order to satisfy every $\{e, e'\}$-cut in time

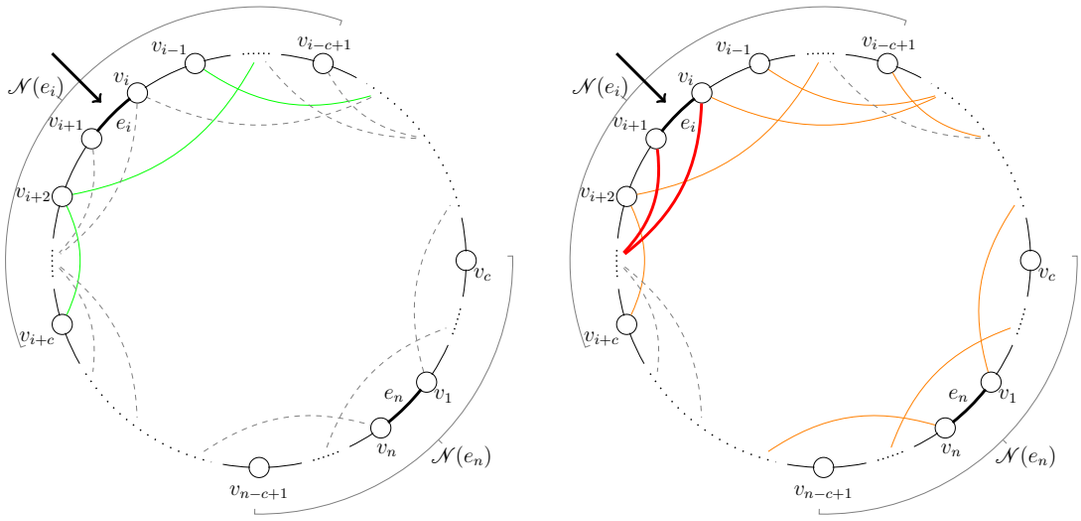


Figure 3.5. Depiction of an iteration of the DP from Lemma 34, where we are currently at edge e_i. **Left**: Green links correspond to S and at this point we must decide which extra links to add to S in order to satisfy the edges $e_1, \ldots, e_i$. **Right**: This computation is done by looking at a proper previous cell in the table (orange links) which contains S and satisfies $e_1, \ldots, e_{i-1}$, and then add the extra required links A^* (red links) in order to satisfy e_i too.

$2^{O(h^2_{\max})}$ just by guessing the subset of links from $\mathcal{N}_L(e)$ that must be added, which are $O(h^2_{\max})$ only. Now we proceed to present the proof.

Proof of Lemma 34. Let us assume that we deal with instances of CycAP such that there exists an optimal solution where every edge is covered by some link. If it is not the case, as there may be only one uncovered edge, we can guess this edge and contract it; this leads to an equivalent instance of the problem where we can require that the optimum solution covers all the edges. We say that an edge e is *satisfied* by a set of links A if it is covered by some link in A and furthermore every $\{e, e'\}$-cut is satisfied by A. In particular A is a feasible solution for the problem iff it satisfies all the edges.

We next design a dynamic programming algorithm to compute a minimum cardinality feasible solution. Let us name the nodes $v_1, v_2, ..., v_n$ in counter-clockwise order starting from some arbitrary node v_1, and let the edges be $e_i = (v_i, v_{i+1})$ for each $i = 1, \ldots, n$ (assuming $v_{n+1} = v_1$).

For each edge e_i and $S \subseteq \mathcal{N}_L(e_i)$, we define a cell $T[i][S]$ which will correspond to a set S' of links of smallest cardinality such that for each $j \in \{1, \ldots, i\}$, e_j is satisfied by S', subject to $S \subseteq S'$. It is then sufficient to return $T[n][\emptyset]$.

We initialize the table by computing $T[1][S]$ for each set $S \subseteq \mathscr{N}_L(e_1)$, which can be done by guessing how to complete S in order to satisfy e_1 with links from $\mathscr{N}_L(e_1)$. Then, for each $i \geq 2$ and $S \subseteq \mathscr{N}_L(e_i)$, in order to fill the cell $T[i][S]$, we consider all the possible subsets $A \subseteq \mathscr{N}_L(e_i)$ such that $S(A) := T[i-1][(S \cup A) \cap \mathscr{N}_L(e_{i-1})] \cup (S \cup A)$ satisfies e_i. Among them we select a set A^* that minimizes $|S(A)|$, and we set $T[i][S] = S(A^*)$ (see Figure 3.5 for a sketch).

The correctness of the computation follows by a simple induction on i. The table can be filled in total time $\text{poly}(n) \cdot 2^{O(h^2_{\max})}$, plus an extra factor n from the initial guessing of an uncovered edge (that is contracted). □

We complement Theorem 35 with an asymptotically matching lower bound.

Lemma 38. *The approximation ratio of the* long-first *algorithm is at least* $\frac{3}{2}$.

Proof. Consider the following construction: for each $k > \frac{1}{2\varepsilon}$ consider an instance (G_k, L_k) of CycAP defined by a cycle of $n = 4k$ nodes (assume that the cycle is defined by the order of the nodes $v_1, v_2, \ldots, v_{4k}$) and the following set of links (see Figure 3.6 (Left)):

- For each $i = 1, \ldots, \frac{n}{2} - 1$, $(v_{i+1}, v_{n+1-i}) \in L_k$;
- $(v_1, v_{\frac{n}{2}+1}) \in L_k$;
- For each $i = 1, \ldots, \frac{n}{4} - 1$, $(v_{i+1}, v_{\frac{n}{2}+1-i}) \in L_k$.

As argued in Lemma 32, the first and second set of links define an optimal solution of size $\frac{n}{2}$. We will now prove that there exists a sequence of choices performed by our algorithm that outputs a solution of size $\frac{3n}{4} - 1$, which implies that the approximation ratio is at least $\frac{3}{2} - \frac{2}{n}$ and this value approaches $\frac{3}{2}$ as k goes to infinity. Notice first that the link $(v_{\frac{n}{4}+1}, v_{\frac{3n}{4}+1}) \in L_k$ has length $2k > \frac{1}{\varepsilon}$ and hence it is long so the first stage of the algorithm can include it in the solution. After doing that, the second and third set of links become external and thus the algorithm will include them in the solution. Once all these links are included and contracted, we get a cactus consisting of two cycles of $\frac{n}{4}$ nodes each and without crossing links (see Figure 3.6 (Right)). Hence, the algorithm must pick all the remaining links to complete the solution. The size then of this solution is $\frac{n}{4} + 1 + 2\left(\frac{n}{4} - 1\right) = \frac{3n}{4} - 1$. □

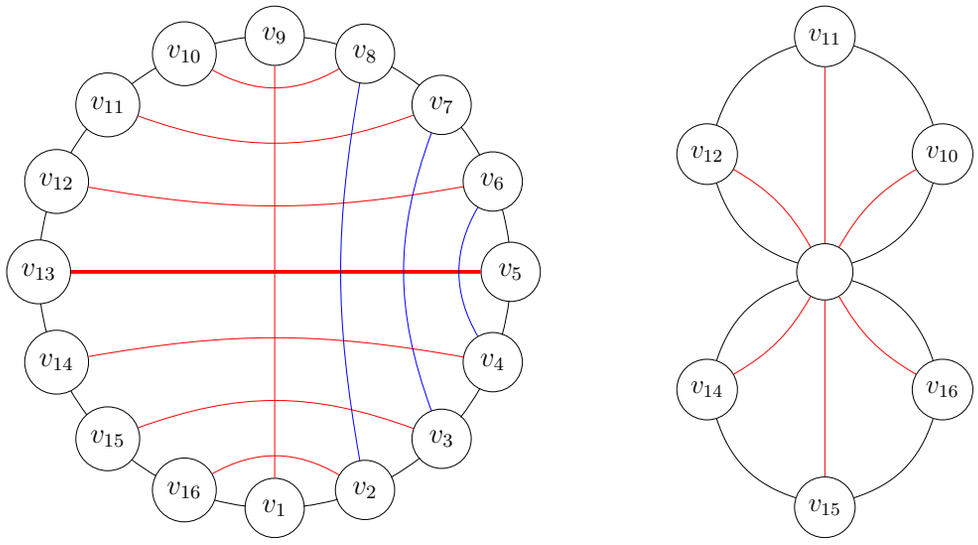


Figure 3.6. **Left**: Instance (G_4, L_4) from the lower bound construction in Lemma 38. An optimal solution is defined by red links. **Right**: If the algorithm picks first the thick red link (which is long) and then the links which become external (blue links and (v_1, v_9)) we obtain this subinstance without crossing pairs of links.

3.3 Hardness of Approximation for CycAP

In this section we prove that CycAP is APX-hard via a reduction from a restricted case of 3-Dimensional Matching (3DM). In the general version of 3DM we are given three disjoint sets W, X and Y having equal cardinality p and a set of m hyperedges $H \subseteq W \times X \times Y$. A (3D) matching is a subset $M \subseteq H$ such that each element of $W \cup X \cup Y$ belongs to at most one hyperedge in M, and this matching is *perfect* if $|M| = p$. Notice that in a perfect matching M each element of $W \cup X \cup Y$ belongs to precisely one hyperedge. The goal is to determine whether a perfect matching exists. We will consider the special case 3DM-K, $K \in \mathbb{N}$, where we add the constraint that each element from $W \cup X \cup Y$ appears in at most K hyperedges. The following result will help us to conclude our final claim.

Theorem 39 (Petrank [1994])**.** *For some fixed $\varepsilon_0 > 0$, it is NP-hard to distinguish whether an instance of 3DM-5 with $|W| = |X| = |Y| = q$ has a perfect matching (of size q) or every matching has size at most $(1 - \varepsilon_0)q$.*

The proof of the following theorem is similar in spirit to the proof of NP-hardness for WTAP due to Frederickson and JáJá [1981] and the extension

presented by Kortsarz et al. [2004]. In the first reduction the authors start from an instance A of 3DM with $3p$ nodes and m hyperedges, and build a WTAP instance B such that: A has a feasible solution (with p hyperedges) iff B has a feasible solution with $p+m$ links. By duplicating the edges in B, one obtains a CacAP instance C with exactly the same property over some cactus G. Our main idea is to turn C into an instance D of CycAP by constructing an Euler tour G' out of G and shortcutting some nodes. However, we need to carefully choose the ordering in the Euler tour in order to preserve a mapping between the feasible solutions of C and D. By following the refined approach from the second reduction, we will show that it is hard to distinguish solutions with a gap depending on the maximum degree in the instance and then use Theorem 39 to conclude the following result.

Theorem 40. *For some fixed $\varepsilon > 0$, it is NP-hard to approximate CycAP within a factor $1+\varepsilon$.*

Construction of the instance: Let $H \subseteq W \times X \times Y$ be an instance of 3DM with $|H| = m$, $W = \{w_1, \ldots, w_p\}, X = \{x_1, \ldots, x_p\}$ and $Y = \{y_1, \ldots, y_p\}$. We will define an instance $(G = (V, E), L)$ of CycAP where nodes are placed on the cycle in the order as they appear below in counterclockwise direction (see Figure 3.7 for a depiction of the instance):

- For each node $x_i \in X$ we define a node x_i;
- For each node $y_i \in Y$ we define a node y_i;
- Let $H(w_i)$ denote the hyperedges in H containing $w_i \in W$. For each hyperedge $h \in H(w_i)$ we define two nodes, namely h_X and h_Y (*hyperedge nodes*). These nodes are added to the cycle in the following order. For each $i \in \{1, \ldots, p\}$, we add first nodes h_X corresponding to hyperedges in $H(w_i)$ (in some arbitrary order) and then the corresponding nodes h_Y respecting the *same order used before*. We will denote the first set of nodes by $H_X(w_i)$, and the second set by $H_Y(w_i)$.

The set of links L is defined as follows:

- For each hyperedge $h \in H$ we add the link (h_X, h_Y);
- For each hyperedge $h \in H$ and a node $x \in X$, we add the link (h_X, x) iff $x \in h$;
- For each hyperedge $h \in H$ and a node $y \in Y$, we add the link (h_Y, y) iff $y \in h$.

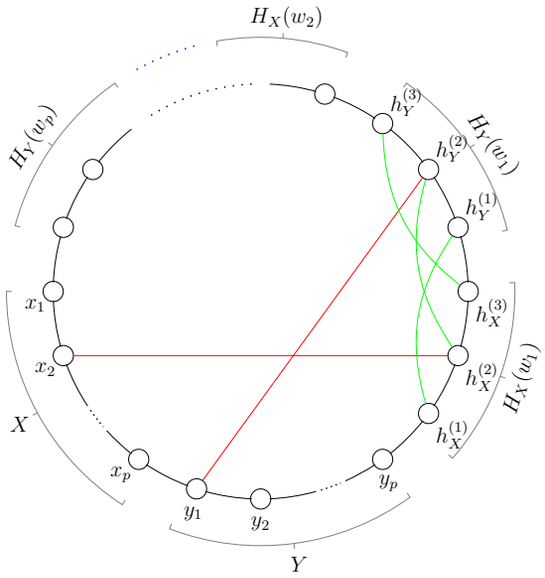


Figure 3.7. Example of the construction in Theorem 40. Red links correspond to hyperedge $h^{(2)} = (w_1, x_2, y_1)$ and green links join the copies of the hyperedges.

