FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Improving current approaches to Min-Power Symmetric Connectivity
Student:	Richard Hartmann
Supervisor:	doc. RNDr. Dušan Knop, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	until the end of summer semester 2024/2025

Instructions

The task is to study the Min-Power Symmetric Connectivity problem as introduced, e.g., by Bentert et al. [1]. This problem is NP-hard and has already been studied from various computational complexity perspectives such as, e.g., parameterized complexity. It is quite common that some published results are not fully optimized as there is a clear fight with the well-known Pareto principle. In this thesis, the task is to investigate possibilities to optimize some of the routines used in the known algorithms for the Min-Power Symmetric Connectivity problem.

Tasks:
- Study [1] in full detail and present an overview of known results
- Describe the approach taken in [1] in detail
- Select some lemma or aspect studied in [1] a try to improve the guarantees

References:
[1] Matthias Bentert, René van Bevern, André Nichterlein, Rolf Niedermeier, Pavel V. Smirnov: Parameterized Algorithms for Power-Efficiently Connecting Wireless Sensor Networks: Theory and Experiments. INFORMS J. Comput. 34(1): 55-75 (2022)

Electronically approved by doc. Ing. Jan Janoušek, Ph.D. on 8 February 2024 in Prague.

Bachelor's thesis

IMPROVING CURRENT APPROACHES TO MIN-POWER SYMMETRIC CONNECTIVITY

Richard Hartmann

Faculty of Information Technology
Department of Theoretical Computer Science
Supervisor: doc. RNDr. Dušan Knop, Ph.D.
May 8, 2024

Contents

List of Figures

List of Tables

I would like to express my sincere gratitude to my supervisor, doc. RNDr. Dušan Knop, Ph.D., for his guidance and valuable advice. I would also like to thank my family and friends for their support during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Czech Technical University in Prague has the right to conclude a licence agreement on the utilization of this thesis as a school work pursuant of Section 60 (1) of the Act.

In Prague on May 8, 2024

Abstract

In this work, we deal with the problem of minimising the total power consumption of a wireless network, which is formally known as Min-Power Symmetric Connectivity. The wireless network is represented by an edge-weighted undirected graph. The goal is to find a connected spanning subgraph with the minimum sum of the costs across all vertices. The cost of a vertex is given by the largest weight of the edge incident to the given vertex. It is known that the problem is NP-hard. We extend this result and show that the problem is NP-hard for a particular class of complete graphs. Next, we present two routines for solving Min-Power Symmetric Connectivity more efficiently. One of the methods focuses on finding the so-called obligatory subgraph. We are able to find it efficiently using lower bounds on the vertex cost. We introduce a method for incrementing these vertex lower bounds. Subsequently, we focus on the possible reduction in the number of edges. We present a method for identifying redundant edges in a graph, thereby reducing the computational complexity of the task.

Keywords wireless networks, connectivity, vertex lower bounds, obligatory subgraph, edge reduction, NP-hard

Abstrakt

V této práci se zabýváme problémem minimalizace celkové spotřeby energie bezdrátové sítě, který je formálně známý jako Min-Power Symmetric Connectivity. Modelem bezdrátové sítě je hranově ohodnocený neorientovaný graf. Cílem je najít souvislý podgraf, který pokrývá všechny vrcholy, s minimálním součtem cen všech vrcholů. Cena vrcholu je dána největší váhou hrany incidentní s daným vrcholem. Je známo, že se jedná o NP-těžký problém. Toto tvrzení rozšíříme a ukážeme, že problém je NP-těžký pro konkrétní třídu úplných grafů. Dále představíme dva postupy umožňující efektivnější řešení Min-Power Symmetric Connectivity problému. Jedna z metod se zaměřuje na hledání tzv. povinného podgrafu. Ten jsme schopni efektivně nalézt za pomoci dolních mezí cen vrcholů. Představíme metodu, jak tyto dolní meze cen vrcholů zvýšit. Dále se zabýváme možnou redukcí počtu hran. Ukážeme metodu identifikující nadbytečné hrany grafu, čímž snížíme výpočetní složitost úlohy.

Klíčová slova bezdrátové sítě, konektivita, dolní mez cen vrcholů, povinný podgraf, redukce hran, NP-těžký

Introduction

Wireless sensor networks receive considerable attention due to their broad range of applications. Sensors capture information from its surrounding environment and communicate with other sensors, forming a wireless network.

The wireless sensor network should be connected. That is, every two nodes of the network must be able to communicate with each other, either directly or via other nodes. We consider a symmetric wireless network, i.e., the sensors communicate bidirectionally. A bidirectional connection is established between two nodes if their transmission power exceeds a threshold determined by various factors, such as the environment in which the nodes communicate.

Typically, each sensor is furnished with limited power source, therefore it is important to minimize the total energy consumption of the network.

The problem is known as Min-Power Symmetric Connectivity. It was first studied by Calinescu et al. in 2002 [1]. Since then researchers studied the problem from various perspectives and on various graph classes. This thesis primarily relies on the work of Bentert et al. [2].

Eventhough the problem has been already well studied, Bentert et al. introduced new findings from a computational complexity perspective, namely parameterized complexity. They proposed a new algorithm for solving Min-Power Symmetric Connectivity.

The goal of this thesis is to further fine-tune the approach taken by Bentert et al. [2].

We acquaint the reader with the problem in more detail and describe the approach taken in [2]. We present new results on the computational complexity of Min-Power Symmetric Connectivity, increase lower bounds of the transmission power of specific nodes, and introduce new preprocessing procedure.

The structure of the thesis is as follows:

Chapter 1 We build a theoretical background, introduce notation, and define concepts from graph theory and computational complexity. We describe the Min-Power Symmetric Connectivity problem with important concepts that are used later in the thesis.

Chapter 2 We study the problem from a computational complexity perspective. We introduce new results concerning NP-hardness of specific instances of Min-Power Symmetric Connectivity, and summarize the results given in [2].

Chapter 3 We propose two optimization routines: incrementation of lower bounds of the transmission power of specific nodes and a new preprocessing procedure regarding the reduction of edges.

Chapter 1
Problem Description

In this chapter, we introduce Min-Power Symmetric Connectivity as an optimization problem in graph theory. First we define concepts from graph theory in Section 1.1 and computational complexity in Section 1.2 that are used in this work. Then we define Min-Power Symmetric Connectivity in Section 1.3. Finally, we focus on vertex lower bounds and the so-called obligatory subgraph in Section 1.4 and 1.5, which are the fundamental concepts on which we build our optimization routines.

1.1 Graph Theory

Here, we introduce basic notions from graph theory. We refer the interested reader to Nešetřil & Matoušek [3] or Diestel [4].

▶ **Definition 1.1** (Undirected graph)**.** Undirected graph G *is a pair* (V, E)*, where* V *is a non empty finite set of* vertices *and* $E \subseteq \{\{u, v\} \mid u \neq v \text{ and } u, v \in V\}$ *is a set of* edges*. By* $V(G)$ *we denote set of vertices* V *of graph* G *and by* $E(G)$ *we denote set of edges* E*.*

Further on, we will refer to the undirected graph simply as a graph.

▶ **Definition 1.2** (Walk)**.** *A* walk *in a graph* $G = (V, E)$ *is a sequence* $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$*, where* $e_i = \{v_{i-1}, v_i\}$ *and* $e_i \in E$ *for all* $i = 1, \ldots, k$*.*

▶ **Definition 1.3** (Path)**.** *A walk in* G *in which all vertices are distinct is called* path *in graph* G*.*

If the end vertices of the path are u and v, we call it a *u-v-path*. The case of equality of u and v is admitted, in which the path length is 0, because it cannot contain other vertices.

▶ **Definition 1.4** (Connected graph)**.** *A graph* $G = (V, E)$ *is* connected *if its vertices are pairwise connected. That is, there exist a* u*-*v*-path in* G *for all* $u, v \in V$*.*

▶ **Definition 1.5** (Subgraph)**.** *A graph* H *is a* subgraph of graph G *if* $V(H) \subseteq V(G)$ *and* $E(H) \subseteq E(G)$*.*

▶ **Definition 1.6** (Spanning subgraph)**.** *A graph* H *is a* spanning subgraph of graph G *if* H *is a subgraph of* G *and* $V(H) = V(G)$*.*

▶ **Definition 1.7** (Spanning tree)**.** *A graph* H *is a* spanning tree of graph G *if* H *is a spanning subgraph of* G *in which any two vertices are connected by exactly one path.*

▶ **Definition 1.8** (Connected component)**.** *Let* G *be a graph. A subgraph* S *of* G *is a* connected component *of* G *if* S *is connected and if* S *is not part of any larger connected subgraph of* G*.*

▶ **Definition 1.9** (Edge-weighted graph)**.** *An* edge-weighted graph (G, w) *consists of a graph* $G = (V, E)$ *and a weight function* $w\colon E \to \mathbb{N}$.

▶ **Definition 1.10** (Clique)**.** *Let* $G = (V, E)$ *be an undirected graph. Clique* C *is a subgraph of* G*, such that there exist an edge* $\{u, v\} \in E(C)$ *for every pair of distinct vertices* $u, v \in V(C)$.

In the rest of this thesis we use the following notation. By $G - u$ we refer to the graph that differs from G by removing vertex u and all its incident edges. By $G - \{u, v\}$ we refer to the graph G that does not include edge $\{u, v\}$ and by $G + \{u, v\}$ we refer to the graph G with an additional edge $\{u, v\}$. Furthermore, we extend this notation to a subset of vertices $U \subseteq V(G)$ of graph G and to a subset of edges $F \subseteq E(G)$ of graph G. That is, by $G - U$ we denote the graph G that does not include vertices from U and all their incident edges in G. By $G - F$ we denote the graph G without edges from F.

1.2 Computational Complexity

In this section, we briefly describe specific concepts of computational complexity that we use throughout this thesis. We refer the interested reader to Arora & Barak [5] and Scott Aaronson's Complexity Zoo [6].

First, we focus on complexity classes P and NP. A complexity class is a set of functions that meets some specific requirements. Concretely:

- *Class P* is the class of decision problems solvable by a deterministic Turing machine in polynomial time with respect to the size of input x.
- *Class NP* is the class of decision problems for which given an input x and a polynomial-size verifier for x we can verify if the answer is YES in polynomial time by a deterministic Turing machine. If the answer is NO, then the algorithm must declare invalid any purpoted polynomial-size solution for x in polynomial time by a deterministic Turing machine.

Suppose we have two decision problems A and B. If there exists a polynomial-time computable function f, such that for every input x to problem A, $f(x)$ is an input to problem B and it holds that $A(x)$ is a YES instance if and only if $B(f(x))$ is a YES instance, then we say that A is *polynomial-time Karp reducible* to B. It means, that decision problem B is at least as hard as problem A.

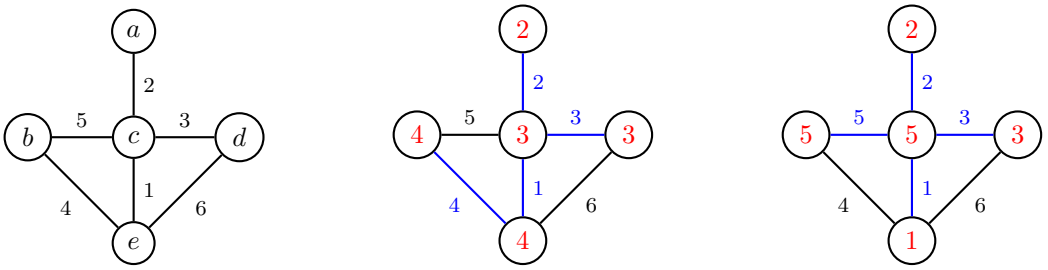
If every decision problem in NP is polynomial-time Karp reducible to a decision problem B, then we say B is *NP-hard*. In practice, to show that a decision problem B is NP-hard, we do not have to reduce every problem in NP to B. It suffices to reduce only one known NP-hard problem A to B because polynomial-time Karp reduction is transitive.

Notice, that to this point we have only studied complexity of problems based on the size of an input x. Parameterized complexity studies the scenario, when we are given some parameter $k \in \mathbb{N}$ alongside with an input x. Such parameter k may help to solve the problem more efficiently. This brings us to *fixed-parameter tractable (FPT)* problems.

- FPT is the class of decision problems of the form (x, k) where x is the input and $k \in \mathbb{N}$ is the parameter, that are solvable in $f(k) \cdot |x|^{O(1)}$ time for some computable function f. An algorithm that solves a decision problem in such time is called a *fixed-parameter* algorithm.

The idea is that even if we do not know how to solve some problems efficiently given the size of the input x, the slow running time may be most affected by the parameter k and not by the size of the input x. Therefore, even if the size of x is large, we are still able to solve the problem in reasonable time given relatively small parameter k.

Another parameterized complexity class is *XP*.



(a) An instance of MinPSC. **(b)** A solution to MinPSC. **(c)** A solution to MinPSC.

■ **Figure 1.1** An example of a MinPSC instance I is visualized in (a). Figures (b) and (c) display two possible solutions of I. The solutions are marked in blue and the costs of individual vertices are marked in red. The costs of the solutions are 16.

- XP is the class of decision problems of the form (x, k) where x is the input and $k \in \mathbb{N}$ is the parameter, that are solvable in $f(k) \cdot |x|^{g(k)}$ time for some computable functions f and g.

It is not difficult to realize that FPT $\subseteq$ XP, but it is an open question whether FPT = XP. If *exponential time hypothesis* holds, then we known that the inclusion is proper, i.e., FPT $\neq$ XP.

- *Exponential time hypothesis (ETH)* states that the satisfiablitity of 3-conjuctive normal form (3-CNF) boolean formulas with n variables and m clauses cannot be solved in $2^{o(n+m)}$ time [7].

For more information about parameterized complexity we refer the reader to the book Parameterized algorithms by Cygan et al. [8].

1.3 Min-Power Symmetric Connectivity

A wireless sensor network can be represented through an edge-weighted undirected graph. Vertices of the graph represent individual nodes of the network and edges represent potential communication links. The weight of an edge represents the minimum transmission power required for the nodes to establish a communication link.

The Min-Power Symmetric Connectivity problem was first introduced by Calinescu et al. in 2002 [1]. We define the problem in accordance with Bentert et al. [2].

▶ **Definition 1.11** (Min-Power Symmetric Connectivity(MinPSC)). *Given a connected, undirected graph $G = (V, E)$ with edge weights $w\colon E \to \mathbb{N}$, find a connected spanning subgraph $S = (V, F)$ of G that minimizes*

$$\sum_{u \in V} \max_{\{u,v\} \in F} w(\{u, v\}). \tag{1.1}$$

Figure 1.1 illustrates a basic example of MinPSC.

Typically, the goal is to find a spanning tree (see, e.g., [1, 9]), but notice that we seek a connected spanning subgraph. It is not a significant matter as these concepts are closely related when it comes to MinPSC. Any spanning tree is also a connected spanning subgraph. If a connected spanning subgraph is not a spanning tree, it can be transformed into a tree by removing some of its edges. Deletion of any edge cannot increase the cost function (1.1). However, we find it convenient to use a connected spanning subgraph due to an obligatory subgraph, a concept that we will define later in this chapter.

The cost of a solution to MinPSC is the sum of the costs of individual vertices.

▶ **Definition 1.12** (Vertex cost)**.** Vertex cost *is an attribute of a vertex of an edge-weighted graph* $G = (V, E)$ *given by function* $\mathrm{c_G} \colon V \to \mathbb{N}$

$$\mathrm{c_G}(u) = \begin{cases} \max_{\{u,v\} \in E} w(\{u, v\}), & \text{if there exists } v \in V \text{ such that } \{u, v\} \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{1.2}$$

By $\mathrm{C_G}$ we denote the sum of costs of individual vertices of G. That is

$$\mathrm{C_G} = \sum_{u \in V(G)} \mathrm{c_G}(u).$$

We refer to $\mathrm{C_G}$ for short as *cost of* G. Notice that we can rewrite (1.1) as $\mathrm{C_S}$.

Bentert et al. [2] gave an algorithm for MinPSC based on so-called vertex lower bounds and obligatory subgraphs.

1.4 Vertex Lower Bounds

In this section, we discuss lower bounds on costs of individual vertices. We adopted the definition from Bentert et al. [2].

▶ **Definition 1.13** (Vertex lower bounds)**.** *Let* (G, w) *be an instance of MinPSC. A mapping* $\ell \colon V(G) \to \mathbb{N}$ *is a* vertex lower bound *if there exists a solution* S *for* (G, w) *such that*

$$\forall u \in V \colon \mathrm{c_S}(u) \geq \ell(u). \tag{1.3}$$

Bentert et al. build their results on following vertex lower bounds.

▶ Observation 1.14. Let $\ell \colon V(G) \to \mathbb{N}$ be a mapping given by

$$\ell(u) = \min_{\{u,v\} \in E(G)} w(\{u, v\}),\ u \in V(G), \tag{1.4}$$

then ℓ is a vertex lower bound.

Proof. Every vertex u is incident with some edge in any solution to MinPSC as it is a connected spanning subgraph of G by definition. ◀

More sophisticated vertex lower bounds utilize the concept of connected components.

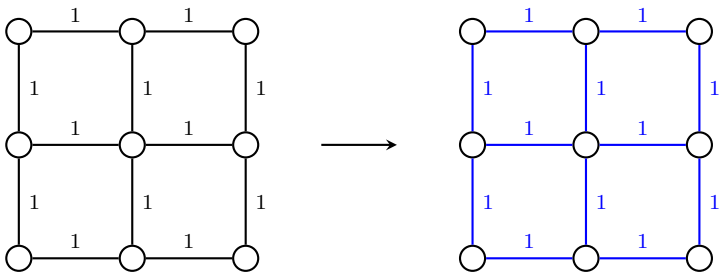
▶ Observation 1.15. Let $\ell \colon V(G) \to \mathbb{N}$ be a mapping given by

$$\ell(u) = \max_{G' \in \mathcal{C}} \min_{\{u,v\} \in G'} w(\{u, v\}), \tag{1.5}$$

where $\mathcal{C}$ is the set of connected components of graph $G - u$. Then ℓ is a vertex lower bound.

Proof. Any solution to MinPSC is a connected spanning subgraph of G by definition. Therefore, vertex u must be incident with at least one vertex from every connected component of $G - u$ in any solution. ◀

Note that Definition 1.13 of vertex lower bounds ensures the existence of at least one solution to MinPSC for which the vertex lower bounds hold, but the vertex lower bounds from Observation 1.14 and Observation 1.15 are more universal, since they hold for every solution to MinPSC.



■ **Figure 1.2** A perfect scenario where even simple vertex lower bounds yield an obligatory subgraph with only one connected component. The left side represents the original graph, while the right side represents an obligatory subgraph. Its obligatory edges are marked in blue.

1.5 Obligatory Subgraph

One of the reasons to study vertex lower bounds is that it allows us to find some parts of a solution to MinPSC effectively. These parts are called obligatory subgraphs.

We define obligatory subgraphs in accordance with Bentert et al. [2].

▶ **Definition 1.16** (Obligatory edge)**.** *Let (G, w) be an instance of MinPSC and ℓ its vertex lower bounds. An edge $e = \{u, v\}$ is obligatory with respect to ℓ if*

$$\min(\ell(u), \ell(v)) \geq w(e). \tag{1.6}$$

▶ **Definition 1.17** (Obligatory subgraph)**.** *Let (G, w) be an instance of MinPSC and ℓ its vertex lower bounds. The graph G_ℓ is an obligatory subgraph with respect to ℓ if $V(G) = V(G_\ell)$ and $E(G_\ell)$ consists of all obligatory edges with respect to ℓ.*

▶ Observation 1.18. For every obligatory subgraph G_ℓ there exist at least one solution to MinPSC that includes G_ℓ.

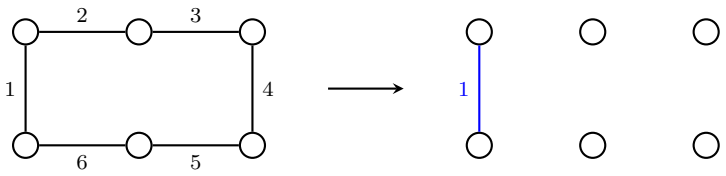
Proof. For vertex lower bounds ℓ there exists a solution S to MinPSC in which the cost of vertex $u \in V(S)$ is at least $\ell(u)$ (it directly follows from Definition 1.13). For every edge $\{u, v\} \in G_\ell$ it holds that $c_S(u) \geq \ell(u) \geq w(\{u, v\})$ and $c_S(v) \geq \ell(v) \geq w(\{u, v\})$. If we add $\{u, v\}$ to S, it does not increase the cost of vertices u and v, hence it does not increase the cost C_S. ◀

Important parameter of an obligatory subgraph is its number of connected components c. Notice that in a perfect scenario, when $c = 1$, the obligatory subgraph G_ℓ is already a solution to MinPSC, because G_ℓ is a connected spanning subgraph of G and the cost (1.1) is minimal as G_ℓ is a subgraph of some solution.