Lemma 41. *If the 3DM instance H contains a 3D matching M with p hyperedges then the CycAP instance (G, L) constructed as above admits a solution A of size $p + m$.*

Proof. Suppose that H contains a 3D matching M of size p. We build a solution A to (G, L) as follows: For each hyperedge $h = (w, x, y) \in M$ we add to A the links (h_X, x) and (h_Y, y). Also, for each hyperedge $h \in H \setminus M$ we add the link (h_X, h_Y) to A. Observe that the total number of links in A is $2p + (m - p) = p + m$.

Let us show that A is a feasible solution. By Observation 22, it is sufficient to consider any pair of edges $\{e_1, e_2\}$, and show that there exists some link $\ell \in A$ satisfying the corresponding $\{e_1, e_2\}$-cut. Let us denote by S' and S'' the sets of nodes induced by the cut. Let H_X (resp., H_Y) be the collection of nodes of type h_X (resp., h_Y). We make the following case distinction: Suppose first that e_1 is incident to two nodes in X or $e_1 = (x_p, y_1)$ (the case e_1 being incident to two nodes in Y is symmetric). We distinguish the following 3 subcases depending on e_2:

1. Suppose e_2 is incident to at least one node in $X \cup Y$. Then one of the sets in the cut, say S'', contains all the hyperedge nodes while S' contains at least one node in $z \in X \cup Y$. By construction each node in X (resp. Y) is adjacent to some node in H_X (resp., in H_Y). Thus this cut is satisfied.

2. Suppose e_2 is not incident to any node in $H_Y(w_p)$. Then one of the sets in the cut, say S', contains completely Y, while S'' contains $H_Y(w_p)$. By construction, for $h = (w_p, x, y) \in M$, $\ell = (h_Y, y) \in A$, hence this cut is satisfied.

3. Suppose e_2 is incident to some node in $H_Y(w_p)$. Then one of the sets in the cut, say S'', contains H_X while the other set contains at least one node x from X. Again by construction, for $h = (w, x, y) \in M$, $\ell = (h_X, x) \in A$. Hence this cut is satisfied.

Suppose on the other hand that e_1 and e_2 are incident to at least one hyperedge node. Notice that one of the sets in the cut, say S', contains $X \cup Y$. We distinguish the following 2 subcases:

1. If S'' contains entirely $H_X(w)$ or $H_Y(w)$ for some $w \in W$, then for $h = (w, x, y) \in M$, (h_X, x) or (h_Y, y) is contained in A and the cut is satisfied.

2. In the remaining case we prove that the following claim holds: There exists an hyperedge h such that $h_X \in S'$ and $h_Y \in S''$.

   Suppose by contradiction that for every hyperedge h both h_X and h_Y belong to the same side of the considered cut. Let w_i be such that either $H_X(w_i)$ or $H_Y(w_i)$ has non-empty intersection with both sides of the cut. Note that such w_i must exist, otherwise there would exist w_j such that S'' contains either $H_X(w_j)$ or $H_Y(w_j)$ completely which was already covered by the previous case. Assume w.l.o.g. that $H_X(w_i) = \{h_X^1, \ldots, h_X^q\}$ is the considered set with elements sorted in counterclockwise direction. Since h_X^1 and h_Y^1 are on the same side of the partition and $H_X(w_i)$ is not fully contained in any side of the partition, it must hold that one set of the partition is properly contained in $H_X(w_i)$. Then any node inside that set has its copy on the other side of the partition. This is in contradiction with the assumption.

   Let h be an hyperedge as in the previous claim. We are either adding to the solution the link that joins both copies of h (i.e. the case when $h \notin M$) and the proof is finished, or we are adding the two links joining the two copies of h to elements in X and Y (i.e. the case when $h \in M$). Since $X \cup Y$ is contained in S' and both copies of h are in different sides of the partition, one of the links satisfies the cut.

□

For any $z \in W \cup X \cup Y$ in a 3DM instance, let $\deg(z)$ be the number of hyperedges in H containing z. Let also Δ denote the maximum degree of the instance, i.e., $\Delta = \max_{z \in W \cup X \cup Y} \deg(z)$. By following an analogous approach to the one from Kortsarz et al. [2004], we can prove that even instances with a gap can be mapped.

Lemma 42. *If the CycAP instance (G, L) constructed as above admits a solution A with $|A| \leq (1+\varepsilon)(p+m)$, then the 3DM instance H contains a 3D matching M with $|M| \geq p - (2+10\Delta)(p+m)\varepsilon$.*

Proof. Let A be a feasible solution to (G, L) with $|A| \leq (1+\varepsilon)(p+m)$. Note that G contains $2(p+m)$ nodes and the links must form an edge cover (otherwise the resulting graph would not be 3-connected). Call a node *permissible* if it is adjacent to exactly one link in A and *impermissible* otherwise. Let V_{perm} and V_{imperm} be the set of permissible and impermissible nodes respectively. We will first prove that the number of impermissible nodes is upper bounded by $2\varepsilon(p+m)$. In fact, if $\deg_A(v)$ denotes the number of links in A incident to v, we have that

$$2|A| = \sum_{v \in V} |\deg_A(v)| = \sum_{v \in V_{\text{perm}}} \deg_A(v) + \sum_{v \in V_{\text{imperm}}} \deg_A(v) \geq |V_{\text{perm}}| + 2|V_{\text{imperm}}|$$

where the last inequality comes from the fact that impermissible nodes are adjacent to at least two links. Since $|A| \leq (1+\varepsilon)(p+m)$, and $|V_{\text{perm}}|+|V_{\text{imperm}}| = 2(p+m)$, we can conclude the claim.

We will now compute a set M' which is almost a matching. We initialize $M' = \emptyset$ and then, iteratively for $j = 1, \ldots, p$, we try to add an hyperedge to M' as follows: if x_j is permissible, then it is adjacent to one node $h_x^{(j)} \in H_X$ (let us assume $h_x^{(j)} \in H_X(w_i)$); if both $h_x^{(j)}$ and its copy $h_y^{(j)} \in H_Y(w_i)$ are permissible, then $h_y^{(j)}$ is adjacent to one node y_k. If y_k is permissible, then we add (w_i, x_j, y_k) to M'. Notice that hyperedges added by this procedure are indeed in H by construction. Our claim is that $|M'| \geq p - 2\Delta(p+m)\varepsilon$. Actually, if x_j, $h_x^{(j)}$ or $h_y^{(j)}$ are impermissible, then only one iteration fails (the one indexed by j). If y_k is impermissible then it can cause at most Δ iterations to fail, since it can be connected to at most Δ nodes in H_Y. If we denote by n_y the number of impermissible nodes y_k involved in the procedure, then the number of iterations that fail is at most $(2\varepsilon(p+m) - n_y) + n_y\Delta$. Since $n_y \leq 2\varepsilon(p+m)$ (the total number of impermissible nodes), the number of iterations that fail is at most $2\Delta(p+m)\varepsilon$, proving the claim.

By construction, hyperedges in M' have different elements from X and Y but elements from W might be repeated. Thus, for every w_i belonging to more

than one hyperedge in M', we remove from M' all but one of such hyperedges, obtaining M'' which is now a matching. Let $\mu = p - |M''|$ be the number of vertices w_i not appearing in any hyperedge of M' (equivalently of M''). Since $|M'| - |M''| \leq p - |M''| = \mu$, we can find a lower bound on the size of M'' by bounding above μ. We indeed claim that $\mu \leq (2 + 8\Delta)(p + m)\varepsilon$.

Let L' be the links in L of the form $(x_j, h_X^{(j)})$ and $(y_k, h_Y^{(j)})$ where $h_X^{(j)}$ corresponds to a hyperedge $(w_i, x_j, y_k) \in M'$ and $h_Y^{(j)}$ corresponds to its copy. We have that $|L'| = 2|M'| \geq 2p - 4\Delta(p + m)\varepsilon$, hence

$$|A \setminus L'| \leq (1 + \varepsilon)(p + m) - 2p + 4\Delta(p + m)\varepsilon = m - p + (1 + 4\Delta)(p + m)\varepsilon.$$

Consider on the other hand the μ nodes w_i which are not intersected by hyperedges in M'. Since A is a feasible solution, for each such w_i there must be a link in A connecting $H_X(w_i) \cup H_Y(w_i)$ and $X \cup Y$, because otherwise we could disconnect $H_X(w_i) \cup H_Y(w_i)$ from the rest of the graph by removing the two edges in the boundary of $H_X(w_i) \cup H_Y(w_i)$, contradicting the feasibility of A. Notice that these μ links are part of $A \setminus L'$. Furthermore, since A is an edge cover, the remaining $2m - 2p - \mu$ nodes in $H_X \cup H_Y$ untouched by L' plus the μ aforementioned links must be incident to some link in A, implying that

$$|A \setminus L'| \geq \mu + \frac{2m - 2p - \mu}{2} = m - p + \frac{\mu}{2}.$$

Combining both inequalities we get that $\mu \leq (2 + 8\Delta)(p + m)\varepsilon$, and hence we conclude that the size of M'' is at least

$$|M'| - \mu \geq p - 2\Delta(p + m)\varepsilon - (2 + 8\Delta)(p + m)\varepsilon = p - (2 + 10\Delta)(p + m)\varepsilon,$$

completing the proof. □

We can now use Lemmas 41 and 42 together with Theorem 39 to conclude the proof of Theorem 40. Notice that in 3DM-5, since $\Delta = 5$, we have that $m = |H| \leq 5|W| = 5p$.

Proof of Theorem 40. We will show that our reduction presented above is gap-preserving. Specifically, we will show that if H is an instance of 3DM-5 and (G, L) is the corresponding CycAP instance, then

1. If H admits a matching of size p, then (G, L) admits a feasible solution of size $p + m$;

2. If H does not admit a matching of size at least $p(1 - \varepsilon_0)$, then (G, L) does not admit a feasible solution of size at most $(p + m)(1 + \frac{\varepsilon_0}{312})$.

The first statement follows directly from Lemma 41, while the second is the contrapositive of Lemma 42 when setting $\varepsilon = \frac{\varepsilon_0}{312}$, as in this case we have that $p - (2 + 10\Delta)(5p + p)\varepsilon = p(1 - 312\varepsilon) = p(1 - \varepsilon_0)$. □

3.4 Hardness of Approximation for WCycAP

We now provide an approximation preserving reduction from WCacAP to WCycAP. Note that finding a better-than-2-approximation for WCacAP is at least as hard as finding such an approximation for WTAP, a big open problem in the area. Therefore our reduction shows that achieving a similar result for WCycAP is a very hard task as well.

Theorem 43. *Given an instance A of WCacAP, it is possible to construct in polynomial time an instance B of WCycAP (whose only possibly new weight value is 0) such that any feasible solution to A can be mapped in polynomial time into a feasible solution to B of the same cost and vice versa.*

To prove Theorem 43 we make use of the "inverse" of the contraction of a link, which we call an *expansion*: Consider a WCacAP instance with a node v with degree greater than 2. An expansion of v will consist of taking two cycles containing v and replacing them by the Eulerian tour that traverses them starting from v. Every node appears exactly once except for v which appears twice, for which we create two copies: v_1 the starting node and v_2 the intermediate one. The links originally incident to v are replaced by links of the same cost incident to v_1, and we also add a link of cost zero between v_1 and v_2 (see Figure 3.8 for an example). The two main properties of this procedure are that: (1) the contraction of a link created by an expansion brings back the graph to the original state and (2) v is replaced by v_1 and v_2, which have degree $\deg(v) - 2$ and 2, respectively.

3.5 LP Relaxations for CycAP

We start by lower-bounding the integrality gap of the standard cut LP for CycAP.

Lemma 44. *The Integrality gap of the standard cut LP for CycAP is at least 2.*

Proof. Consider the input to be a graph G that is a cycle of size k. Now for each edge in edge in G, we introduce a link which is parallel to e.

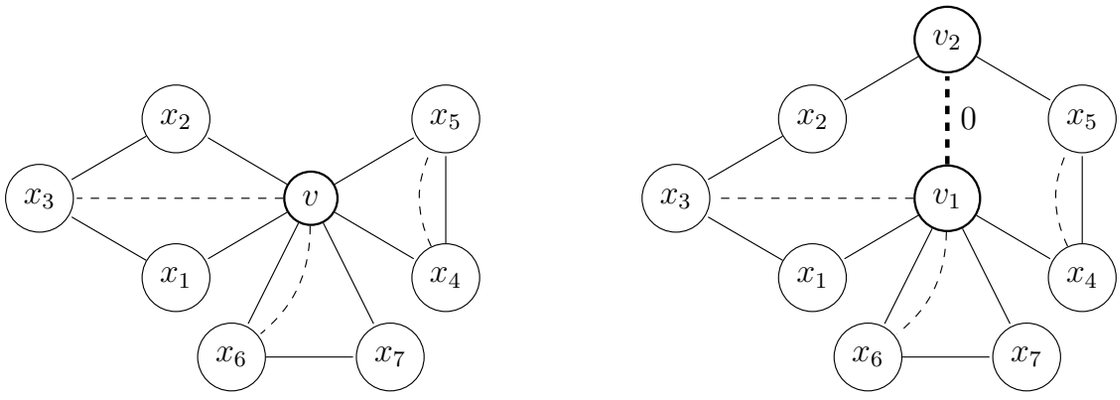


Figure 3.8. Depiction of an expansion applied to node v in the left graph considering the cycles to the left and right of v. Dashed edges correspond to links and the highlighted link in the right graph corresponds to the extra link of cost zero added by the expansion.

In this case, since the 2-edge cuts are any pair of the edges of G, then a subset L' of links is a feasible solution if and only if $|L| > k - 2$. Moreover, setting each variable to $\frac{1}{2}$, we can satisfy all the constraints of the standard cut LP, which yields a feasible fractional solution of cost $\frac{k}{2}$. □

This shows that the standard cut LP is not strong enough even for instances without crossing nor long links, cases that we can handle optimally via combinatorial algorithms. We next present a stronger LP that exploits a more general set of constraints.

Let $(G = (V, E), L)$ be a CycAP instance and $S \subseteq E$. We define the *S-reduced* instance (G_S, L_S) as follows: We contract the edges of $E \setminus S$, obtaining a cycle with $|S|$ edges which defines G_S, and the set of links L_S will correspond to the images of L. Notice that there is a one-to-one relation between L_S and the links in L which satisfy some cut defined by a pair of edges from S. We denote by OPT_S the optimal solution for the instance (G_S, L_S)[2]. The following lemma characterizes the feasibility of a solution.

Lemma 45. *Given an instance (G, L) of CycAP, a solution $A \subseteq L$ is feasible iff for every $S \subseteq E$ it holds that $|A \cap L_S| \geq |\mathsf{OPT}_S|$.*

Proof. Suppose that there exists $S \subseteq E$ such that $|A \cap L_S| < |\mathsf{OPT}_S|$. This means that $A \cap L_S$ is not a feasible solution for (G_S, L_S) and hence there exist two edges $e_i, e_j \in S$ such that no link in $A \cap L_S$ satisfies the $\{e_i, e_j\}$-cut. As

[2]For $|S| \leq 1$, we simply set $\mathsf{OPT}_S = \emptyset$.

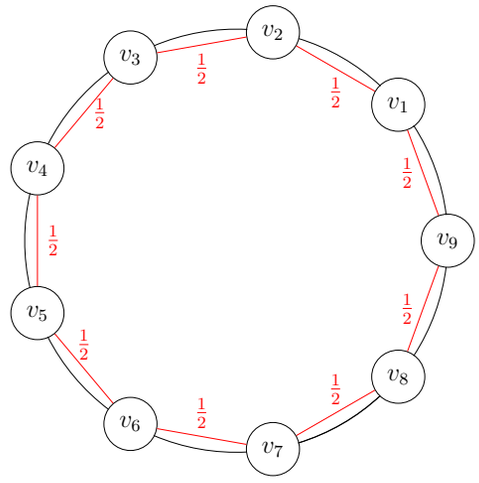


Figure 3.9. Instance showing that the standard cut LP has integrality gap 2. The label on the links correspond to their LP value.

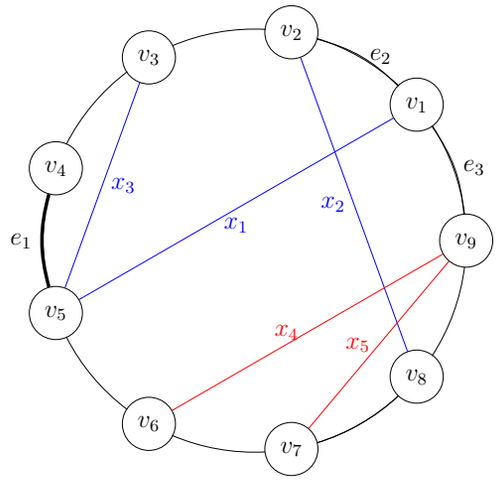


Figure 3.10. Depiction of the k-edge-cut LP. The set $L_{\{e_1,e_2,e_3\}}$ is formed by the blue links. This corresponds to the following constraint: $x_1 + x_2 + x_3 \geq 1$, which does not appear in the standard cut LP.

the remaining links in $A \setminus L_S$ also do not satisfy the cut by definition, this cut remains unsatisfied in the original instance, implying that A is not feasible.

On the other hand, suppose that A satisfies the claimed property for every set S. If we consider just sets S consisting of two edges this is exactly the characterization of feasibility shown in observation 22, implying that A is feasible. □

This implies that we can add the constraint $\sum_{\ell \in L_S} x_\ell \geq |\mathsf{OPT}_S|$ for $S \subseteq E$. Unfortunately there is an exponential number of such constraints and most of them require to compute $|\mathsf{OPT}_S|$ for large instances. However, if we restrict ourselves to sets of edges having constant size, we get an LP formulation with polynomially many constraints that can be written in polynomial time. We call this LP the k-edge-cut LP for a given constant $k \in \mathbb{N}$, which is similar in spirit to the *bundle-LP* for TAP introduced by Adjiashvili [2017].

$$\begin{aligned} \min \quad & \sum_{\ell \in L} x_\ell && (k\text{-edge-cut LP}) \\ s.t. \quad & \sum_{\ell \in L_S} x_\ell \geq |\mathsf{OPT}_S| && \forall S \subseteq E, |S| \leq k \\ & 0 \leq x_\ell \leq 1 && \forall \ell \in L \end{aligned}$$

Notice that for $k = 2$ this is exactly the standard cut LP. Now we will prove some properties of this relaxation and bound its integrality gap.

Lemma 46. *Given $\varepsilon > 0$, for $k = \frac{1}{\varepsilon^2}$ the k-edge-cut LP restricted to instances with links of length at most $\frac{1}{\varepsilon}$ has integrality gap at most $(1 + 2\varepsilon)$.*

Proof. We will assume w.l.o.g. that the set of links L contains every possible link of length 1. If it is not the case, let us include them obtaining a new set of links $L' \supseteq L$. The optimal LP value can only decrease while the size of the optimal solution cannot decrease, implying that the integrality gap can only increase due to this operation. To see this last fact, assume by contradiction that there exists a solution OPT' for the new instance having strictly smaller size than OPT. Consider now a solution S consisting of $\mathsf{OPT}' \cap L$ plus a minimal set of links from L that makes S feasible (this is possible since the instance admits a feasible solution). If we in parallel iteratively contract the common links in S and OPT' we arrive to the same CacAP instance, but now the remaining links from OPT' have length 1 and the contraction of each of them reduces the number of nodes in the instance by exactly one node while the contraction of the remaining links in S reduces the number of nodes by at least 1. Thus $|S| \leq |\mathsf{OPT}'|$ which is not possible since $S \subseteq L$.

Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge-cut LP. We will construct an integral feasible solution of size at most $(1+\varepsilon)\sum_{\ell \in L} x_\ell$. To do so, we will partition the cycle into disjoint intervals as follows: We will first define an interval of size k (which we will call a *long* interval) and then an interval of size $\frac{1}{\varepsilon}$ (which we will call a *short* interval), and then continue with this procedure until it is not possible to continue. If in the end there are at most $\frac{1}{\varepsilon}$ edges we define a last short interval consisting of these remaining edges, otherwise we define a short interval consisting of the last $\frac{1}{\varepsilon}$ edges and a long interval consisting of the remaining edges (which will have size at most k). The number of short intervals is upper bounded by $1 + \left\lfloor \frac{n}{1/\varepsilon^2 + 1/\varepsilon} \right\rfloor \leq 1 + \frac{\varepsilon^2 n}{1+\varepsilon} \leq \varepsilon^2 n$ assuming w.l.o.g. that n is lower bounded by a large enough constant.

Notice that $\sum_{\ell \in L} x_\ell \geq n/2$ by a simple averaging argument over the n constraints corresponding to all the pairs of consecutive edges: every link appears in exactly two such constraints and the right-hand side of each constraint is 1. Since the total number of links of length 1 having both endpoints in a short interval is at most $\varepsilon^2 n \cdot \frac{1}{\varepsilon} = \varepsilon n \leq 2\varepsilon \sum_{\ell \in L} x_\ell$, we can add them to our solution at a negligible cost.

Consider now the set of long intervals $S_1, S_2, \ldots, S_T$. Notice that no link has endpoints in different long intervals, and hence the LP constraints associated to such intervals do not share common variables. This implies that $\sum_{\ell \in L} x_\ell \geq \sum_{i=1}^{T} |\mathsf{OPT}_{S_i}|$. Our feasible solution will consist of all the links of length 1 with both endpoints in a short interval plus the optimal solutions OPT_{S_i} for each long interval S_i. As argued before, the size of this solution is at most $(1+2\varepsilon)\sum_{\ell \in L} x_\ell$ and the feasibility of the solution follows since every $\{e, e'\}$-cut where e is in a short interval is satisfied by a link of length 1, while the remaining cuts are satisfied by the links computed optimally. □

Lemma 47. *Given $\varepsilon > 0$, for $k = \frac{1}{\varepsilon^2}$ the k-edge-cut formulation has integrality gap at most $(1+4\varepsilon)$ restricted to instances without crossing pairs of links.*

Proof. Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge-cut LP. Suppose that the instance does not contain links of length at least $\frac{1}{\varepsilon}$, then we can conclude the claim thanks to Lemma 46. Otherwise, we will pick any link of length at least $\frac{1}{\varepsilon}$ and contract it, obtaining a CacAP instance consisting of two cycles without external links (as there are no crossing links), both of size at least $\frac{1}{\varepsilon}$. If any cycle still contains some long link, we iterate this procedure. Let L_{long} be the set of long links we picked during this procedure and $C_1, C_2, \ldots, C_T$ be the set of cycles at the end. By the same argument as in Theorem 35, we have that $|L_{\text{long}}| \leq \varepsilon n \leq 2\varepsilon \sum_{\ell \in L} x_\ell$.

Applying Lemma 46 to each cycle, we obtain a feasible solution of size at most $(1+2\varepsilon)\sum_{i=1}^{T} \mathsf{OPT}_{\mathrm{LP}_i} + |L_{\mathsf{long}}|$, where LP_i is the k-edge-cut LP defined by each cycle C_i and its internal links. As there are no external links, the sum of the previous LP solutions is the optimal solution for the following LP:

$$\begin{aligned} \min \quad & \sum_{\ell \in L \setminus L_{\mathsf{long}}} x_\ell & & \\ s.t. \quad & \sum_{\ell \in L_S} x_\ell \geq |\mathsf{OPT}_S| & \forall i \in \{1,\dots,T\}, \forall S \subseteq E(C_i), |S| \leq \tfrac{1}{\varepsilon^2} \\ & 0 \leq x_\ell \leq 1 & \forall \ell \in L \setminus L_{\mathsf{long}} \end{aligned}$$

The set of constraints of this LP is a subset of the constraints of the original LP as links in L_{long} do not appear in these constraints and the set of variables is a subset of the original one. Thus we have $\sum_{i=1}^{T} \mathsf{OPT}_{\mathrm{LP}_i} \leq \sum_{\ell \in L} x_\ell$, and then we can conclude that the constructed solution has size at most $(1+4\varepsilon)\sum_{\ell \in L} x_\ell$. □

Following the proof of Theorem 35 plus the previous results we can get the following bound on the integrality gap for general instances of CycAP.

Corollary 48. *For any $\varepsilon > 0$, the integrality gap of the k-edge-cut LP for $k = \frac{1}{\varepsilon^2}$ is at most $\frac{3}{2} + O(\varepsilon)$.*

Proof. Let $X = (x_\ell)_{\ell \in L}$ be an optimal solution for the k-edge cut LP and consider the output of the $\left(\frac{3}{2}+\varepsilon\right)$-approximation from Section 3.2 decomposed into L_{long}, L_{ext} and L_{short} as in the proof of Theorem 35. As argued before, we know that $\sum_{\ell \in L} x_\ell \geq \frac{n}{2}$ and analogously to the proof of Lemma 47 we have that $|L_{\mathsf{short}}| \leq (1+2\varepsilon)\sum_{\ell \in L} x_\ell$. Hence essentially the same analysis as in Theorem 35 provides the same bound of $3/2 + O(\varepsilon)$ up to an extra $(1+\varepsilon)$ factor. □

Chapter 4

The Cactus Augmentation Problem

In this section we present our improved approximation algorithm for the Cactus Augmentation problem (CacAP). As mentioned earlier, this implies an improved approximation algorithm for the Connectivity Augmentation problem (CAP).

Our result is based on a reduction to the (cardinality) Steiner tree problem by Basavaraju et al. [2014], which is described later. The authors use this connection to design improved parameterized algorithms (see also Marx and Végh [2015] for a related result). Recall that in the *Steiner Tree* problem we are given an undirected graph $G_{ST} = (T \cup S, E_{ST})$, where T is a set of t *terminals* and S a set of *Steiner nodes*. Our goal is to find a tree (*Steiner tree*) $OPT_{ST} = (T \cup A, F)$ that contains all the terminals (and possibly a subset of Steiner nodes A) and has the minimum possible number of edges $|OPT_{ST}|$. Basavaraju et al. observed that, given a CacAP instance $(G = (V, E), L)$, it is possible to construct (in polynomial time) an *equivalent* Steiner Tree instance $G_{ST} = (T \cup L, E_{ST})$. Here T corresponds to the nodes of degree 2 in G, L are the Steiner nodes, and the edges E_{ST} are defined properly. In particular, an optimal solution to G_{ST} induces an optimal solution to (G, L) and vice versa. An example of the reduction is given in Figure 4.2.