▶ **Example 1.19.** A wireless sensor network arranged in a grid pattern is an efficient way to minimize communication energy usage per unit area [10]. An example of such network is illustrated in Figure 1.2. When we use vertex lower bounds from Observation 1.14 or Observation 1.5, all of the edges are obligatory and the graph becomes an obligatory subgraph itself. The graph itself is a solution to MinPSC.

▶ **Example 1.20.** Vertex lower bounds from Observation 1.14 or Observation 1.5 may induce only one obligatory edge. For instance, it occurs in the case of graph with n vertices and pairwise disctinct edge-weights, resulting in an obligatory subgraph with $n - 1$ connected components. Figure 1.3 illustrates such a scenario.

After we find an obligatory subgraph, it remains to connect its connected components to get a connected spanning subgraph minimizing (1.1) (i.e., solve MinPSC). We could use a naive approach as follows.



■ **Figure 1.3** A worst case scenario where obligatory subgraph contains only one obligatory edge. The left side represents the original graph with n vertices, while the right side represents an obligatory subgraph with $n-1$ connected components. Its obligatory edge is marked in blue.

To make G_ℓ connected, we need to add $c-1$ edges, where c is a number of connected components of G_ℓ. A graph with n vertices has at most $\frac{n(n-1)}{2}$ edges. One has $\binom{\frac{n(n-1)}{2}}{c-1} \in O(n^{2c-2})$ possibilities of how to choose the edges. For each of these selections, we need to check that it yields a connected graph. With the deapth-first search (DFS) algorithm it takes $O(n+m)$ steps where m denotes the number of edges. Because $m \leq \frac{n(n-1)}{2}$ we say that DFS takes $O(n^2)$ steps.

This gives us a brute-force algorithm that finds an optimal solution to MinPSC of (G, w), given an obligatory subgraph G_ℓ, in $O(n^{2c})$ steps. Notice that it classifies MinPSC with parameter c as XP. Bentert et al. [2] present a more efficient algorithm, which we briefly describe in Section 2.2.1.

Now we understand why increasing the vertex lower bounds is an imporant challenge. The larger vertex lower bounds the more obligatory edges, which may induce an obligatory subgraph with fewer connected components (the number of connected components may remain the same but cannot be increased). The fewer connected components an obligatory subgraph has, the fewer number of connected components we have to then reconnect. We cover this topic of increasing vertex lower bounds in Section 3.1.

▶ Note. Vertex lower bounds are not the only way we obtain obligatory subgraphs. In real-life scenarious, a wireless sensor network may include faulty sensors, causing the network to lose connectivity and become separated into different components that need to reconnect at a minimum additional cost.

Chapter 2

Computational Complexity Analysis

In this chapter, we study MinPSC from the perspective of computational complexity. Kirousis et al. [11] proved that MinPSC (its decision version) is NP-hard. We extend the results about NP-hardness concerning complete graphs and summarize the results of Bentert et al. [2] about the parametrized complexity of MinPSC.

2.1 Complete Graphs

To study computational complexity of MinPSC, we define its decision version.

▶ **Definition 2.1** (k-MinPSC)**.** *Let $G = (V, E)$ be a connected undirected graph with edge weights $w\colon E \to \mathbb{N}$ and let $k \in \mathbb{N}$. The problem is to decide whether there exists a solution to MinPSC of (G, w) with cost C for which $C \leq k$.*

Although k-MinPSC is NP-hard, there are specific instances of MinPSC that can be solved in polynomial time.

▶ **Proposition 2.2.** *MinPSC is polynomially solvable on instances of complete graphs with only two distinct edge weigths.*

Erzin et al. [12] presented a simple proof.

Proof. Let (G, w) be an instance of MinPSC. Let G be a complete graph with $w(e) \in \{a, b\}$, $\forall e \in E(G)$. Assume that $a < b$.

Construct a graph $S = (V, \emptyset)$. Each edge has to pay at least a in any solution. We add these edges of weight a to S. Let m be a number of connected components of S. We need to add $m-1$ edges of weight b to S to make it connected. Choose one vertex in every connected component and connect the components via these vertices.

S is an optimal solution to MinPSC of (G, w), because number of vertices incident with an edge of weight b is minimized. ◀

When considering complete graphs with three disctinct edge-weights, the situation changes dramatically. We prove that k-MinPSC is NP-hard for complete graphs with three distinct edge-weights. To prove this statement, we modify a reduction of minimum set covering problem to MinPSC presented by Erzin et al. [12] with aspects inspired by Bentert et al. [2].

▶ **Definition 2.3** (Minimum Set Cover (MinSC)). *Let U be a universe and let $\mathscr{F}$ be a family of subsets of U satisfying $\bigcup_{S\in\mathscr{F}} S = U$. A* set cover *$\mathscr{F}' \subseteq \mathscr{F}$ satisfies $\bigcup_{S\in\mathscr{F}'} S = U$.* Minimum Set Cover *is a set cover $\mathscr{F}'$ of minimum size.*

▶ **Definition 2.4** (d-MinSC). *Let U be a universe, let $\mathscr{F}$ be a family of subsets of U satisfying $\bigcup_{S\in\mathscr{F}} S = U$ and let $d \in \mathbb{R}$. The problem is to decide whether there exists a set cover $\mathscr{F}'$ of $(U, \mathscr{F})$ with $|\mathscr{F}'| \leq d$.*

▶ **Theorem 2.5.** *k-MinPSC is NP-hard for complete graphs with three distinct edge-weights $a_1, a_2, a_3 \in \mathbb{N}$ with $a_1 < a_2 < a_3$ and $a_3 \geq 2a_2 - a_1$.*

To simplify the proof of the theorem, we define the notion of additional cost.

▶ **Definition 2.6** (Additional cost). *Let G be a graph with edge weights $w\colon E \to \mathbb{N}$ and let $\{u, v\} \in E(G)$ be its edge.* Additional cost *of edge $\{u, v\}$ in G is defined as*

$$\mathrm{add}_G(\{u, v\}) = \mathrm{C}_G - \mathrm{C}_{G-\{u,v\}}\,. \tag{2.1}$$

Let $A \subseteq E(G)$ be a set of edges of G. Additional cost *of A in G is defined as*

$$\mathrm{add}_G(A) = \mathrm{C}_G - \mathrm{C}_{G-A}\,. \tag{2.2}$$

The idea of additional cost is that it indicates the amount of cost introduced by an edge (or set of edges) to a graph.

Proof. (Theorem 2.5) We prove the theorem using a polynomial-time Karp reduction of d-MinSC into a k-MinPSC. Consider an instance $(U, \mathscr{F}, d)$ of d-MinSC and an instance (G, w, k) of k-MinPSC. The proof is divided into following parts:

1. We show how to construct graph G with edge-weights w and how to determine a decision boundary k (forming a k-MinPSC instance (G, w, k)) in polynomial time.
2. We show how a solution to (G, w, k) corresponds to a solution to $(U, \mathscr{F}, d)$.

(Step 1) Construct a graph G with vertices $V(G) = \{s\} \dot\cup \mathscr{F} \dot\cup U$, where s is a new vertex. By $\dot\cup$ we denote the disjoint union of sets. Connect each pair of vertices $V(G)$ by an edge to transform it to a complete graph and set the edge-weights as follows:

1. $\forall F \in \mathscr{F}\colon w(\{s, F\}) = a_1$
2. $\forall F \in \mathscr{F}, \forall u \in U$ such that $u \in F\colon w(\{u, F\}) = a_2$
3. Set $w(\{u, v\}) = a_3$ for all remaining edges $\{u, v\}$.

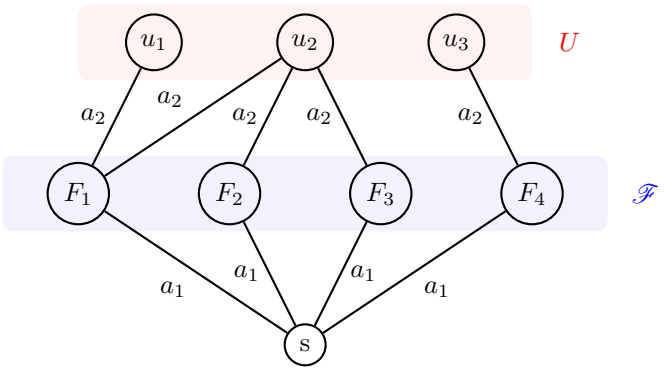
Finally, set the budget to $k = a_1 + |U|a_2 + |\mathscr{F}|a_1 + d(a_2 - a_1)$. It is straightforward to verify that the entire construction of a k-MinPSC instance (G, w, k) is done in polynomial time.

See an example of constructing (G, w, k) in the following Figure 2.1.

(Step 2) We show how a solution to (G, w, k) corresponds to a solution to $(U, \mathscr{F}, d)$.

First we prove that we can transform every connected spanning subgraph S of G that contains at least one edge of weight a_3 into another connected spanning subgraph S' of G that contains only edges of weight a_1 and a_2 but no edge of weight a_3 for which it holds that $\mathrm{C}_S \geq \mathrm{C}_{S'}$.

Without loss of generality, assume that there is an edge $\{s, F\} \in E(S)$ for every $F \in \mathscr{F}$ with edge-weight a_1. Notice, that it is a valid assumption, because we can add an edge of weight a_1 to any connected spanning subgraph without increasing its cost (remember that a_1 is the smallest edge-weight in S).



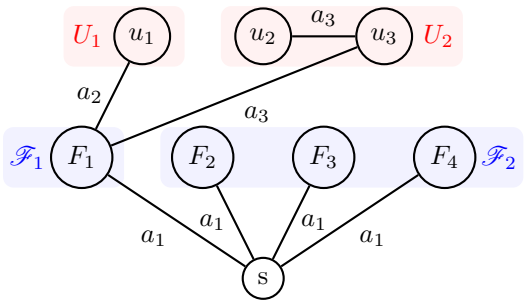
■ **Figure 2.1** An example of an instance (G, w, k) constructed from $(U, \mathscr{F}, d)$, where $U = \{u_1, u_2, u_3\}$, $\mathscr{F} = \{\{u_1, u_2\}, \{u_2\}, \{u_2\}, \{u_3\}\}$ and $d = 2$. For the sake of clarity, we have excluded all edges with weight a_3 in the visualization. We set the decision boundary k to $a_1 + 3a_2 + 4a_1 + 2(a_2 - a_1) = 3a_1 + 5a_2$.

Let $\mathscr{F}_1 = \{F \mid F \in \mathscr{F}, u \in U, u \in F, \{u, F\} \in E(S)\}$, i.e., all vertices from $\mathscr{F}$ in S that are connected to an edge with weight a_2. Let $U_1 = \{u \mid u \in U, F \in \mathscr{F}, u \in F, \{u, F\} \in E(S)\}$, i.e., all vertices from U in S that are connected to an edge with weight a_2. Let $\mathscr{F}_2 = \mathscr{F} \setminus \mathscr{F}_1$ and let $U_2 = U \setminus U_1$.