Unfortunately, this reduction is not approximation-preserving. In particular, by working out the simple details (see also Section 4.1), one obtains that a ρ_{ST}-approximation for Steiner Tree implies a $\rho \leq 3\rho_{ST} - 2$ approximation for CacAP. The current best value of ρ_{ST} is $\ln 4 + \varepsilon < 1.39$ due to Byrka et al. [2013]. Hence this is not good enough to obtain $\rho < 2$[1]. However the main

[1]One would need $\rho_{ST} < \frac{4}{3}$ here. Notice that this is not ruled out by the current lower

observation here is that after the reduction we obtain an instance of Steiner Tree which is more structured than an arbitrary instance of this problem. This allows us to use the same algorithm as Byrka et al. [2013], but with a refined analysis that takes advantage of the special structures of our Steiner tree instances. Using this we break the barrier of 2 for CacAP for the first time. More precisely, We show that:

Theorem 49. *For any constant $\varepsilon > 0$, there is a polynomial-time $2\ln(4) - \frac{967}{1120} + \varepsilon < 1.9092 + \varepsilon$ approximation algorithm for the Cactus Augmentation Problem.*

Corollary 50. *For any constant $\varepsilon > 0$, there is a polynomial-time $2\ln(4) - \frac{967}{1120} + \varepsilon < 1.9092 + \varepsilon$ approximation algorithm for the Connectivity Augmentation Problem.*

Proof. It follows directly from Theorem 49 and the reduction to CacAP implied by Dinitz et al. [1976]. □

We remark that, from a technical point of view, our result deviates quite substantially from prior approximation algorithms for TAP. The first improvements on a 2 approximation where achieved via greedy-style algorithms and a complex case analysis (see Even et al. [2009]; Kortsarz and Nutov [2016a,b]; Nagamochi [2003]). More recent approaches are based on rounding stronger and stronger LP (or SDP) relaxations for the problem (Adjiashvili [2017]; Cheriyan and Gao [2018a,b]; Fiorini et al. [2018]; Grandoni et al. [2018]). We also use an LP-based rounding algorithm, which is however defined for a generic Steiner tree instance (while the properties of TAP are used only in the analysis).

4.1 Steiner Tree and Connectivity Augmentation

In this section we describe the mentioned reduction by Basavaraju et al. [2014] from CacAP to Steiner tree.

Consider a CacAP instance $(G = (V, E), L)$. For a link $\ell = (v_0, v_{q+1})$, let $v_1, \ldots, v_q$ be the sequence of nodes of degree at least 4 other than v_0 and v_{q+1} that lie along every simple v_0-v_{q+1} path. Notice that each pair $\ell_i = \{v_i, v_{i+1}\}$ lies along a distinct cycle C_i visited by the mentioned path. We call each such ℓ_i the *projection* of ℓ on C_i. (See figure 4.1 for an example)

Consider two links $\ell = \{x, y\}$ and $\ell' = \{x', y'\}$ that have endpoints in the same cycle C. Then we say that ℓ and ℓ' *interact* if one of the following two

bounds on the approximability of Steiner tree.

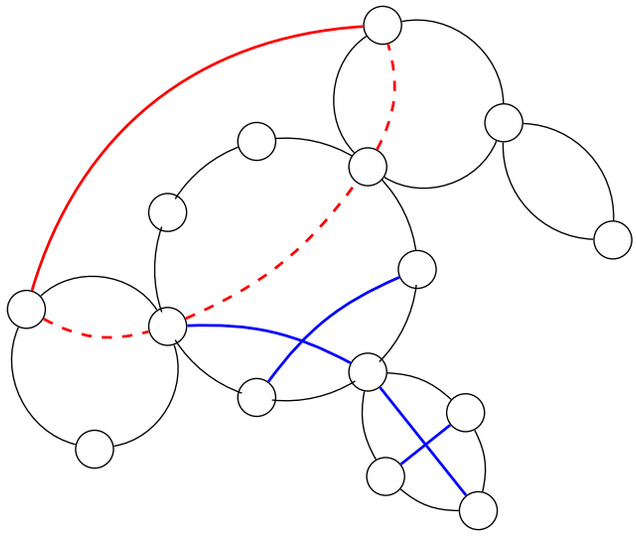


Figure 4.1. An example of projection. The dashed red edges are the projection of the red link.

conditions hold: (1) they share one endpoint or (2) taking one simple x-y path P along C, P contains exactly one node in $\{x', y'\}$ as an internal node. We say that any two links ℓ and ℓ' *interact* if there exists a projection ℓ_i of ℓ and a projection ℓ'_j of ℓ' such that ℓ_i and ℓ'_j belong to the same cycle and interact. See Figure 4.2 (left) for an example. From (G, L) we construct a Steiner tree instance $G_{ST} = (T \cup S, E_{ST})$ as follows. For each one of the t nodes v of degree 2 in G, add a terminal v to T; for each link $\ell \in L$, add a Steiner node ℓ to S (i.e., $S = L$); for each $\ell \in L$ and endpoint $v \in T$ of ℓ, add $\{\ell, v\}$ to E_{ST}; finally, for any two links ℓ and ℓ' that *interact*, add $\{\ell, \ell'\}$ to E_{ST}. See Figure 4.2 (right) for an example. Note that the instance G_{ST} has the following properties:

Remark 51. *Each Steiner node is adjacent to at most* 2 *terminals.*

Remark 52. *The neighbors of each terminal are Steiner nodes and form a clique.*

We will critically exploit the following lemma sketched in Basavaraju et al. [2014] (Lemma 1). For the sake of completeness we give a (more detailed) proof of it.

Lemma 53 (Basavaraju et al. [2014])**.** *$A \subseteq L$ is a feasible solution to a CacAP instance (G, L) iff, in the corresponding Steiner tree instance $G_{ST} = (T \cup L, E_{ST})$, $G_{ST}[T \cup A]$ is connected.*

Proof. $\Leftarrow$ Assume by contradiction that A is not a feasible CacAP solution. Then there exists a 2-edge cut $\{e_1, e_2\}$, for two edges e_1, e_2 belonging to some cycle C of G, which is not covered by any link in A. Let $G_L = (V_L, E_L)$ and $G_R = (V_R, E_R)$ be the two (vertex disjoint) connected components identified by this cut. Let also t_L and t_R be any two degree 2 nodes in V_L and V_R, respectively. (Observe that these nodes must exist.) By assumption there exists a (simple) path $P = t_L, \ell_1, \ldots, \ell_q, t_R$ between t_L and t_R in $G_{ST}[T \cup A]$, where all ℓ_i's are link nodes. Since $\{e_1, e_2\}$ is not covered, each such link has both endpoints either in V_L or in V_R. Furthermore, ℓ_1 and ℓ_q have one endpoint in V_L and V_R, resp. Hence there must be two consecutive links ℓ_i and ℓ_{i+1} where ℓ_i has both endpoints in V_L and ℓ_{i+1} both endpoints in V_R. These links cannot be interacting, therefore contradicting the fact that $\{\ell_i, \ell_{i+1}\}$ is an edge of G_{ST}.

$\Rightarrow$ We first observe that, w.l.o.g., we can replace each link ℓ with its projections $proj(\ell)$. The feasibility of A is preserved. The same holds for the connected components of $G_{ST}[T \cup A]$ since the links in $proj(\ell)$ induce a path in G_{ST}. Thus for simplicity we assume that all links in A have both their endpoints in the same cycle. Let $C_1, \ldots, C_k$ be the cycles of G. For any cycle C_i of the cactus G let A_i be the set of links in A with both their endpoints in C_i. The following lemma shows that $G_{ST}[A_i]$ is connected.

Lemma 54. *Let $G = (V, E)$ be an input cactus of CacAP which consists of exactly one cycle and let A be a feasible solution for G. Then $G_{ST}[A]$ is connected.*

Proof. Assume that $G_{ST}[A]$ is not connected. Then A can be partitioned in L_R and L_B, such that for any $l_R \in L_R$ and $l_B \in L_B$, l_R and l_B do not interact. We call the links in L_R red links and the links in L_B blue links. We can also partition V in V_R and V_B, such that the endpoints of red links belong to V_R and the endpoints of blue links belongs to V_B. Therefore we call V_B and V_R, blue vertices and red vertices respectively.

Let $V_1, V_2, \ldots, V_{2k}$ be the partition of vertices of the cycle G into maximal consecutive blocks of vertices of the same color, so that $V_1 \cup V_3 \cup \cdots \cup V_{2k-1} = V_R$ and $V_2 \cup V_4 \cup \cdots \cup V_{2k} = V_B$.

We say that a link $\ell = \{u, w\} \in A$ is *nice*, if u and v belong to different blocks V_i and V_j, $i \neq j$. We say that an edge $e = \{u, v\} \in E$ is *colorful* if u is red and v is blue or vice versa. Note that G has precisely $2k$ colorful edges. If there is no nice link in A, then any pair of colorful edges of G is not covered by A, which is a contradiction.

Assume that $\ell = \{u, v\} \in A$ is a nice link, such that the distance between u and v in the cycle G is minimum. Assume that $u \in V_1$ and $v \in V_{2x+1}$ (and

therefore these are red vertices) and also that the vertices of V_2 are in the shortest path from u to v. Now let e_1 and e_2 be the colorful edges such that exactly one of their endpoints is in V_2. Now we show that the cut formed by e_1 and e_2 is not covered by A.

Assume that $\{e_1, e_2\}$ is covered, then there should be a link $\ell_1 = (w, z)$ such that $w \in V_2$ and $z \notin V_2$. Then either this link is a nice link that interact with ℓ, which is a contradiction since $\ell \in L_R$ and $\ell_1 \in L_B$, or ℓ_1 is a nice link such that the distance of w and z is less than the distance of u and v.

□

For every pair of cycles C_i and C_j that share a vertex v, there is a link $\ell_i \in A_i$ and $\ell_j \in A_j$ which are incident to v, thus ℓ_i and ℓ_j interact. We can conclude that $G_{ST}[A]$ is connected. Finally, since A is feasible, there exists at least one link $\ell \in A$ incident to each node t of degree 2 in G, which implies that the edge $\{\ell, t\}$ belongs to E_{ST}. Thus $G_{ST}[T \cup A]$ is also connected.

□

Notice that the above reduction is not approximation-preserving. Still, we can state the following.

Corollary 55. *The optimum solution OPT to the input CacAP instance, induces a solution OPT_{ST} of cost $|OPT_{ST}| = |OPT| + t - 1$ for the associated Steiner tree instance. Vice versa, given a solution APX_{ST} to the Steiner tree instance, one can construct in polynomial time a solution APX to the input CacAP instance with $|APX| = |APX_{ST}| - t + 1$.*

Proof. Both claims follow directly from Lemma 53. For the first claim, it is sufficient to observe that a spanning tree of $G_{ST}[T \cup OPT]$ contains $t + |OPT| - 1$ edges. For the second claim, observe that the Steiner nodes in APX_{ST} induce a feasible solution to CacAP. The claim follows since $|APX_{ST}| = s + t - 1$, where s is the number of Steiner nodes in APX_{ST}. □

We will exploit also the following simple fact.

Lemma 56. *There is a feasible solution OPT_{ST} to the Steiner tree instance with $|OPT_{ST}| = |OPT| + t - 1$ where terminals have degree exactly 1.*

Proof. Given any feasible solution ST to the problem, we can transform it into a solution ST' of the same cost where some terminal v of degree $d(v) \geq 2$ in ST has degree $d(v) - 1$ in ST'. In order to do that, consider any terminal v adjacent to two Steiner nodes ℓ and ℓ' in ST. By Remark 52, ℓ and ℓ' are

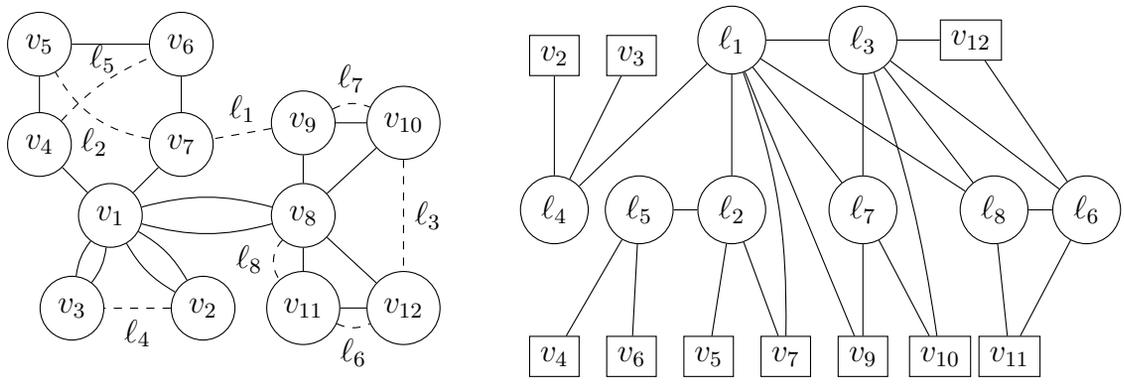


Figure 4.2. **(left)** Instance of CacAP, where dashed edges denote links. The projections of ℓ_1 are $\{v_7, v_1\}$, $\{v_1, v_8\}$ and $\{v_8, v_9\}$. Link ℓ_2 is interacting with ℓ_1 and ℓ_5. **(right)** The Corresponding Steiner tree instance, where square nodes denote terminals.

adjacent. Hence $ST' := ST \cup \{\ell, \ell'\} \setminus \{v, \ell'\}$ is a feasible Steiner tree of the same cost and with the desired property.

By iteratively applying the above process to the solution OPT_{ST} guaranteed by Corollary 55 one obtains the desired solution. □

As mentioned earlier, a ρ_{ST} approximation for Steiner tree (used as a *black box*) provides a $3\rho_{ST} - 2$ approximation for CacAP by the above construction. Indeed, the Steiner tree instance has cost at most $|OPT|+t-1$ by Corollary 55, hence an approximate solution APX_{ST} would cost at most $\rho_{ST}(|OPT|+t-1)$. By the same corollary, we can convert this into a solution APX to CacAP of cost at most $\rho_{ST}(|OPT| + t - 1) - t + 1$. Next observe that $|OPT| \geq t/2$. Indeed, any node of degree 2 in the CacAP instance needs to have at least one link incident to it in a feasible solution, and a link can be incident to at most 2 such nodes. Thus $|APX| \leq 3\rho_{ST}|OPT| - 2|OPT|$. In order to improve on this simple bound, we will have to *open the box*.

4.1.1 Steiner Tree via Iterative Randomized Rounding

As we mentioned in the introduction, the current best $(\ln 4 + \varepsilon)$-approximate Steiner tree algorithm from Byrka et al. [2013], used as a black box, is not good enough to break the 2-approximation barrier for CacAP. However, it turns out that the same algorithm achieves this goal in combination with a different analysis that exploits the properties of the specific Steiner tree instances arising from CacAP.

We next sketch the basic properties of the algorithm and analysis in Byrka et al. [2013] that we need here. A more detailed description is given in Section A in the Appendix for the sake of completeness. The authors of Byrka et al. [2013] consider an LP relaxation DCR_k for the problem based on *directed k-components* for a proper constant parameter k depending on ε. They iteratively solve this LP, sample a directed k-component C with probability proportional to the LP values, and contract C. The process ends when all terminals are contracted into one node. This algorithm can be derandomized, and the deterministic version is good enough for our application. We do not need more details about this algorithm, other than that it runs in polynomial time.

In the analysis the authors of Byrka et al. [2013] consider any feasible Steiner tree $ST = (T \cup A, F)$, which is seen as rooted at some arbitrary node r. Then the authors define a *marking scheme* where some child edge of each internal (Steiner) node is marked. A given marking scheme defines a *witness set* $W(e)$ for each edge e: this consists of pairs of terminals $\{t', t''\}$ such that the (simple) t'-t'' path in ST contains e and precisely one unmarked edge. We let $w(e) = |W(e)|$. Notice that marked edges form a forest spanning all nodes, with exactly one terminal in each connected component. Therefore, each unmarked edge e is on a single path only, hence $w(e) = 1$. Then the authors prove the following, where $H_i := 1 + \frac{1}{2} + \ldots + \frac{1}{i}$ is the i-th harmonic number.

Lemma 57 (Byrka et al. [2013])**.** *For any feasible Steiner tree $ST = (T \cup A, F)$ and marking scheme, for a large enough parameter $k = O_\varepsilon(1)$, the cost of the solution computed by the above algorithm is at most $(1 + \varepsilon) \sum_{e \in F} E[H_{w(e)}]$.*

4.2 An Improved CacAP Approximation Algorithm

In this section we present our improved approximation for CacAP. The algorithm is rather simple: we just build the Steiner tree instance $G_{ST} = (T \cup L, E_{ST})$ associated with the input CacAP instance (G, L) and compute an approximate solution APX_{ST} to G via the algorithm in Byrka et al. [2013] sketched in Section 4.1.1. Then we derive from APX_{ST} a feasible solution APX to the input CacAP instance as described in Corollary 55. We let apx denote the approximation ratio of this algorithm.

In Section 4.2.1 we describe our alternative marking scheme and prove some of its properties. In Section 4.2.2 we complete the analysis of the approximation factor.

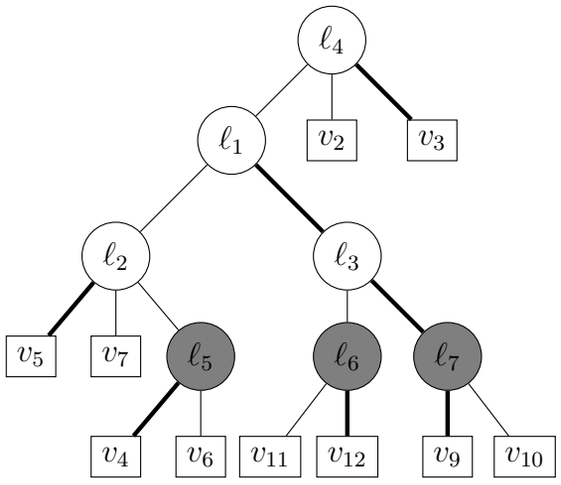


Figure 4.3. A feasible Steiner tree for the instance of Figure 4.2, which happens to be well-structured. Bold edges denote a possible marking. One has $m(\ell_3) = e := \{\ell_3, \ell_7\}$, and $W(e)$ contains $\{v_9, v_{12}\}$, $\{v_9, v_5\}$ and $\{v_9, v_3\}$. Notice that $w(e) = |W(e)| = d(\ell_3) + d(\ell_1) - 1$. Leaf-Steiner nodes are drawn in grey. Here ℓ_2 (resp., ℓ_3) is a good (resp., bad) father. Consequently ℓ_5 (resp., ℓ_6) is good (resp., bad). A feasible grouping is $g(\ell_2) = \{\ell_2\}$, $g(\ell_3) = \{\ell_3, \ell_7\}$, $g(\ell_1) = \{\ell_1, \ell_6\}$, $g(\ell_4) = \{\ell_4\}$, and $g(\ell_5) = \{\ell_5\}$.

4.2.1 An Alternative Marking Scheme

Recall that in the analysis of the Steiner tree approximation algorithm in Byrka et al. [2013], one can focus on a specific feasible Steiner tree ST and on a specific marking scheme (so that Steiner nodes are connected to some terminal via paths of marked edges). As feasible solution ST we consider the solution $OPT_{ST} = (T \cup OPT, F)$, of cost $|OPT| + t - 1$ and with terminals being leaves, guaranteed by Lemma 56.

We mark edges in the following way. Let us root OPT_{ST} at some Steiner node r which is adjacent to at least one terminal. For a Steiner node ℓ, we let $d(\ell)$, $s(\ell)$ and $t(\ell)$ be the number of its children, Steiner children, and terminal children, resp. In particular $d(\ell) = s(\ell) + t(\ell)$ and (by Remark 51) $t(\ell) \leq 2$.

For each link node ℓ, there are two options. If ℓ has at least one terminal child, we select one such child t uniformly at random, and mark edge $\{\ell, t\}$. Otherwise, we choose a child ℓ' of ℓ (ℓ' being a Steiner node) uniformly at random, and mark edge $\{\ell, \ell'\}$. Notice that this is obviously a feasible marking scheme. Observe also that in our marking we *favor* edges connecting Steiner nodes to terminals: this will be critical in our analysis. See Figure 4.3 for a possible marking of this type.

Let APX_{ST} be the Steiner tree computed by the algorithm. Let F_{mar} and F_{unm} be the (random) sets of marked and unmarked edges, resp., that partition F. Recall that for each $e \in F$, there exists a (random) witness set $W(e)$ of size $w(e) = |W(e)|$. Observe that each Steiner node ℓ has precisely one marked child edge $m(\ell)$. We let the *cost* $c(\ell)$ of ℓ be $E[H_{w(m(\ell))}]$. The following bound on the approximation ratio holds.

Lemma 58. $apx \leq 2\varepsilon + \frac{1+\varepsilon}{|OPT|} \sum_{\ell \in OPT} c(\ell)$.

Proof. Recall that by Lemma 57 the expected cost of the computed Steiner tree APX_{ST} is, modulo a factor $(1+\varepsilon)$, at most

$$\begin{aligned} & E[\textstyle\sum_{e\in F} H_{w(e)}] = E[\sum_{e\in F_{mar}} H_{w(e)} + \sum_{e\in F_{unm}} H_{w(e)}] \\ = \;& E[\textstyle\sum_{e\in F_{mar}} H_{w(e)} + |F_{unm}|] = E[\sum_{e\in F_{mar}} H_{w(e)}] + t - 1. \end{aligned}$$

In the second-last equality above we used the fact that $w(e) = 1$ deterministically for an unmarked edge, and in the last equality above the fact that there are precisely $|OPT|$ marked edges and consequently exactly $t-1$ unmarked ones. From APX_{ST} we derive a feasible solution APX to the input instance of cost $|APX| = |APX_{ST}| - 1 + t$ by Corollary 55. Hence

$$|APX| \leq (1+\varepsilon)(E[\sum_{e\in F_{mar}} H_{w(e)}]+t-1)-1+t \leq (1+\varepsilon)E[\sum_{e\in F_{mar}} H_{w(e)}]+2\varepsilon|OPT|.$$

In the last inequality above we used the trivial lower bound $|OPT| \geq t/2$ that we mentioned earlier. The claim follows since by definition $\sum_{e\in F_{mar}} E[H_{w(e)}] = \sum_{\ell\in OPT} c(\ell)$. □

From the above lemma, modulo factors $(1+\varepsilon)$, the approximation ratio of our algorithm is given by the *average cost* of Steiner nodes. The following lemma gives a generic upper bound on the cost for each non-root Steiner node based on the degree sequence of its ancestors[2].

Lemma 59. *Given a non-root Steiner node ℓ, let ℓ_q be the lowest proper ancestor[3] of ℓ with $t(\ell_q) > 0$. Let $\ell = \ell_1, \ell_2, \ldots, \ell_q$, $q \geq 2$, be the simple path between ℓ and ℓ_q, and let $d_i = d(\ell_i)$. Then*

$$c(\ell) = \sum_{h=1}^{q-2} \frac{(d_{h+1}-1)H_{d_1+\ldots+d_h-h+1}}{d_2 \cdot \ldots \cdot d_{h+1}} + \frac{H_{d_1+\ldots+d_{q-1}-q+2}}{d_2 \cdot \ldots \cdot d_{q-1}}.$$

[2]Observe that for the root r, $c(r) = H_{d(r)-1}$ deterministically.

[3]Observe that this ancestor exists since the root has this property by assumption.

Proof. By definition $c(\ell) = c(\ell_1) = E[H_{w(e)}]$, where $e = m(\ell_1) = \{\ell_1, \ell_0\}$ is the marked child edge of ℓ_1. Recall that $W(e)$ contains one entry for each path in the tree between terminals that contains e and precisely one unmarked edge. In our specific case, condition on $\{\ell_0, \ell_1\}, \{\ell_1, \ell_2\}, \ldots, \{\ell_{h-1}, \ell_h\}$ being a maximal sequence of consecutive marked edges. Notice that by construction $\{\ell_{q-1}, \ell_q\}$ is unmarked (since ℓ_q has a terminal child by definition), hence $h \leq q-1$. In this case $w(e) = d_1 + \ldots + d_h - (h-1)$. For $h < q-1$, the mentioned event happens with probability $\frac{1}{d_2} \cdot \ldots \cdot \frac{1}{d_h} \cdot \frac{d_{h+1}-1}{d_{h+1}}$. For $h = q-1$, this probability is $\frac{1}{d_2} \cdot \ldots \cdot \frac{1}{d_h}$. The claim follows by computing the expectation of $H_{w(e)}$. □

We next provide an upper bound on $c(\ell)$ as a function of $d(\ell)$ only. Let us define the following variant of H_i:

$$\hat{H}_i := \frac{1}{2}H_i + \frac{1}{4}H_{i+1} + \ldots = \sum_{j \geq 0} \frac{1}{2^{j+1}} H_{i+j}.$$

One has that $\hat{H}_1 = \ln(4)$ and $\hat{H}_{j+1} = 2\hat{H}_j - H_j$. Notice that, modulo an additive ε, $\hat{H}_1$ is precisely the approximation factor for Steiner tree achieved in Byrka et al. [2013]. The first few approximate values of $\hat{H}_i$ are $\hat{H}_1 < 1.3863$, $\hat{H}_2 < 1.7726$, $\hat{H}_3 < 2.0452$, $\hat{H}_4 < 2.2571$, $\hat{H}_5 < 2.4308$, $\hat{H}_6 < 2.5781$, $\hat{H}_7 < 2.7062$, and $\hat{H}_8 < 2.8195$.

The proof of the following lemma, though not entirely trivial, is mostly based on algebraic manipulations and therefore we postpone it to the appendix.

Lemma 60. *For any $\ell \in OPT$, $c(\ell) \leq \hat{H}_{d(\ell)}$.*

In next subsection we will see that for a carefully defined subset of Steiner nodes ℓ it is possible to obtain a better upper bound on $c(\ell)$ than the one provided by Lemma 60. This will be critical in our analysis since the latter bound is not strong enough.

4.2.2 Analysis of the Approximation Factor

In this section we upper bound the approximation factor *apx* as given by Lemmas 58 and 59. In order to simplify our analysis, it is convenient to focus our attention on a specific class of well-structured Steiner trees OPT_{ST} (see also Figure 4.3). The following lemma shows that this is (essentially) w.l.o.g.

Definition 61. *A rooted Steiner tree is well-structured if, for every Steiner node ℓ: (1) ℓ has at least 2 children and (2) ℓ has 0 or 2 terminal children.*

Lemma 62. *Let ρ be the supremum of $\rho(OPT_{ST}) = \frac{1}{|OPT|}\sum_{\ell \in OPT} c(\ell)$ over Steiner trees $OPT_{ST} = (T \cup OPT, F)$, and ρ_{ws} be the same quantity computed over the subset of well-structured Steiner trees OPT_{ST} of the mentioned type. Then $\rho \leq \max\{\hat{H}_1, \rho_{ws}\}$.*

Proof. Recall that in OPT_{ST} each Steiner node ℓ has at most 2 terminal children. Consider any such tree where some Steiner node ℓ' has precisely one terminal child t. Consider the tree OPT'_{ST} which is obtained from OPT_{ST} by appending to ℓ' a second terminal child t'. Observe that the value of $c(\ell)$ does not decrease for any ℓ, and it increases for $\ell = \ell'$. Thus $\rho(OPT'_{ST}) > \rho(OPT_{ST})$. Hence ρ is equal to the supremum of $\rho(OPT_{ST})$ over the subfamily of trees that satisfies (2) in Definition 61.