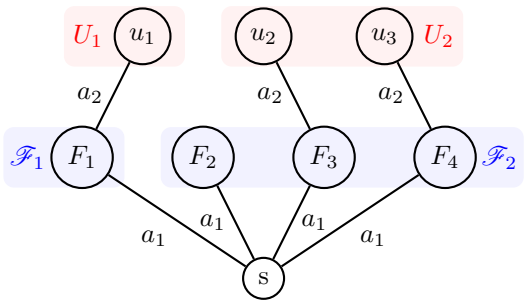
We construct S' the following way:

1. The subgraph S' contains all edges $\{s, F\}, F \in \mathscr{F}$ of weight a_1.
2. The subgraph S' contains all edges $\{u, F\}, u \in U, F \in \mathscr{F}, u \in F, \{u, F\} \in E(S)$. That is, all edges in S with weights a_2. Note that with this step we have connected all vertices from U_1 and $\mathscr{F}_1$.
3. It remains to connect the rest of vertices that are not yet incident with any edge in S', that is all vertices from U_2. Notice that vertices $u \in U_2$ are connected in S using edges with weights a_3. Connect every $u \in U_2$ to exactly one vertex $F \in \mathscr{F}$ for which $u \in F$. Note that such vertex F always exists(it follows directly from the definition of MinSC). Set the weight of those edges to a_2.

See the following visualization of such transformation of S to S'.



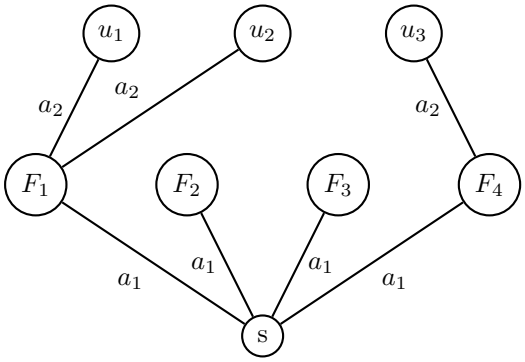
(a) A connected spanning subgraph S of graph G that includes two edges with weight a_3.



(b) A connected spanning subgraph S' of G without any edge-weight a_3.

■ **Figure 2.2** An example of transformation of a connected spanning subgraph S of the graph G from Figure 2.1 to a connected spanning subgraph S' without any edge-weight a_3.

Notice that the graph S' is in fact a connected spanning subgraph of G. This follows directly from the construction of S'.



■ **Figure 2.3** The solution T to MinPSC of (G, w) from Figure 2.1. The cost of the solution is $\mathrm{C}_T = a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1) = a_1 + 3a_2 + 4a_1 + 2(a_2 - a_1)$.

It remains to prove that the cost $\mathrm{C}_{S'}$ of S' is not greater than the cost C_S of S. Step 1 and 2 only introduced edges that are already in S. Step 3 established edges connecting vertices from U_2, let us denote the edges by A. It holds that

$$\mathrm{add}_{S'}(A) \leq |U_2|(2a_2 - a_1), \tag{2.3}$$

that is, the edges from A introduced a maximum of $|U_2|(2a_2 - a_1)$ additional cost to S'.

Let us denote by $B \subseteq E(S)$ all edges from S with edge-weight a_3. We analyze the additional cost of edges B in S. It holds that

$$\mathrm{add}_S(B) \geq |U_2|a_3 + a_3 - a_2. \tag{2.4}$$

If $\mathrm{add}_{S'}(A) \leq \mathrm{add}_S(B)$ then $C_{S'} \leq C_S$. It follows from the definition of additional cost add and the fact that $\mathrm{C}_{S-B} = \mathrm{C}_{S'-A}$ because $S - B = S' - A$.

Let us compare the right-hand sides of inequalities (2.3) and (2.4). It holds that

$$|U_2|a_3 + a_3 - a_2 \geq |U_2|(2a_2 - a_1) + a_3 - a_2 \geq |U_2|(2a_2 - a_1) \tag{2.5}$$

because $a_3 \geq 2a_2 - a_1$ and $a_3 > a_2$. Inequality (2.5), when combined with inequalities (2.3) and (2.4), results in $\mathrm{add}_{S'}(A) \leq \mathrm{add}_S(B)$, which proves that $C_{S'} \leq C_S$.

We proved that we are able to transform any S to S' that does not include any edge-weight a_3 for which it holds that $\mathrm{C}_S \geq \mathrm{C}_{S'}$. Notice, that this claim also proves that there always exists a solution T to a MinPSC instance (G, w) that does not include any edge with weight a_3.

The cost of T is following. Let $\mathscr{F}' = \{F \mid F \in \mathscr{F}, u \in U, \{u, F\} \in E(T)\}$. That is, $\mathscr{F}' \subseteq \mathscr{F}$ is a subset of vertices of T that pay the cost exactly a_2, whereas vertices $F \in \mathscr{F} \setminus \mathscr{F}'$ pay a_1. Vertex s pays a_1 and all vertices $u \in U$ pay the cost a_2, so the cost of a solution T is

$$\mathrm{C}_T = a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1) \tag{2.6}$$

An example of T is visualized in Figure 2.3.

Now, we can finally show how a solution to (G, w, k) corresponds to a solution to $(U, \mathscr{F}, d)$. We prove the following two implications.

1. If (G, w, k) is a YES instance, then $(U, \mathscr{F}, d)$ is a YES instance.

2. If (G, w, k) is a NO instance, then $(U, \mathscr{F}, d)$ is a NO instance.

(Implication 1) If (G, w, k) is a YES instance, it means there exist a solution T to MinPSC of (G, w) with cost $\mathrm{C}_T \leq k$ such that T does not include any edge of weight a_3. The cost of T is $\mathrm{C}_T = a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1)$, hence we can rewrite $\mathrm{C}_S \leq k$ as

$$a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1) \leq a_1 + |U|a_2 + |\mathscr{F}|a_1 + d(a_2 - a_1),$$

which simplifies to

$$|\mathscr{F}'| \leq d.$$

The subset $\mathscr{F}' \in \mathscr{F}$ is a set cover of U, hence the implication is proved.

(Implication 2) We rewrite the second implication as: if $(U, \mathscr{F}, d)$ is a YES instance, then (G, w, k) is a YES instance.

Suppose there exists a minimum set cover $\mathscr{F}'$ of U with size $|\mathscr{F}'| \leq d$. We construct a connected spanning subgraph S of G with edges

$$E(S) = \{\{s, F\} \mid F \in \mathscr{F}\} \cup \{\{u, F\} \mid u \in U, F \in \mathscr{F}', u \in F\}.$$

The cost of S is

$$\mathrm{C}_S = a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1),$$

for which it holds that

$$\mathrm{C}_S = a_1 + |U|a_2 + |\mathscr{F}|a_1 + |\mathscr{F}'|(a_2 - a_1) \leq a_1 + |U|a_2 + |\mathscr{F}|a_1 + d(a_2 - a_1) \leq k.$$

This proves the second implication.

So far, we have constructed a polynomial-time Karp reduction of d-MinSC to k-MinPSC. Since d-MinSC is an NP-hard problem [13], this proves that k-MinPSC is also NP-hard. ◀

2.2 On Parametrized Complexity

Bentert et al. [2] study the MinPSC problem from the perspective of parameterized computational complexity. In this section, we summarize their results and describe the core ideas of their work.

Two types of parametrization are studied: parametrization by d, the difference between the cost of a solution to MinPSC and the sum of vertex lower bounds from Observation 1.14 of all vertices, and parametrization by c, the number of connected components of the obligatory subgraph w.r.t. vertex lower bounds ℓ. The latter is of greater interest for us, because we use its ideas in our data reduction methods in Section 3.2.

We dwell on the parametrization by d first. Consider a MinPSC instance (G, w), $G = (V, E)$, and its solution $S = (V, F)$. Then, we describe the parameter d by

$$d = \mathrm{C}_S - \sum_{u \in V} \min_{\{u,v\} \in E} w(\{u, v\}). \tag{2.7}$$

Bentert et al. present the following results (under the Exponential Time Hypothesis): There is no constant factor approximation of d that runs in polynomial time and there is no FPT algorithm with respect to parameter d.

2.2.1 Parametrization by the number of connected components of the obligatory subgraph

We have already briefly described a brute-force algorithm that exploits an obligatory subgraph with a running-time of $O(n^{2c})$ steps. Bentert et al. [2] presented two new algorithms that also utilize an obligatory subgraph: a practical randomized algorithm and its derandomized counterpart.

- MinPSC is solvable in $O\left(\ln \frac{1}{\varepsilon} \cdot \left(\frac{4e^2}{\sqrt{2\pi}}\right)^c \cdot \frac{1}{\sqrt{c}} \cdot (9^c m + 4^c nm + nm \log n)\right)$ by a randomized algorithm, where n is the number of vertices, m is the number of edges, ε $(0 < \varepsilon < 1)$ is an upper bound on an error probability, and c is the number of connected components of an obligatory subgraph G_ℓ w.r.t. vertex lower bounds ℓ.
- MinPSC is solvable in $c^{O(c \log c)} n^{O(1)}$, where n is the number of vertices and c is the number of connected components of an obligatory subgraph G_ℓ w.r.t. vertex lower bounds ℓ.

2.2.1.1 Outline of the randomized algorithm

Let $I = (G, w)$ be an instance of MinPSC and let G_ℓ be an obligatory subgraph w.r.t. vertex lower bounds ℓ.

A solution to I is a connected spanning subgraph, therefore we need to add $c - 1$ edges connecting c connected components of G_ℓ. Note that these edges have at most $2c - 2$ endpoints.

First, the graph G is transformed into a padded version $G^\bullet_\ell$. Every connected component of G_ℓ in G is turned into a clique.