Now consider any tree OPT_{ST} that satisfies (2), and let $o(OPT_{ST})$ be the number of its Steiner nodes with precisely one child. We prove by induction on $o(OPT_{ST})$ that $\rho(OPT_{ST}) \leq \max\{\hat{H}_1, \rho_{ws}\}$. The claim is trivially true for $o(OPT_{ST}) = 0$ since in this case OPT_{ST} is well-structured. Assume the claim is true up to $q-1 \geq 0$, and consider $OPT_{ST} = (T \cup OPT, F)$ with $o(OPT_{ST}) = q$. Let ℓ' be any Steiner node with precisely one child ℓ''. Observe that ℓ'' has to be a Steiner node as well by (2), and that $c(\ell') \leq \hat{H}_1$ by Lemma 60. Consider the tree $OPT'_{ST} = (T \cup OPT', F')$ obtained by contracting edge (ℓ', ℓ''). We observe that OPT'_{ST} satisfies (2), $o(OPT'_{ST}) = q-1$ and $|OPT'| = |OPT| - 1$. Note also that for any Steiner node ℓ different from ℓ' and ℓ'' the value of $c(\ell)$ does not change, while for the new node $\tilde{\ell}$ resulting from the contraction one has $c(\tilde{\ell}) = c(\ell'')$. We can conclude that

$$\frac{1}{|OPT|}\sum_{\ell \in OPT} c(\ell) \leq \frac{1}{|OPT|}(\hat{H}_1 + \sum_{\ell \in OPT \setminus \{\ell''\}} c(\ell))$$

$$= \frac{1}{|OPT|}(\hat{H}_1 + \sum_{\ell \in OPT'} c(\ell)) \leq \max\{\hat{H}_1, \frac{1}{|OPT'|}\sum_{\ell \in OPT'} c(\ell)\} \leq \max\{\hat{H}_1, \rho_{ws}\},$$

where in the last inequality we used the inductive hypothesis. □

We next show an upper bound on ρ_{ws} which is strictly greater than $\hat{H}_1$. It then follows from Lemma 62 that the same upper bound holds on ρ. For this goal, we next assume that OPT_{ST} is well-structured.

The upper bound on $c(\ell)$ from Lemma 60 is not sufficient to achieve a good approximation factor. In order to achieve a tighter bound, we consider the following classification of the Steiner nodes (see also Figure 4.3).

Definition 63. *A Steiner node ℓ' is a* good father *if it has at least one terminal child (hence precisely 2 such children by the above assumptions), and a* bad father *otherwise. Each Steiner child ℓ of a good father ℓ' is* good*, and all other*

Steiner nodes are bad. *Let OPT_{gf}, OPT_{bf}, OPT_g and OPT_b denote the sets of good fathers, bad fathers, good nodes and bad nodes, resp.*

Notice that the above classification is not affected by the random choices in the marking scheme. For good nodes, the analysis of the cost can be refined as follows.

Lemma 64. *For any $\ell \in OPT_g$, $c(\ell) \leq H_{d(\ell)}$.*

Proof. Suppose ℓ has a parent ℓ', which is a good father by definition. This implies that the edge (ℓ', ℓ) is deterministically unmarked, hence $w(m(\ell)) = d(\ell)$ deterministically. If ℓ has no parent (i.e., it is the root r), then $w(m(\ell)) = d(\ell) - 1$. The claim follows. □

Putting everything together, we obtain the following.

Lemma 65. *$apx \leq 2\varepsilon + \frac{1+\varepsilon}{|OPT|} \sum_{\ell \in OPT} c'(\ell)$ where $c'(\ell) = \begin{cases} H_{d(\ell)} & \text{if } \ell \in OPT_g; \\ \hat{H}_{d(\ell)} & \text{if } \ell \in OPT_b. \end{cases}$*

Proof. It follows from Lemma 58, by replacing $c(\ell)$ as in Lemma 59 with the upper bounds given by Lemmas 60 and 64. □

We rewrite the upper bound from Lemma 65 as follows. Let $p \in [0, \hat{H}_2 - H_2]$ be a parameter to be fixed later. Intuitively, each good Steiner node $\ell \in OPT_g$ pays a *present* p to its (good) father $\ell' \in OPT_{gf}$ to thank ℓ' for making itself good. This increases the cost of ℓ by p. Symmetrically, each good father $\ell' \in OPT_{gf}$ collects presents from its (good) Steiner children and uses them to lower its own cost. Clearly by definition the total modification of the cost is zero. Let us call $c''(\ell)$ the modified costs. Then one obtains the following equality:

$$\frac{1}{|OPT|} \sum_{\ell \in OPT} c'(\ell) = \frac{1}{|OPT|} \sum_{\ell \in OPT} c''(\ell) \tag{4.1}$$

where

$$c''(\ell) = \begin{cases} H_{d(\ell)} + p - s(\ell)p & \text{if } \ell \in OPT_g \cap OPT_{gf}; \\ H_{d(\ell)} + p & \text{if } \ell \in OPT_g \cap OPT_{bf}; \\ \hat{H}_{d(\ell)} - s(\ell)p & \text{if } \ell \in OPT_b \cap OPT_{gf}; \\ \hat{H}_{d(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{bf}. \end{cases}$$

In order to upper bound (4.1), we partition OPT into groups of nodes as follows (see also Figure 4.3).

Definition 66. *A Steiner node ℓ is* leaf-Steiner *if it has no Steiner children (i.e., $d(\ell) = t(\ell) = 2$) and* internal-Steiner *otherwise (i.e., $s(\ell) > 0$). We let OPT_{lf} and OPT_{in} be the set of leaf-Steiner and internal-Steiner nodes, resp.*

We associate to each $\ell \in OPT_{in}$ a distinct subset $OPT_{lf}(\ell)$ of precisely $s(\ell) - 1$ leaf-Steiner nodes, and let $g(\ell) = \{\ell\} \cup OPT_{lf}(\ell)$ be the *group* of ℓ. The mapping is constructed iteratively in a bottom-up fashion as follows. Initially all Steiner nodes are unprocessed. We maintain the invariant that the subtree rooted at an unprocessed leaf-Steiner node or at a processed node with unprocessed parent contains precisely one unprocessed leaf-Steiner node. Clearly the invariant holds at the beginning of the process. We consider any unprocessed internal-Steiner node ℓ whose Steiner descendants are either processed or leaf-Steiner nodes. By the invariant, each subtree rooted at a Steiner child of ℓ (which is either an unprocessed leaf-Steiner node or a processed internal-Steiner node) contains one unprocessed leaf-Steiner node. Among this set of $s(\ell)$ unprocessed leaf-Steiner nodes, we select arbitrarily a set $OPT_{lf}(\ell)$ of size $s(\ell) - 1$ and set $g(\ell) = \{\ell\} \cup OPT_{lf}(\ell)$. All nodes in $g(\ell)$ are marked as processed. Observe that the subtree rooted at ℓ still contains an unprocessed leaf-Steiner node, hence the invariant is preserved in the following steps. At the end of the process (i.e., after processing the root r) there will be precisely one leaf-Steiner node ℓ^* which is still unprocessed, which forms a special group $g(\ell^*) = \{\ell^*\}$ on its own. Notice that the groups define a partition of OPT. In particular, $OPT = \{\ell^*\} \cup \bigcup_{\ell \in OPT_{in}} g(\ell)$. Notice also that $|g(\ell)| = s(\ell)$ for all $\ell \in OPT_{in}$ (while $|g(\ell^*)| = 1$).

Let $a(\ell)$ be the average value of $c''(\cdot)$ over the elements of $g(\ell)$. Then obviously the maximum value of $a(\ell)$ over the groups upper bounds the average value of $c''(\cdot)$:

$$\frac{1}{|OPT|} \sum_{\ell \in OPT} c''(\ell) \leq \max_{\ell \in OPT_{in} \cup \{\ell^*\}} \{a(\ell)\}. \tag{4.2}$$

For $\ell = \ell^*$ one has that $a(\ell^*) = c''(\ell^*) = \hat{H}_2$ if ℓ^* is bad, and $a(\ell^*) = c''(\ell^*) = H_2 + p \leq \hat{H}_2$ otherwise. For the other groups $g(\ell)$, there is always a subset of $s(\ell) - 1$ leaves whose contribution to the cost is at most $\hat{H}_2$ each by the same argument as above. Furthermore, we have to add the cost $c''(\ell)$. We can

conclude that:

$$a(\ell) \leq \begin{cases} a_1(s(\ell)) := \frac{H_{s(\ell)+2}+p-s(\ell)p+(s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_g \cap OPT_{gf}; \\ a_2(s(\ell)) := \frac{H_{s(\ell)}+p+(s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_g \cap OPT_{bf}; \\ a_3(s(\ell)) := \frac{\hat{H}_{s(\ell)+2}-s(\ell)p+(s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{gf}; \\ a_4(s(\ell)) := \frac{\hat{H}_{s(\ell)}+(s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{bf}; \\ \hat{H}_2 & \text{if } \ell = \ell^*. \end{cases}$$

In the first and third case above we used the fact that $d(\ell) = s(\ell) + 2$ (ℓ is a good father, hence has 2 terminal children), while in the second and fourth case the fact that $d(\ell) = s(\ell)$ (ℓ is a bad father, hence has no terminal child).

We are now ready to prove the main result of this paper.

Proof of Theorem 49. Consider the above algorithm. Combining Lemma 65 with (4.1) and (4.2) one gets

$$apx \leq 2\varepsilon + (1+\varepsilon) \max_{i \geq 1} \{\hat{H}_2, a_1(i), a_2(i), a_3(i), a_4(i)\}. \tag{4.3}$$

We need the following technical result (proof in Appendix).

Claim 67. *For any $p \in [0, \hat{H}_2 - H_2]$, the maximum of $a_1(i)$, $a_2(i)$, $a_3(i)$, and $a_4(i)$ is achieved for i at most 6, 8, 6 and 8, resp.*

From (4.3) and Claim 67, for any $p \in [0, \hat{H}_2 - H_2]$, one has

$$apx \leq 2\varepsilon + (1+\varepsilon) \max\{\hat{H}_2, \max_{1 \leq i \leq 6}\{a_1(i)\}, \max_{1 \leq i \leq 8}\{a_2(i)\}, \max_{1 \leq i \leq 6}\{a_3(i)\}, \max_{1 \leq i \leq 8}\{a_4(i)\}\}. \tag{4.4}$$

Numerically the minimum of the right-hand side of (4.4) is achieved for $p \simeq 0.135$, and the two largest values inside the maximum turn out to be $a_2(7)$ and $a_3(1)$. By imposing $\frac{H_7+6\hat{H}_2+p}{7} = a_2(7) = a_3(1) = \hat{H}_3 - p$ one gets $p = \frac{7\hat{H}_3 - H_7 - 6\hat{H}_2}{8}$. For that value of p the value of the maximum is precisely $\frac{H_7+6\hat{H}_2+\hat{H}_3}{8} = 2\ln 4 - \frac{967}{1120}$. The claim follows by scaling ε properly. □

Chapter 5

Conclusions

In this thesis, we discussed several variants of Network Augmentation problems, with the focus on the Connectivity Augmentation problem.

In the first attempt, we identified Cycle Augmentation as a challenging special case. We showed that CycAP does not admit a PTAS using a reduction to the 3-Dimensional Matching problem and then we presented two approximation algorithms for this problem. Our first algorithm was simple and fast with the approximation factor of $\frac{5}{3}$ using a greedy approach. Then using a dynamic programming we achieved a $(\frac{3}{2}+\varepsilon)$-approximation algorithm for this problem and showed that this problem is FPT on the length of the longest link.

In chapter 4 we presented the first approximation algorithm with factor below 2 for CAP using a reduction to the Steiner Tree problem.

In the next section we propose some relevant open problems and interesting directions for future research.

5.1 Open Problems

There are many Survivable Network Design problems that are worth investigating in terms of approximability.
Even after the worthy result of Cecchetto et al. [2021], still it is possible to improve the approximability of CAP. In particular, $\frac{4}{3}$ seems to be a challenging but reasonable goal. Proving an α-inapproximability result for some α significantly above 1 is also of great importance.

A major problem that needs a substantial progress is WTAP. This problem has received a lot of attention in the past decades, however, there has been no

success in breaking the 2-approximation barrier[1].

The k-ECSS problem is another well-motivated problem that leaves room for improvement. Obtaining an approximation algorithm of factor below 2 for the weighted case would be an amazing breakthrough. Also improving the current-best $\frac{4}{3}$ approximation factor for 2-ECSS. There have been a few attempts to improve the mentioned factor down to $\frac{5}{4}$, however they all contained technical bugs.

The last open problem that we wish to mention is related to FAP, an interesting and natural generalization of TAP and 2-ECSS. Breaching the 2-approximation barrier for this problem is an intriguing open problem. One possibility is to consider first some interesting special cases of this problem, e.g. when the forest is a collection of paths (the *Path Augmentation* problem).

[1]We recall that recently Traub and Zenklusen [2021] published a preprint (on arXiv) which claims a $(1 + \ln 2 + \epsilon)$-approximation for WTAP.

Appendix A

Details of the Steiner Tree Approximation Algorithm of Byrka et al. [2013]

We will briefly discuss the $\ln(4) + \epsilon$ approximation algorithm from Byrka et al. [2013] for the Steiner tree problem. For a complete presentation of the Steiner tree algorithm we refer to the original paper (Byrka et al. [2013]). The algorithm is based on the Directed Component Relaxation (DCR) of the Steiner tree problem.

$$\min \quad \sum_{C \in \mathscr{C}} c(C) x_C \qquad \text{(DCR)} \tag{A.1}$$

$$\text{s.t.} \quad \sum_{C \in \delta^+_{\mathscr{C}}(U)} x_C \geq 1 \qquad \forall \emptyset \neq U \subseteq T \setminus \{r\} \tag{A.2}$$

$$x_C \geq 0 \qquad \forall\, C \in \mathscr{C}. \tag{A.3}$$

Here $\mathscr{C}$ is a set of directed components, where each directed component C is a minimum-cost Steiner tree (of cost $c(C)$) over a subset of terminals. Furthermore, the leaves of C are precisely its terminals, and C is directed towards a specific terminal: the latter node is the sink of C, and the remaining terminals are the sources of C. Intuitively, our goal is to buy a minimum-cost subset of directed components so that they induce a directed path from each terminal to the root. In more detail, for any cut U that separates some non-root terminal from the root, let $\delta^+_{\mathscr{C}}(U)$ be the set of components with some source in U and the sink not in U. Then every feasible solution has to buy some component in $\delta^+_{\mathscr{C}}(U)$. The DCR relaxation follows naturally.

After restricting DCR to solutions that only use components with at most k terminals we obtain DCR_k. For constant k, DCR_k has a polynomial number of variables. Furthermore, the separation problem can be solved in polynomial time via a reduction to minimum cut. Therefore DCR_k can be solved in polynomial time. Moreover, the value of DCR_k is known to be a $(1+\epsilon)$-approximation of the value of DCR for large enough $k = O_\varepsilon(1)$.

The iterative randomised rounding algorithm from Byrka et al. [2013], until all terminals are connected to the root, in iterations $t = 1, 2, 3 \ldots$, does the following:

- solve DCR_k for the current instance of the Steiner tree problem to get x^t;
- sample a component C^t from $\mathscr{C}_k$ with probability proportional to x^t_C;
- contract the sampled component C^t.

For the ease of the analysis, by adding dummy components w.l.o.g, one may assume that the total number of components in the fractional solution remains constant across the iterations of the algorithm, i.e., $\sum_{C \in \mathscr{C}} x^t_C = M$ for a proper M for all $t = 1, 2, \ldots$. It is argued that after t iterations of the algorithm, having bought the first t sampled components, the residual instance of the problem is expected to be less costly. To this end a reference solution S^t is constructed such that $S^t \cup \bigcup_{t'=1}^{t-1} C^t$ connects all the terminals. The initial reference solution $S^1 = OPT_{ST}$ is an optimal solution to the Steiner tree instance of cost *opt*. Consecutive reference solutions $S^2, S^3, \ldots$ are obtained by gradually deleting edges that are no longer necessary due to the connectivity provided by the already sampled components.

Key to estimate the expected cost of the final solution is to bound the number of iterations until a particular edge $e \in S^1$ can be removed. Define $D(e) = \max\{t | e \in S^t\}$. In Byrka et al. [2013] (proof of Theorem 21) it is shown that there exist a randomised process of constructing reference solutions $S^1, S^2, \ldots$ such that $E[D(e)] \leq \ln(4) \cdot M$, which allows one to bound the total expected cost of sampled components as $E\left[\sum_{t \geq 1} c(C^t)\right] \leq (ln(4) + \epsilon) \cdot opt$. Note that the above *per-edge* guaranty allows for easily handling arbitrary costs of individual edges. In our application to (unweighted) CacAP, we need to average over multiple edges to achieve a good enough bound.

A.1 Witness Tree and Witness Sets

We next slightly abuse notation and sometimes denote in the same way a tree and its set of edges. The construction of reference solutions $S^1, S^2, \ldots$ is not trivial. It involves:

- construction of a terminal spanning tree W, called the *witness tree*, based on randomised marking (selection) of a subset of edges of S^1. Each edge e of S^1 is associated with a proper subset $W(e) \subseteq W$ (witness set of e);
- randomised deletion of a proper subset of W in response to selecting a particular component C^t in iteration t;
- removing an edge e from S^t when all edges $W(e)$ have already been deleted.

In the following we discuss the main idea behind and the key properties of each of the three above mentioned processes. We also pin-point the element of the analysis that can be modified in order to utilise the specific properties of the instance we obtain from the reduction from CacAP.

Construction of the witness tree. The high level idea behind the witness tree is that we need to always satisfy the condition that $S^t \cup \bigcup_{t'=1}^{t-1} C^t$ connects all the terminals, which is that the remaining fragments of the initial reference solution S^1 together with the already sampled components must provide sufficient connectivity. To this end a simpler object providing connectivity is constructed. It is an auxiliary tree W whose node set is the terminals of the instance (edges of W are independent of the edges of the input graph). It will be easier to delete edges from W in response to sampling components rather than deleting them directly from S^t.

We will now discuss methods to construct W. Intuitively, removing edges from a Steiner tree (in response to receiving connectivity from a component) is directly possible for only a subset of edges of the Steiner tree. In particular it appears more difficult to remove a Steiner vertex (and hence a path connecting a Steiner vertex to a terminal). This is related to the concept of *Loss* and *Loss contracting algorithms* (see, e.g., Robins and Zelikovsky [2005]), where one accepts that the cost of the system of paths connecting Steiner nodes to terminals is not removable.

Consider the following procedure: For each component[1] S' of the Steiner tree S^1 select a single Steiner vertex s and draw the component as a tree

[1]Recall that a full component is a maximal subtree whose terminals are exactly its leaves.

rooted in s. For every Steiner vertex of S' select and mark a single edge going down (away from s). Note that for each Steiner vertex v the marked edges will form a unique path towards a leaf containing terminal $t(v)$. Note also that connected components formed by the marked edges will all have a single terminal node. Construct $W(S')$ by adding to $E(W(S'))$ an edge $\{t(u), t(v)\}$ for each unmarked edge $\{u, v\}$ of S'.[2] Observe that the above constructed graph $W(S')$ is a tree spanning the terminals of S'. By repeating this procedure for all full components of S^1 we obtain tree W spanning all terminals of the Steiner tree instance.

So far we did not specify how to select the edge below Steiner node $v \in S'$ to be marked. In Byrka et al. [2013] the tree was assumed to be binary, and the edge would be selected at random by tossing a fair coin. In the current paper we use a different marking strategy as discussed in Section 4.2.1.

Marking edges of the witness tree. When edges of the witness tree W become unnecessary, we mark them. We keep the invariant that the unmarked edges of W together with the already collected components are sufficient to connect all terminals. Still, given a fixed collection of the already sampled components, the choice of which edges of W to mark is not obvious. In Byrka et al. [2013] a randomised marking scheme was considered. It was shown (Lemma 19 in Byrka et al. [2013]) that there exists a random process marking edges in W in response to sampled components, such that for every edge $e \in W$ not marked until iteration t the probability that it is marked in iteration t is at least $1/M$. In the current work we continue using the mentioned "uniform" witness tree marking process, and utilise the following lemma.

Lemma 68 (lemma 20 in Byrka et al. [2013])**.** *Let $\tilde{W} \subseteq W$. Then the expected number of iterations until all edges in $\tilde{W}$ are marked is at most $H_{|\tilde{W}|} \cdot M$.*

Removing edges of the reference tree S^t. Which edges of the reference tree can be removed? Clearly it suffices if S^t provides the same terminal connectivity as the unmarked edges of the witness tree W. Note that a single edge $e \in W$ corresponds to a single path $p(e)$ in S^1. It then suffices to keep the edges of S^1 that occur in a path $p(e)$ of at least one unmarked edge $e \in W$.

[2]Note that in Byrka et al. [2013] the role of marked and unmarked edges was reversed. It was irrelevant for the analysis in Byrka et al. [2013] as it was assumed that the tree S' is binary. In this paper however we will exploit the high degree of Steiner nodes in S' and hence prefer to mark the "Loss" edges.

We introduce the following notation: for an edge f in S^1 let $W(f) = \{e \in W | f \in p(e)\}$, we call $W(f)$ to be the *witness set* of f. Therefore, at iteration t, the reference solution S^t contains the edges from S^1 whose witness sets are not fully marked until iteration $t-1$.

Observe that the expected number of iterations an edge f from the reference solution survives (until being removed) $E[D(f)]$ can be expressed using only the size of its witness set $W(f)$.

Corollary 69. *Let $f \in S^1$, then $E[D(f)] \leq H_{|W(f)|} \cdot M$.*

Following the argument from the proof of Theorem 21 in Byrka et al. [2013], we also get

Corollary 70. *For $k = O_\varepsilon(1)$ large enough, the total cost of components bought by the algorithm is at most*

$$\frac{1+\varepsilon}{M} \sum_{f \in S^1} E[D(f)] \cdot c(f) \leq (1+\epsilon) \cdot \sum_{f \in S^1} H_{|W(f)|} \cdot c(f)$$

Therefore, it suffices to analyse how the marking scheme used in the construction of the witness tree affects distributions of the sizes of the witness sets for the individual edges of S^1. To this end we will exploit two properties of our instances: the high degree of the Steiner vertices in the optimal solutions, and the fact that all edges of S^1 have the same cost.

Appendix B

Omitted Proofs from Section 4.2

Claim 71. $H_{d_1} + \sum_{j=d_1+1}^{\infty} \frac{1}{j \cdot 2^{j-d_1}} = \hat{H}_{d_1}$.

Proof. Note that

$$\hat{H}_{d_1} = \sum_{i=0}^{\infty} \frac{H_{d_1+i}}{2^{i+1}} = \sum_{i=0}^{\infty} \frac{H_{d_1} + \sum_{j=1}^{i} \frac{1}{d_1+j}}{2^{i+1}} = H_{d_1} + \sum_{j=1}^{\infty} \frac{\sum_{i=j}^{\infty} \frac{1}{2^{i+1}}}{d_1+j} = H_{d_1} + \sum_{j=1}^{\infty} \frac{1}{(d_1+j)2^j}.$$

□

Proof of Lemma 60. The claim is trivially true if ℓ is the root since in that case $c(\ell) = H_{d(\ell)-1} < \hat{H}_{d(\ell)}$. So assume ℓ is not the root. For a generic sequence $S = (d_1, \ldots, d_k)$ of positive integers, let us define

$$f(S) = \sum_{j=1}^{k-1} \frac{(d_{j+1} - 1) \cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2 \cdot d_3 \ldots d_{j+1}} + \frac{H_{d_1+d_2+\cdots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k}.$$

Intuitively, this is the right-hand side of the equation in Lemma 59. For an infinite sequence $S' = (d_1, d_2, \ldots)$ of positive integers, we analogously define

$$f(S') = \sum_{j=1}^{\infty} \frac{(d_{j+1} - 1) \cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2 \cdot d_3 \ldots d_{j+1}}$$

Given a finite sequence $S = (d_1, \ldots, d_k)$ of the above type, let $\bar{S} = (d_1, \ldots, d_k, 2, 2, \ldots)$ be its infinite extension where we add an infinite sequence of 2 at the end.

Claim 72. $f(S) \leq f(\bar{S})$.

Proof. By definition

$$\begin{aligned} f(\bar{S}) - f(S) &= \sum_{j=k}^{\infty} \frac{(d_{j+1}-1)\cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2\cdot d_3\ldots d_{j+1}} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k} \\ &\geq \sum_{j=k}^{\infty} \frac{(d_{j+1}-1)\cdot H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_{j+1}} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k} \\ &= \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k}\sum_{j=1}^{\infty}\frac{1}{2^j} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k} \\ &= \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2\cdot d_3\ldots d_k} = 0. \end{aligned}$$

□

By the above claim it is sufficient to consider infine sequences of type $\bar{S}$. We can also assume w.l.o.g. that all d_i, $i \geq 2$, in such sequences are at least 2 by the following claim.

Claim 73. *Let $\bar{S} = (d_1, \ldots, d_k, 2, 2, \ldots)$ and assume there exists $d_i = 1$ in the sequence for some $i \geq 2$. Let $\bar{S}_i = (d_1, \ldots, d_i - 1, d_i + 1, \ldots, d_k, 2, 2, \ldots)$ be the subsequence where the i-th entry is removed. Then $f(\bar{S}) = f(\bar{S}_i)$.*

Proof. Consider the entries in the sum defining $f(\bar{S})$ and $f(\bar{S}_i)$. The entry $j = i-1$ in $f(\bar{S})$ has value 0. For $j < i-1$, the j-th entries in $f(\bar{S})$ and $f(\bar{S}_i)$ are identical. For $j > i-1$, the j-th entry in $f(\bar{S})$ is equal to the $j-1$-th entry in $f(\bar{S}_i)$. □

By the above claims we can focus on infinite sequences $S = (d_1, \ldots, d_k, 2, 2, \ldots)$ where $d_i \geq 2$ for $i \geq 2$. Let us prove by induction on $k \geq 2$ that $f(S) \leq \hat{H}_{d_1}$. The claim is true by definition for $k = 2$. Next consider any $k > 2$ and assume the claim is true for all values up to $k-1$. Define $S' = (d_1 + d_2 - 1, d_3, \ldots, d_k, 2, 2, \ldots)$. By definition and inductive hypothesis:

$$f(S) = H_{d_1}\frac{d_2-1}{d_2} + \frac{f(S')}{d_2} \leq H_{d_1}\frac{d_2-1}{d_2} + \frac{\hat{H}_{d_1+d_2-1}}{d_2}.$$

By Claim 71,

$$\begin{aligned} H_{d_1}\frac{d_2-1}{d_2} + \frac{\hat{H}_{d_1+d_2-1}}{d_2} &= H_{d_1}\frac{d_2-1}{d_2} + \frac{1}{d_2}\left(H_{d_1+d_2-1} + \sum_{j\geq d_1+d_2}\frac{1}{j\cdot 2^{j-d_1-d_2+1}}\right) \\ &= H_{d_1} + \sum_{j=d_1+1}^{d_1+d_2-1}\frac{1}{j\cdot d_2} + \sum_{j\geq d_1+d_2}\frac{1}{j\cdot d_2\cdot 2^{j-d_1-d_2+1}} \\ &= H_{d_1} + \sum_{j\geq d_1+1}\frac{\alpha_j}{j}, \end{aligned}$$

where

$$\alpha_j := \begin{cases} \frac{1}{d_2} & \text{for } d_1+1 \le j \le d_1+d_2-1; \\ \frac{1}{j \cdot 2^{i-d_1-d_2+1}} & \text{for } j \ge d_1+d_2. \end{cases}$$

We observe the following simple facts about the coefficients α_j.