▶ **Definition 2.7** (Padded graph)**.** *Let $I = (G, w)$ be a MinPSC instance and let G_ℓ be an obligatory subgraph induced by vertex lower bounds ℓ.*

Edge-weighted graph $(G^\bullet_\ell, w^\bullet_\ell)$ is a padded graph *of G w.r.t. ℓ if $G^\bullet_\ell = (V(G), E^\bullet_\ell)$, where*

$$E^\bullet_\ell = E(G) \cup \{\{u, v\} \mid u, v \in V(G),\ u \text{ and } v \text{ are in the same conected component of } G_\ell\},$$

with edge-weights $w^\bullet_\ell \colon E^\bullet_\ell \to \mathbb{N}$ set as follows

$$w^\bullet_\ell(\{u, v\}) = \begin{cases} 0, & \text{if } u \text{ and } v \text{ are in the same connected component of } G_\ell \\ w(\{u, v\}), & \text{otherwise.} \end{cases} \tag{2.8}$$

In the padded graph $G^\bullet_\ell$ we seek a connected subgraph T that meets the following requirements:

1. The subgraph T has at most $2c - 2$ vertices (because we need to add $c - 1$ edges to G_ℓ to make it connected).
2. The subgraph T contains at least one vertex from each connected component of G_ℓ (because we need to connect all the connected components).
3. The subgraph T minimizes the total additional cost w.r.t. vertex lower bounds ℓ (see (2.9)).

To find such a connected subgraph T, Bentert et al. use the color-coding technique introduced by Alon et al. [14]. Randomly color all the vertices of $G^\bullet_\ell$ using at most $2c - 2$ colors. Search for a connected subgraph T such that each vertex of T is colored by distinct color and all the vertices exhaust all the colors (i.e., a bijection between the colors and vertices of T). Formally, the problem is defined as Min-Power Increment Colorful Connected Subgraph.

▶ **Definition 2.8** (Min-Power Increment Colorful Connected Subgraph (MinPICCS))**.** *Let $G = (V, E)$ be a connected undirected graph with edge weights $w\colon E \to \mathbb{N}$, let ℓ be its vertex lower bounds, let* col$\colon V \to \mathbb{N}$ *be vertex colors, and let $C \subseteq \mathbb{N}$ be a color subset.*

Find a connected spanning subgraph $T = (W, F)$ of G such that col *is a bijection between W and C and such that T minimizes*

$$\sum_{u \in W} \max\Big(0, \max_{\{u,v\} \in F} w\{u, v\} - \ell(u)\Big). \tag{2.9}$$

Now we iteratively choose random vertex coloring col and solve a MinPICCS instance. Out of all these iterations, we choose such a solution with overall minimum objective function (2.9).

Notice, that if we choose the vertex colors col uniformly at random, we may not satisfy the requirement that the sought graph should countain at least one vertex from each connected component of G_ℓ, therefore we would need to execute much more iterations to find an optimal solution satisfying all the requirements. The idea is that we do not color the vertices completely arbitrarily. We color each connected component with unique colors, which directly yields a solution to MinPICCS countaining at least one vertex from each connected component of G_ℓ.

However, it raises the question of how many colors we should use to randomly color each of the connected components of G_ℓ. We simply try all the possibilities. Formally, by $G_\ell^1, G_\ell^2, \ldots, G_\ell^c$ we denote c connected components of G_ℓ. We color vertices of every connected component G_ℓ^i using z_i colors. That is, for each $z_1, z_2, \ldots, z_c \in \mathbb{N} \setminus \{0\}$ such that $\sum_{i=1}^{c} z_i \leq 2c - 2$, we choose pairwise disjoint color sets $C_i \in \mathbb{N}$ such that $|C_i| = z_i$ for $i = 1, 2, \ldots, c$. We color randomly vertices of every connected component G_ℓ^i with colors from C_i and solve the particular MinPICCS instance. We repeat the procedure of random coloring of vertices and solving the associated MinPICCS instance several times.

Out of all the MinPICCS solutions we have gathered, we keep the one, say T, with minimum objective function (2.9). We need to convert T to S, a solution to a MinPSC instance I. Let $E_\ell^\circ := \{\{u, v\} \in E(T) \,|\, u \text{ and } v \text{ connect different connected components of } G_\ell\}$, then $V(S) = V(T)$ and $E(S) = E(G_\ell) \cup E_\ell^\circ$.

It remains to answer how many times should we randomly color the vertices and solve the associated MinPICCS instance. The more repetitions we execute the more likely we are to obtain an optimal solution to MinPSC. Let us denote an upper bound on the error probability by ε, i.e., the probability that we do not find an optimal solution to MinPSC. Bentert et al. proved in [2] that we should perform at least

$$\left\lceil \frac{\ln \varepsilon}{\ln\left(1 - \prod_{i=1}^{c} z_i!/z_i^{z_i}\right)} \right\rceil$$

iterations. If the logarithm in the denominator is undefined, i.e., when $z_i = 1$ for $i = 1, 2, \ldots, c$, perform just one iteration.

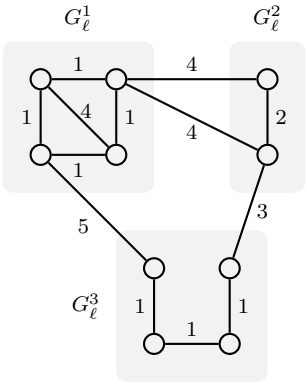
See an example of how the algorithm works in Figure 2.4.

We refer the reader to [2], where the algorithm is described in full detail, including how to solve MinPICCS instances based on dynamic programming and a proof of correctness.

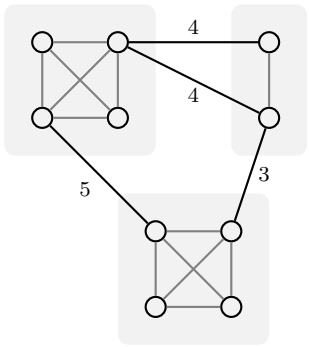
2.2.1.2 A Fixed-parameter Algorithm

Bentert et al. present a derandomized version of the algorithm described in the previous section. To solve actual MinPSC instances, it does not make sense to use the exact derandomized version of the algorithm in comparison with the randomized version. However, it is an interesting matter in terms of computational complexity, as it classifies MinPSC as a fixed-parameter tractable problem when parametrized by the number of connected components of the obligatory subgraph.

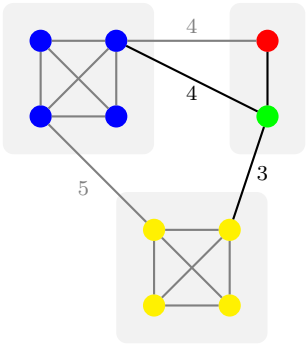
When derandomizing algorithms based on color-coding, it is common to use a technique that involves constructing a perfect hash family of coloring functions [15].



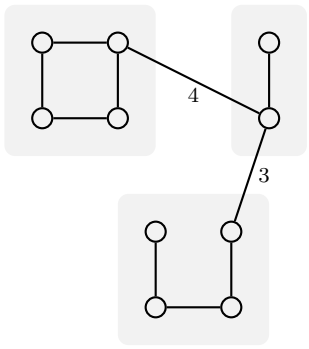
(a) An instance of MinPSC with three connected components G_ℓ^1, G_ℓ^2 and G_ℓ^3 of an obligatory subgraph G_ℓ.



(b) A padded graph $G_\ell^\bullet$ of G. The gray edges connect vertices from the same connected component of G_ℓ and have an edge-weight 0.



(c) An example of a MinPICCS instance when $z_1 = 1$, $z_2 = 2$ and $z_3 = 1$. Black edges form a solution T.



(d) A solution to MinPSC instance I constructed from T in (c).

■ **Figure 2.4** An example of the workflow of the randomized algorithm that uses a color-coding technique. Consider a MinPSC instance $I = (G, w)$ in (a) with vertex lower bounds from Observation 1.14 or Observation 1.15. Its padded version $G_\ell^\bullet$ is visualized in (b). Figure (c) illustrates a possible vertex coloring col and a solution to the related MinPICCS instance. Figure (d) shows the result, a solution to the MinPSC instance I.

▶ **Definition 2.9** ((k,v)-perfect hash family)**.** *A (k,v)-perfect hash family $\mathscr{H}$ is a family of functions $h\colon A \to B$, where $|A| = k$ and $|B| = v$, such that for each subset $W \subseteq A$, where $|W| = v$, there exists a function $h_i \in \mathscr{H}$ that is a bijection between W and B.*

The idea is that we do not rely on random coloring of vertices of a connected component of G_ℓ. Instead, we color the vertices systematically using coloring functions from perfect hash family, which guarantees that at least one function will color the vertices in the desired way.

Let us denote the number of vertices of i-th connected component G_ℓ^i of the obligatory subgraph G_ℓ by n_i, i.e., $n_i = |G_\ell^i|$. It is possible to construct (n_i, z_i)-perfect hash family $\mathscr{H}_i$ in $n_i e^{z_i} z_i^{O(\log z_i)} \log n_i$ time. The size of $\mathscr{H}_i$ is $e^{z_i} z_i^{O(\log z_i)} \log n_i$ [15]. Therefore, when we color the vertices of all connected components, we do so by using $\prod_{i=1}^{c} e^{z_i} z_i^{O(\log z_i)} \log n_i$ functions.

This summarizes the key idea of derandomization of the MinPSC algorithm described above. The running-time analysis is described in [2] in full detail. The running-time is $c^{O(c \log c)} n^{O(1)}$ which classifies MinPSC as FPT when parameterized by the number c of connected components of the obligatory subgraph G_ℓ.

Chapter 3
Optimization Routines

In this chapter, we present two routines, that make the process of solving MinPSC more efficient. In Section 3.1, we present a method using which it is possible to increment vertex lower bounds and therefore obtain more obligatory edges. In this way, the vertex lower bounds induce an obligatory subgraph that may have fewer connected components. We already know that MinPSC is FPT when parameterized by the number c of connected components of the obligatory subgraph G_ℓ, hence decrementation of c is our point of interest.

In Section 3.2, we present a method for identification of redundant edges, i.e., edges we do not need to take into consideration when solving MinPSC.

3.1 Increasing Vertex Lower Bounds

This section introduces a new method for increasing vertex lower bounds for some specific vertices. The idea is as follows. Every vertex must be connected to another vertex in any solution to MinPSC, as it must be a connected spanning subgraph. Suppose we need to connect vertex u to the rest of the graph with either edge e or f, and suppose that $w(e) < w(f)$. Although the weight of f is higher than the weight of e, f may introduce overall less additional cost and therefore should be included in an optimal solution. The following theorem formalizes this idea.

Note that the technique works for arbitrary vertex lower bounds, not just those given in Observation 1.14 or Observation 1.15.

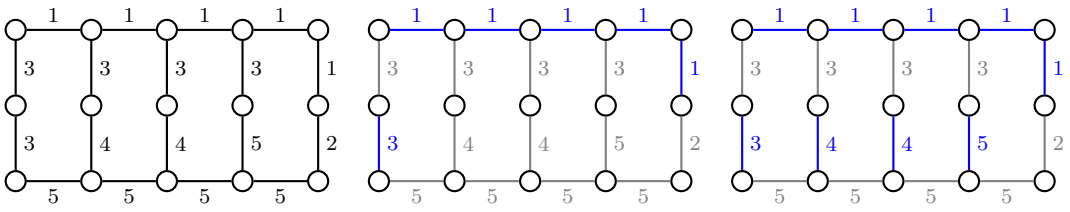
▶ **Theorem 3.1.** *Let $I = (G, w)$ be an instance of MinPSC and let ℓ be vertex lower bounds of I. Let $u \in V(G)$ be a vertex with $k \geq 2$ adjacent vertices $v_1, \ldots, v_k \in V(G)$ such that $w(\{u, v_1\}) < w(\{u, v_2\}) \leq \cdots \leq w(\{u, v_k\})$. If it holds that*

$$w(\{u, v_2\}) \leq \ell(v_2), \tag{3.1}$$

$$w(\{u, v_1\}) = \max_{\{v_1, v_1'\} \in E(G)} w(\{v_1, v_1'\}), \tag{3.2}$$

$$w(\{u, v_2\}) \leq 2w(\{u, v_1\}) - \max_{\substack{\{v_1, v_1'\} \in E(G) \\ v_1' \neq u}} w(\{v_1, v_1'\}), \tag{3.3}$$

then we can set $\ell(u) = w(\{u, v_2\})$.



(a) An instance $I = (G, w)$ of MinPSC

(b) Obligatory edges highlighted in blue

(c) Obligatory edges (highlighted in blue) after incrementation of vertex lower bounds

■ **Figure 3.1** Visualization of how does Theorem 3.1 affect the number of connected components of an obligatory subgraph of a MinPSC instance in (a). Both vertex lower bounds from Observation 1.14 and Observation 1.15 induce the same obligatory subgraph with $c = 9$ connected components (b). When we exploit the incrementation of vertex lower bounds, number of connected components is reduced to $c = 6$ (c).

Proof. Let S be a solution to a MinPSC instance I. If the solution S includes an edge $\{u, v_x\}$ from $\{\{u, v_i\} \mid i = 3, \ldots, k\}$, we set $\ell(u) = w(\{u, v_x\}) \geq w(\{u, v_2\})$ and we are done. Hence, assume that no edge $\{u, v_x\}$ from $\{\{u, v_i\} \mid i = 3, \ldots, k\}$ is in S.

Let $T = S - \{\{u, v_1\}, \{u, v_2\}\}$. Assume that $\ell(u) = w(\{u, v_1\})$. If vertices v_1 and v_2 are not in the same connected component of the graph T, both edges $\{u, v_1\}$ and $\{u, v_2\}$ must be in S, as any solution to MinPSC must be connected. In such a case, we set $\ell(u) = w(\{u, v_2\})$ and we are done.

Therefore we assume that vertices v_1 and v_2 are in the same connected component of the graph T. We need to insert edge $\{u, v_1\}$ or $\{u, v_2\}$ back to the graph T to reconstruct a solution. Let $S_1 = T + \{u, v_1\}$ and let $S_2 = T + \{u, v_2\}$.

The cost of S_1 is

$$\sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_1\}) + \max\big(0, w(\{u, v_1\}) - \mathrm{c}_{\mathrm{T}}(v_1)\big). \tag{3.4}$$

From condition (3.2), we infer that

$$\sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + 2w(\{u, v_1\}) - \mathrm{c}_{\mathrm{T}}(v_1). \tag{3.5}$$

Similarly, we obtain the cost of S_2 by

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_2\}) + \max\big(0, w(\{u, v_2\}) - \mathrm{c}_{\mathrm{T}}(v_2)\big). \tag{3.6}$$

From condition (3.1), we simplify the equation to

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_2\}). \tag{3.7}$$

In order to set $\ell(u) = w(\{u, v_2\})$, we must be sure there exists a solution to MinPSC that contains $\{u, v_2\}$. It means that the cost of S_1 must be higher than or equal to the cost of S_2, that is

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) \leq \sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v), \tag{3.8}$$

which, using equations (3.5) and (3.7), gives us

$$w(\{u, v_2\}) \leq 2w(\{u, v_1\}) - \mathrm{c_T}(v_1). \tag{3.9}$$

Notice that (3.9) always holds because of assumption (3.3). This proves that we can set $\ell(u) = w(\{u, v_2\})$. ◀

According to Theorem 3.1, there are special cases, where it may be more effective to include edge f in a solution instead of edge e, even though $w(e) < w(f)$. Figure 3.1 illustrates such a scenario.

This method of vertex lower bounds incrementation works for any vertex lower bounds. We explore if this idea of vertex lower bounds incrementation could be used in other scenarious, not only for vertices with two edges with the smallest edge-weight as in Theorem 3.1.

We present the following theorem, which exploits exactly the same idea as in Theorem 3.1, but is not as universal. We cannot apply such a vertex lower bounds incrementation method for arbitrary vertex lower bounds, but only to those stated in Observation 1.14 and Observation 1.15.

▶ **Theorem 3.2.** *Let $I = (G, w)$ be an instance of MinPSC and let ℓ be vertex lower bounds of I from Observation 1.14 or Observation 1.15. Let $u \in V(G)$ be a vertex in G and let $v_1, \ldots, v_k \in V(G)$ be all vertices from one connected component of $G - u$ which are connected to vertex u in G, for which it holds that $w(\{u, v_1\}) < w(\{u, v_2\}) \leq \cdots \leq w(\{u, v_k\})$.*

If it holds that

$$w(\{u, v_2\}) \leq \ell(v_2), \tag{3.10}$$

$$w(\{u, v_1\}) = \max_{\{v_1, v_1'\} \in E(G)} w(\{v_1, v_1'\}), \tag{3.11}$$

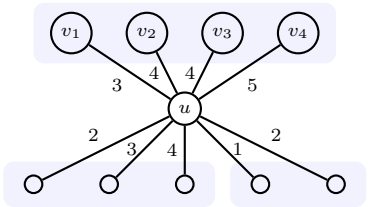
$$w(\{u, v_2\}) \leq 2w(\{u, v_1\}) - \max_{\substack{\{v_1, v_1'\} \in E(G) \\ v_1' \neq u}} w(\{v_1, v_1'\}) \tag{3.12}$$

and if there does not exists an edge $\{u, x\}$ in G such that

$$w(\{u, v_1\}) < w(\{u, x\}) < w(\{u, v_2\}), \tag{3.13}$$

then we can set $\ell(u) = w(\{u, v_2\})$.

See Figure 3.2, in which we show vertices v_1 and v_2 that meet the assumptions from Theorem 3.2.



■ **Figure 3.2** A graph setup for potential incrementation of vertex lower bounds of u. The three blue rectangles visualize connected components of $G - u$.

Proof. (Theorem 3.2) Let S be a solution to a MinPSC instance I. If the solution S includes an edge $\{u, x\}$ with $w(\{u, x\}) > w(\{u, v_1\})$, we set $\ell(u) = w(\{u, v_x\}) \geq w(\{u, v_2\})$ and we are done. Hence, assume that no edge $\{u, v_x\}$ with $w(\{u, x\}) > w(\{u, v_1\})$ is in S.

Let $T = S - \{u, v_1\}$. If vertices v_1 and v_2 are not in the same connected component of the graph T, both edges $\{u, v_1\}$ and $\{u, v_2\}$ must be in S, as any solution to MinPSC must be connected. In such a case, we set $\ell(u) = w(\{u, v_2\})$ and we are done.

Therefore we assume that vertices v_1 and v_2 are in the same connected component of the graph T. We need to insert edge $\{u, v_1\}$ or $\{u, v_2\}$ to the graph T to reconstruct a solution. Let $S_1 = T + \{u, v_1\}$ and let $S_2 = T + \{u, v_2\}$.

The cost of S_1 is

$$\sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_1\}) + \max\bigl(0, w(\{u, v_1\}) - \mathrm{c}_{\mathrm{T}}(v_1)\bigr). \tag{3.14}$$

From condition (3.11), we infer that

$$\sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + 2w(\{u, v_1\}) - \mathrm{c}_{\mathrm{T}}(v_1). \tag{3.15}$$

Similarly, we obtain the cost of S_2 by

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_2\}) + \max\bigl(0, w(\{u, v_2\}) - \mathrm{c}_{\mathrm{T}}(v_2)\bigr). \tag{3.16}$$

From condition (3.10), we simplify the equation to

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) = \sum_{v \in V(T)} \mathrm{c}_{\mathrm{T}}(v) + w(\{u, v_2\}). \tag{3.17}$$

In order to set $\ell(u) = w(\{u, v_2\})$, we must be sure there exists a solution to MinPSC that contains $\{u, v_2\}$. It means that the cost of S_1 must be higher than or equal to the cost of S_2, that is

$$\sum_{v \in V(S_2)} \mathrm{c}_{\mathrm{S}_2}(v) \leq \sum_{v \in V(S_1)} \mathrm{c}_{\mathrm{S}_1}(v), \tag{3.18}$$

which, using equations (3.15) and (3.17), gives us

$$w(\{u, v_2\}) \leq 2w(\{u, v_1\}) - \mathrm{c}_{\mathrm{T}}(v_1). \tag{3.19}$$

Notice that (3.19) always holds because of assumption (3.12). This proves that we can set $\ell(u) = w(\{u, v_2\})$. ◀

3.2 Data Reduction

We study the possibility of reducing the number of edges in an instance of MinPSC, while preserving the existence of an optimal solution.

Consider an instance $I = (G, w)$ of MinPSC and an upper bound M on the cost C of a solution to I. Basic data reduction rule is to delete any edge $e \in E(G)$ such that $w(e) > M$, because e cannot possibly be in any solution of I.

Bentert et al. [2] took the thought further and incorporated vertex lower bounds into the rule.

▶ Observation 3.3. Let ℓ be vertex lower bounds of I. There exists a solution S to I that does not contain an edge $\{u, v\} \in E(G)$ such that

$$\sum_{x \in V(G)} \ell(x) + \max\Bigl(0, w\bigl(\{u, v\} - \ell(u)\bigr)\Bigr) + \max\Bigl(0, w\bigl(\{u, v\} - \ell(v)\bigr)\Bigr) > M, \tag{3.20}$$

hence we can delete edge $\{u, v\}$ from G.

Proof. There exists an optimal solution S with $\mathrm{c}_S(u) \geq \ell(u)$ for all $u \in V(S)$. It follows directly from Definition 1.13. For contradiction, let us assume $\{u, v\} \in E(S)$ satisfying the inequality (3.20).