Claim 74. *One has:*

1. $\sum_{j \ge d_1+1} \alpha_j = 1$.
2. *For every* $i > 1$, $\sum_{j \ge d_1+i} \alpha_j \ge \frac{1}{2^{i-1}}$.

Proof. 1. $\sum_{j \ge d_1+1} \alpha_j = \frac{d_2-1}{d_2} + \sum_{j=d_1+d_2}^{\infty} \frac{1}{d_2 \cdot 2^{j-d_1-d_2+1}} = 1 - \frac{1}{d_2} + \frac{1}{d_2}$.

2. For $i \ge d_2$, one has

$$\sum_{j \ge d_1+i} \alpha_j = \sum_{j=d_1+i}^{\infty} \frac{1}{d_2 \cdot 2^{j-d_1+d_2-1}} = \frac{1}{d_2 \cdot 2^{i-d_2}} \ge \frac{1}{2^{i-1}},$$

where in the inequality we used the fact that $k \le 2^{k-1}$ for any integer $k \ge 1$.

For $2 \le i \le d_2 - 1$, one has:

$$\sum_{j \ge d_1+i} \alpha_j = \frac{d_2-i}{d_2} + \frac{1}{d_2} = \frac{d_2-i+1}{d_2} \ge \frac{1}{i} \ge \frac{1}{2^{i-1}},$$

where in the first inequality above we used the fact that $\frac{k-j+1}{k}$ is a decreasing function of $k \ge j+1$ and $d_2 \ge i+1$, and in the second inequality again the fact that $k \le 2^{k-1}$ for $k \ge 1$.

□

Intuitively, the term $A = \sum_{j=d_1+1}^{\infty} \frac{\alpha_j}{j}$ is a convex combination of terms of type $1/j$ under the constraint that the sum of the tail coefficients is large enough. An obvious upper bound on A is obtained by choosing coefficients β_j that respect the constraints on α_j given by Claim 74, and at the same time are as large as possible on the smallest terms of the sum. An easy induction shows that the best choice is $\beta_j = \frac{1}{2^{j-d_1}}$ for all $j \ge d_1 + 1$. Thus we can conclude

$$f(S) \le H_{d_1} + \sum_{j \ge d_1+1} \frac{\alpha_j}{j} \le H_{d_1} + \sum_{j \ge d_1+1} \frac{\beta_j}{j} = H_{d_1} + \sum_{j=d_1+1}^{\infty} \frac{1}{j \cdot 2^{j-d_1}} = \hat{H}_{d_1},$$

where last equality comes from Claim 71. □

Proof of Claim 67. Consider $a_1(i)$. Excluding a fixed additive term $\hat{H}_2 - p$, the value of this function is $a'_1(i) := \frac{H_{i+2}-x}{i}$, where $x = \hat{H}_2 - p \in (0, \hat{H}_2]$. Taking the discrete derivative

$$a'_1(i+1) - a'_1(i) = \frac{x + \frac{i-1}{i+3} - H_{i+3}}{i(i+1)}$$

one might observe that this is negative for $i \geq 6$ since $x + \frac{i-1}{i+3} \leq \hat{H}_2 + 1 < 2.7726 < H_9 > 2.8289$. The reader might skip the following cases that are analogous.

Consider now $a_2(i)$. Excluding a fixed additive term $\hat{H}_2$, the value of this function is $a'_2(i) := \frac{H_i - x}{i}$, where $x = \hat{H}_2 - p \in (0, \hat{H}_2]$. One has

$$a'_2(i+1) - a'_2(i) = \frac{x + 1 - H_{i+1}}{i(i+1)},$$

which is negative for $i \geq 8$ since $x + 1 \leq \hat{H}_2 + 1 < 2.7726 < H_9 > 2.8289$.

Consider next $a_3(i)$. Excluding a fixed additive term $\hat{H}_2 - p$, the value of this function is $a'_3(i) := \frac{\hat{H}_{i+2} - \hat{H}_2}{i}$. One has

$$a'_3(i+1) - a'_3(i) = \frac{\hat{H}_2 - \hat{H}_{i+2}}{i(i+1)} + \sum_{j \geq 1} \frac{1}{2^j (i+1)(i+j+2)} \leq \frac{\hat{H}_2 + 1 - \hat{H}_{i+2}}{i(i+1)},$$

which is negative for $i \geq 6$ since $\hat{H}_2 + 1 < 2.7726 < \hat{H}_8 > 2.8194$.

It remains to consider $a_4(i)$. Excluding a fixed additive term $\hat{H}_2$, the value of this function is $a'_4(i) := \frac{H_i - \hat{H}_2}{i}$. One has

$$a'_4(i+1) - a'_4(i) = \frac{\hat{H}_2 - \hat{H}_i}{i(i+1)} + \sum_{j \geq 1} \frac{1}{2^j (i+1)(i+j)} \leq \frac{\hat{H}_2 + 1 - \hat{H}_i}{i(i+1)},$$

which is negative for $i \geq 8$ since $\hat{H}_2 + 1 < 2.7726 < \hat{H}_8 > 2.8194$. □

Bibliography

Adjiashvili, D. [2017]. Beating Approximation Factor Two for Weighted Tree Augmentation with Bounded Costs, *SODA 2017*, pp. 2384–2399.

Akbari, S., Ghodrati, A., Jabal Ameli, A. and Saghafian, M. [2017]. Chromatic number and dichromatic polynomial of digraphs.

Basavaraju, M., Fomin, F. V., Golovach, P., Misra, P., Ramanujan, M. and Saurabh, S. [2014]. Parameterized Algorithms to Preserve Connectivity, *ICALP 2014*, pp. 800–811.

Bondy, J. and Murty, U. [2008]. *Graph Theory*, 1st edn, Springer Publishing Company, Incorporated.

Byrka, J., Grandoni, F. and Jabal Ameli, A. [2020]. Breaching the 2-approximation barrier for connectivity augmentation: A reduction to steiner tree, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, Association for Computing Machinery, New York, NY, USA.

Byrka, J., Grandoni, F., Rothvoß, T. and Sanità, L. [2013]. Steiner tree approximation via iterative randomized rounding, *J. ACM* **60**(1): 6:1–6:33.

Cecchetto, F., Traub, V. and Zenklusen, R. [2021]. Bridging the gap between tree and connectivity augmentation: Unified and stronger approaches, *STOC 2021* .

Cheriyan, J., Cummings, R., Dippel, J. and Zhu, J. [2020]. An improved approximation algorithm for the matching augmentation problem.

Cheriyan, J., Dippel, J., Grandoni, F., Khan, A. and Narayan, V. V. [2020]. The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm, *Math. Program.* **182**(1): 315–354.

Cheriyan, J. and Gao, Z. [2018a]. Approximating (Unweighted) Tree Augmentation via Lift-and-Project, Part I: Stemless TAP, *Algorithmica* **80**: 530–559.

Cheriyan, J. and Gao, Z. [2018b]. Approximating (unweighted) tree augmentation via lift-and-project, part II, *Algorithmica* **80**(2): 608–651.

Cheriyan, J., Jordán, T. and Ravi, R. [1999]. On 2-coverings and 2-packings of laminar families, *Algorithms - ESA '99, 7th Annual European Symposium, Prague, Czech Republic, July 16-18, 1999, Proceedings*, pp. 510–520.

Cheriyan, J., Karloff, H., Khandekar, R. and Könemann, J. [2008]. On the Integrality Ratio for Tree Augmentation, *Oper. Res. Lett.* **36**: 399–401.

Cheriyan, J., Sebo, A. and Szigeti, Z. [2001]. Improving on the 1.5-approximation of a smallest 2-edge connected spanning subgraph, **14**(2).

Chlebík, M. and Chlebíková, J. [2008]. The steiner tree problem on graphs: Inapproximability results, *Theoretical Computer Science* **406**(3): 207–214. Algorithmic Aspects of Global Computing.

Cohen, N. and Nutov, Z. [2011]. A $(1 + ln2)$-Approximation Algorithm for Minimum-Cost 2-Edge-Connectivity Augmentation of Trees with Constant Radius, *APPROX/RANDOM 2011*, pp. 147–157.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. [2009]. *Introduction to Algorithms, Third Edition*, 3rd edn, The MIT Press.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M. and Saurabh, S. [2015]. *Parameterized Algorithms*, Springer Science & Business Media.

Dinitz, E., Karzanov, A. and Lomonosov, M. [1976]. On the Structure of the System of Minimum Edge Cuts of a Graph, *Studies in Discrete Optimization* pp. 290–306.

Even, G., Feldman, J., Kortsarz, G. and Nutov, Z. [2009]. A 1.8 Approximation Algorithm for Augmenting Edge-Connectivity of a Graph from 1 to 2, *ACM Trans. Algorithms* **5**: 21:1–21:17.

Fiorini, S., Groß, M., Könemann, J. and Sanità, L. [2018]. Approximating Weighted Tree Augmentation via Chvátal-Gomory Cuts, *SODA 2018*, pp. 817–831.

Frederickson, G. N. and JáJá, J. [1981]. Approximation algorithms for several graph augmentation problems, *SIAM Journal on Computing* **10**(2): 270–283.

Gabow, H. N., Goemans, M. X., Tardos, and Williamson, D. P. [2009]. Approximating the smallest k-edge connected spanning subgraph by lp-rounding, *Networks* **53**(4): 345–357.

Gálvez, W., Grandoni, F., Jabal Ameli, A., Jansen, K., Khan, A. and Rau, M. [2020]. A tight (3/2+ϵ) approximation for skewed strip packing, *in* J. Byrka and R. Meka (eds), *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, Vol. 176 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 44:1–44:18.

Gálvez, W., Grandoni, F., Jabal Ameli, A. and Sornat, K. [2021]. On the Cycle Augmentation Problem: Hardness and Approximation Algorithms, *Theory of Computing Systems* **21**(1): 39–60.

Goemans, M. X., Goldberg, A. V., Plotkin, S., Shmoys, D. B., Tardos, E. and Williamson, D. P. [1994]. Improved Approximation Algorithms for Network Design Problems, *SODA 1994*, pp. 223–232.

Grandoni, F., Kalaitzis, C. and Zenklusen, R. [2018]. Improved approximation for tree augmentation: saving by rewiring, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pp. 632–645.

Hunkenschröder, C., Vempala, S. and Vetta, A. [2019]. A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem, **15**(4).

Jain, K. [2001]. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem, *Combinatorica* **21**(1): 39–60.

Khuller, S. and Thurimella, R. [1993]. Approximation Algorithms for Graph Augmentation, *J. Algorithms* **14**: 214–225.

Khuller, S. and Vishkin, U. [1994]. Biconnectivity approximations and graph carvings, *Journal of the ACM (JACM)* **41**(2): 214–235.

Knuth, D. E. [1974]. Postscript About NP-hard Problems, *SIGACT News* **6**(2): 15–16.

Kortsarz, G., Krauthgamer, R. and Lee, J. R. [2004]. Hardness of Approximation for Vertex-Connectivity Network Design Problems, *SIAM J. Comput.* **33**(3): 704–720.

Kortsarz, G. and Nutov, Z. [2016a]. LP-Relaxations for Tree Augmentation, *APPROX/RANDOM 2016*, pp. 13:1–13:16.

Kortsarz, G. and Nutov, Z. [2016b]. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2, *ACM Transactions on Algorithms (TALG)* **12**(2): 23:1–23:20.

Marx, D. and Végh, L. A. [2015]. Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation, *ACM Transactions on Algorithms (TALG)* **11**(4): 27.

Nagamochi, H. [2003]. An approximation for finding a smallest 2-edge-connected subgraph containing a specified spanning tree, *Discrete Applied Mathematics* **126**(1): 83–113.

Nutov, Z. [2017]. On the tree augmentation problem, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pp. 61:1–61:14.

Nutov, Z. [2020]. 2-node-connectivity network design, *WAOA 2020* .

Petrank, E. [1994]. The Hardness of Approximation: Gap Location, *Computational Complexity* **4**: 133–157.

Robins, G. and Zelikovsky, A. [2005]. Tighter bounds for graph Steiner tree approximation, *SIAM Journal on Discrete Mathematics* **19**(1): 122–134.

Sebö, A. and Vygen, J. [2014]. Shorter tours by nicer ears: 7/5-approximation for graphic tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs, *Combinatorica* **34(5)**: 597–629.

Traub, V. and Zenklusen, R. [2021]. A better-than-2 approximation for weighted tree augmentation.

Watanabe, T. and Nakamura, A. [1987]. Edge-connectivity augmentation problems, *Journal of Computer and System Sciences* **35**(1): 96–144.