Notice that we can rewrite the left-hand side of (3.20) as

$$\sum_{x \in V(G)} \ell(x) + \max\Big(0, w\big(\{u,v\} - \ell(u)\big)\Big) + \max\Big(0, w\big(\{u,v\} - \ell(v)\big)\Big) = \sum_{x \in V(G)\setminus\{u,v\}} \ell(x) + \max\Big(w\big(\{u,v\}, \ell(u)\big)\Big) + \max\Big(w\big(\{u,v\}, \ell(v)\big)\Big).$$

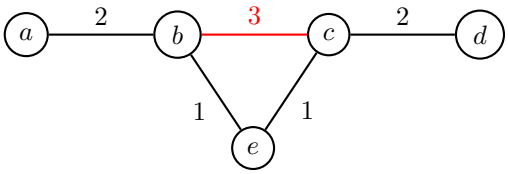
Because it holds that $\mathrm{c}_S(u) \geq \ell(u)$ for all $u \in V(S)$, $\mathrm{c}_S(u) \geq w\big(\{u,v\}\big)$ and $\mathrm{c}_S(v) \geq w\big(\{u,v\}\big)$, the left-hand side of (3.20) gives us a lower-bound on the cost of S

$$\sum_{x \in V(G)} \ell(x) + \max\Big(0, w\big(\{u,v\} - \ell(u)\big)\Big) + \max\Big(0, w\big(\{u,v\} - \ell(v)\big)\Big) \leq \sum_{u \in V(S)} \mathrm{c}_S(u) \leq M.$$

This contradicts our assumption (3.20). ◀

The cost of a minimum spanning tree (MST), in terms of a solution to MinPSC, is commonly used as an upper bound on the cost of a solution to MinPSC. We use it in Example 3.4.

▶ **Example 3.4.** Consider a MinPSC instance $I = (G, w)$ in Figure 3.3 and naive vertex lower bounds from Observation 1.14. We show how does the data reduction technique (3.20) work in practice.



■ **Figure 3.3** An example of the edge reduction technique (3.20).

Let T be a minimum spanning tree of G. For the cost C_T it holds that $\mathrm{C}_T = 9$. It gives us an upper bound $M = \mathrm{C}_T$ on the cost of a solution S to I. We analyze, if we can delete an edge $\{b, c\}$ from graph G. For vertex lower bounds ℓ it holds that $\ell(a) = 2$, $\ell(b) = 1$, $\ell(c) = 1$, $\ell(d) = 2$ and $\ell(e) = 1$. We calculate the left-hand side of (3.20)

$$2 + 1 + 1 + 2 + 1 + \max\big(0, 3 - 1\big) + \max\big(0, 3 - 1\big) = 11 > 9 = M, \tag{3.21}$$

hence we can delete an edge $\{b, c\}$ from G.

Bentert et al. [2] analyzed the edge reduction method (3.20) (using the cost of MST as an upper-bound M) on certain triangular grid graphs and were able to delete up to 75% of edges.

We can reduce more edges by obtaining a tighter upper bound on the cost of a MinPSC solution. The weight M' of an MST of graph G gives us a 2-approximation on the cost C of an optimal solution to MinPSC (i.e., $C \leq 2M'$) [1]. Park et al. [16] studied two approximation schemes based on Kruskal's and Jarnik's algorithm for MST. Both algorithms provide a 2-approximation on the MinPSC cost, but perform much better than MST in experiments. Other approximation schemes have been studied, such as the one with approximation ratio of $\frac{11}{6}$ [17]. The question arises, how accurate an approximation can we actually find? Fuchs proved that it is NP-hard to approximate MinPSC within a factor of $1 + 1/260$ [18].

3.2.1 An Edge Reduction Technique

We look at reducing the edges in more detail and present new data reduction technique.

To simplify the following definitions, we denote a set of edges that connects two components of an obligatory subgraph G_ℓ in graph G by $\mathrm{e}(G^i_\ell, G^j_\ell)$, that is,

$$\mathrm{e}(G^i_\ell, G^j_\ell) \coloneqq \big\{\{u,v\} \in E(G) \mid u \in V(G^i_\ell),\ v \in V(G^j_\ell)\big\}.$$

▶ **Definition 3.5** (Maximum additional cost)**.** *Let $I = (G, w)$ be an instance of MinPSC, let ℓ be vertex lower bounds of I, let G_ℓ be an obligatory subgraph of I w.r.t. ℓ and let $e \in E(G) \setminus E(G_\ell)$. Denote by G_e an obligatory subgraph with additional edge e, that is, $G_e = G_\ell + e$.*

Maximum additional cost *is a function* $\mathrm{add}_{\max}\colon E \to \mathbb{N}$ *defined as*

$$\mathrm{add}_{\max}(e) = \mathrm{add}_{G_e}(e). \tag{3.22}$$

The idea of maximum additional cost is following. We know that there exists a solution S to MinPSC of (G, w) that includes the obligatory subgraph G_ℓ. Maximum additional cost of an edge e answers the question of what maximum cost can e introduce to a solution S. That is, maximum additional cost provides an upper bound on how much of an additional cost will an edge e introduce to a solution S.

Notice that for maximum additional cost of an edge $e = \{u, v\}$ it holds that

$$\mathrm{add}_{\max}(\{u,v\}) = \max\big(0, w(\{u,v\}) - \mathrm{c}_{G_\ell}(u)\big) + \max\big(0, w(\{u,v\}) - \mathrm{c}_{G_\ell}(v)\big). \tag{3.23}$$

Similarly, we define the notion of minimum additional cost.

▶ **Definition 3.6** (Minimum additional cost)**.** *Let $I = (G, w)$ be an instance of MinPSC, let ℓ be vertex lower bounds of I, let G_ℓ be an obligatory subgraph of I w.r.t. ℓ with c connected components $G^1_\ell, G^2_\ell, \ldots, G^c_\ell$ and let $e \in \mathrm{e}(G^1_\ell, G^2_\ell)$. Finally, let L be a subgraph of G where $V(L) = V(G)$ and*

$$E(L) = E(G_\ell) \cup \left(\bigcup_{\substack{i,j=1,\ldots,c \\ i \neq j}} \mathrm{e}(G^i_\ell, G^j_\ell) \right) \setminus \mathrm{e}(G^1_\ell, G^2_\ell).$$

Minimum additional cost *is a function* $\mathrm{add}_{\min}\colon E \to \mathbb{N}$ *defined as*

$$\mathrm{add}_{\min}(e) = \mathrm{add}_{G_{L+e}}(e). \tag{3.24}$$

Minimum additional cost gives us a lower bound on how much of an additional cost will an edge $e = \{u, v\}$ introduce to an optimal solution, in which no other edge connects components G^1_ℓ with G^2_ℓ.

For minimum additional cost of an edge $e = \{u, v\}$ it holds that

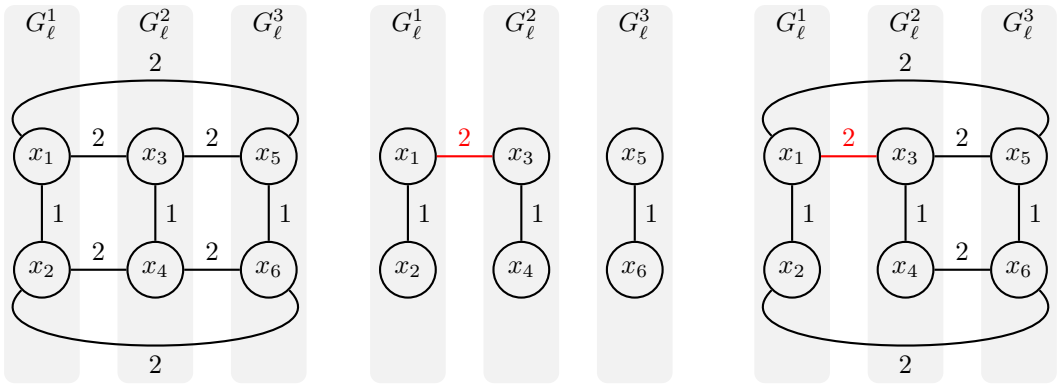
$$\mathrm{add}_{\min}(\{u,v\}) = \max\big(0, w(\{u,v\}) - \mathrm{c}_L(u)\big) + \max\big(0, w(\{u,v\}) - \mathrm{c}_L(v)\big). \tag{3.25}$$

Figure 3.4 illustrates maximum and minimum additional cost on an example.

▶ **Theorem 3.7.** *Let $I = (G, w)$ be an instance of MinPSC, let ℓ be vertex lower bounds of I and let G_ℓ be an obligatory subgraph of I w.r.t. ℓ. Denote c connected components of G_ℓ by $G^1_\ell, G^2_\ell, \ldots, G^c_\ell$. Let e and f be two distinct edges from $\mathrm{e}(G^1_\ell, G^2_\ell)$. If*

$$\mathrm{add}_{\max}(e) \leq \mathrm{add}_{\min}(f), \tag{3.26}$$

then there exists a solution to MinPSC instance I that does not include f. Furthermore, any solution to MinPSC instance $\big((V(G), E(G) \setminus \{f\}), w\big)$ is a solution to I as well.



(a) Graph G with highlighted connected components of an obligatory subgraph G_ℓ. Both vertex lower bounds from Observation 1.14 and Observation 1.15 induces such G_ℓ.

(b) Obligatory subgraph G_ℓ and one additional edge $\{x_1, x_3\}$ highlighted in red.

(c) Graph L from Definition 3.6 and one additional edge $\{x_1, x_3\}$ highlighted in red.

■ **Figure 3.4** An example of maximum and minimum additional cost. Consider an instance (G, w) of MinPSC and its obligatory subgraph w.r.t. ℓ (see (a)). Maximum additional cost $\mathrm{add}_{\max}$ of edge $\{x_1, x_2\}$ is 2 ((b) illustrates parts of G needed to calculate $\mathrm{add}_{\max}(\{x_1, x_2\})$). Minimum additional cost $\mathrm{add}_{\min}$ of edge $\{x_1, x_2\}$ is 0 ((c) illustrates parts of G needed to calculate $\mathrm{add}_{\min}(\{x_1, x_2\})$).

Proof. Assume S is a solution to MinPSC instance I and assume $f \in E(S)$. Without loss of generality, assume that f is the only edge connecting G_ℓ^1 and G_ℓ^2 in S because all other edges from $E(S) \cap \mathrm{e}(G_\ell^1, G_\ell^2)$ could be deleted and S would still be a solution to I. The removal of edges cannot introduce additional cost and S remains connected spanning subgraph of G, since $f \in E(S) \cap \mathrm{e}(G_\ell^1, G_\ell^2)$.

Construct a new graph $T = \big(V(S), (E(S) \cup \{e\}) \setminus \{f\}\big)$. Notice that T is connected spanning subgraph of G, because e connects the same connected components of G_ℓ as the removed edge f.

We show that

$$\mathrm{C}_T \leq \mathrm{C}_S, \tag{3.27}$$

which implies that T is a solution to MinPSC instance I as well. First, expand both sides of equation (3.27) as

$$\sum_{u \in V(T)} \mathrm{c}_T(u) \leq \sum_{u \in V(S)} \mathrm{c}_S(u). \tag{3.28}$$

Let $P = \big(V(S), E(S) \setminus \{f\}\big)$. We split the left-hand side of (3.28)

$$\sum_{u \in V} \mathrm{c}_\mathrm{T}(u) = \sum_{u \in V} \mathrm{c}_\mathrm{P}(u) + \Big(\max\big(0, w(e) - \mathrm{c}_\mathrm{P}(u_1)\big) + \max\big(0, w(e) - \mathrm{c}_\mathrm{P}(v_1)\big)\Big), \tag{3.29}$$

and the right-hand side of (3.28)

$$\sum_{u \in V} \mathrm{c}_\mathrm{S}(u) = \sum_{u \in V} \mathrm{c}_\mathrm{P}(u) + \Big(\max\big(0, w(f) - \mathrm{c}_\mathrm{P}(u_2)\big) + \max\big(0, w(f) - \mathrm{c}_\mathrm{P}(v_2)\big)\Big), \tag{3.30}$$

where u_1 and v_1 are the vertices incident to e and u_2 and v_2 are the vertices incident to f.

The first terms of the sums of (3.29) and (3.30) are identical, hence we rewrite (3.28) as

$$\begin{aligned} &\max\big(0, w(e) - \mathrm{c}_\mathrm{P}(u_1)\big) + \max\big(0, w(e) - \mathrm{c}_\mathrm{P}(v_1)\big) \\ &\qquad \leq \max\big(0, w(f) - \mathrm{c}_\mathrm{P}(u_2)\big) + \max\big(0, w(f) - \mathrm{c}_\mathrm{P}(v_2)\big). \end{aligned} \tag{3.31}$$

The following chain of inequalities proves (3.31):

$$\begin{aligned}\max\big(0, w(e) - \mathrm{c_P}(u_1)\big) + \max\big(0, w(e) - \mathrm{c_P}(v_1)\big) &\\ \leq \max\big(0, w(e) - \mathrm{c}_{\mathrm{G}_\ell}(u_1)\big) + \max\big(0, w(e) - \mathrm{c}_{\mathrm{G}_\ell}(v_1)\big) &\\ \leq \max\big(0, w(f) - \mathrm{c_L}(u_2)\big) + \max\big(0, w(f) - \mathrm{c_L}(v_2)\big) &\\ \leq \max\big(0, w(f) - \mathrm{c_P}(u_2)\big) + \max\big(0, w(f) - \mathrm{c_P}(v_2)\big)&\end{aligned}$$

The first inequality holds because $\mathrm{c_P}(u_1) \geq \mathrm{c}_{\mathrm{G}_\ell}(u_1)$ and $\mathrm{c_P}(v_1) \geq \mathrm{c}_{\mathrm{G}_\ell}(v_1)$, since every edge in G_ℓ is in P as well. The second inequality holds, because it is is assumption (3.26) written out using equations (3.23) and (3.25). The third inequality holds because $\mathrm{c_L}(u_2) \geq \mathrm{c_P}(u_2)$ and $\mathrm{c_L}(v_2) \geq \mathrm{c_P}(v_2)$, since every edge in P is in L as well.

This proves that there exists a solution to MinPSC instance I that does not include f. This implies that any solution to MinPSC instance $\big((V(G), E(G) \setminus \{f\}), w\big)$ is a solution to I as well. ◀

Theorem 3.7 formalizes an edge reduction method, which exploits the idea that if some edge e always induces less or equal additional cost to an optimal solution to MinPSC than some other edge f, then we can delete edge f.

▶ **Example 3.8** (Reduction of edges)**.** We demonstrate the described edge reduction method on a MinPSC instance $I = (G, w)$ illustrated in Figure 3.5a.

According to the table 3.1, we can delete the following edges. Between connected components G_ℓ^1 and G_ℓ^2 we can delete edges a, b, and c because $\mathrm{add}_{\max}(d) \leq \mathrm{add}_{\min}(a)$, $\mathrm{add}_{\max}(d) \leq \mathrm{add}_{\min}(b)$ and $\mathrm{add}_{\max}(d) \leq \mathrm{add}_{\min}(b)$. Between connected components G_ℓ^2 and G_ℓ^3 we can delete e because $\mathrm{add}_{\max}(f) \leq \mathrm{add}_{\min}(e)$.

G_ℓ^1–G_ℓ^2	$\mathrm{add}_{\max}$	$\mathrm{add}_{\min}$
a	1	1
b	3	2
c	5	5
d	1	0

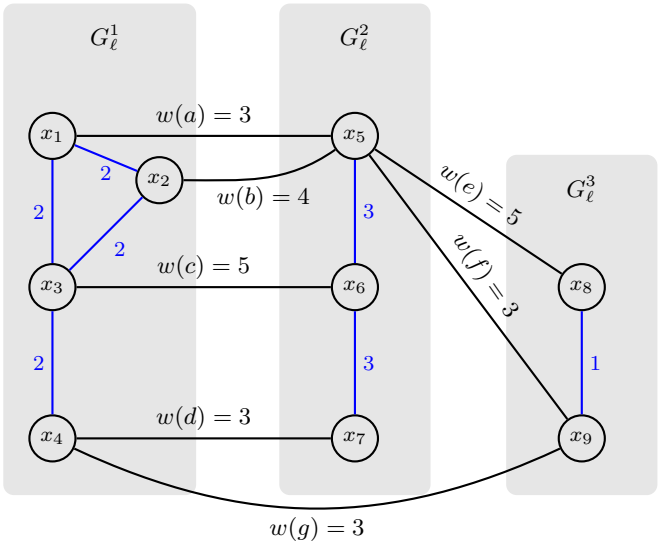
G_ℓ^2–G_ℓ^3	$\mathrm{add}_{\max}$	$\mathrm{add}_{\min}$
e	6	5
f	2	0

■ **Table 3.1** Maximum additional cost $\mathrm{add}_{\max}$ and minimum additional cost $\mathrm{add}_{\min}$ of edges between components G_ℓ^1 and G_ℓ^2 and between components G_ℓ^2 and G_ℓ^3.

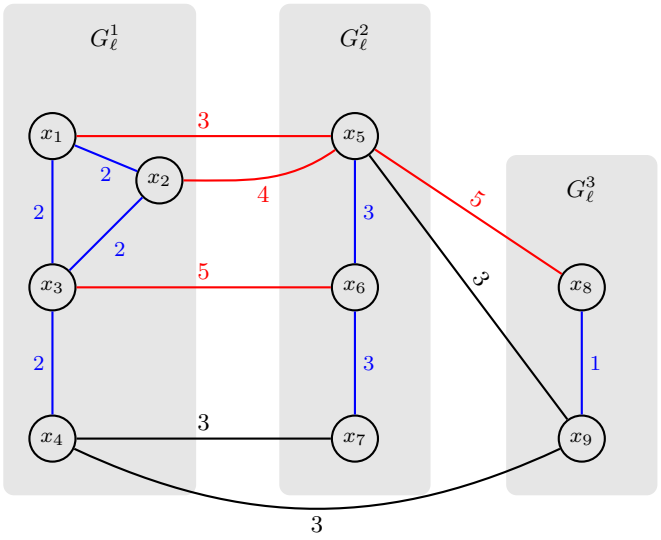
Figure 3.5b illustrates our MinPSC instance after application of the data reduction. This is an example of a perfect scenario when only single edge between two connected components of an obligatory subgraph remains.

3.2.2 On Redundant Vertices

To this point, we analyzed only techniques involving reductions of edges, but Bentert et al. [2] also proposed a basic vertex reduction method. The method is applicable, when we solve MinPSC using the algorithm stated in Section 2.2.1.1. In the algorithm, we find an obligatory subgraph G_ℓ and turn its connected components into cliques, i.e., we obtain the padded graph $G_\ell^\bullet$. Next, we need to connect the connected components of the obligatory subgraph to get a connected spanning subgraph. In terms of the algorithm, it means to solve several MinPICCS instances. Notice that there always exists a solution to any MinPICCS instance that does not include any vertex that is adjacent only to vertices from its own connected component of G_ℓ. Hence, we can delete such vertices, that is, all the vertices that are in $G_\ell^\bullet$ adjacent only to vertices from their own connected component of G_ℓ.

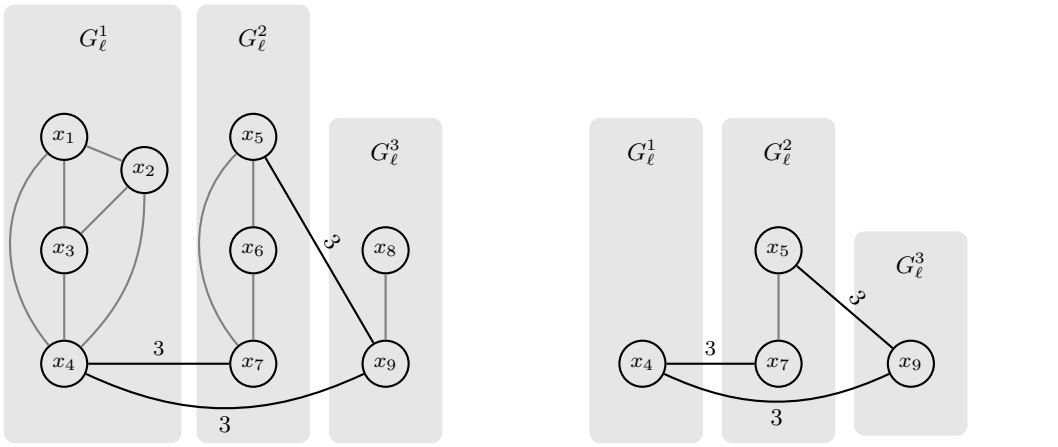


(a) An example of a MinPSC instance. Edges marked in blue are obligatory w.r.t. ℓ from Observation 1.14 or Observation 1.15. Edges marked in black are those edges we analyze for possible reduction.



(b) Edges marked in red can be deleted.

■ **Figure 3.5** An example of reduction of edges



(a) A padded graph $G_\ell^\bullet$

(b) A padded graph $G_\ell^\bullet$ after reduction of vertices x_1, x_2, x_3 and x_6.

■ **Figure 3.6** An example of reduction of vertices

▶ **Example 3.9.** We demontrate this technique on the reduced MinPSC instance from Example 3.8. Figure 3.6a visualizes a padded graph $G_\ell^\bullet$. Vertices x_1, x_2, x_3 and x_6 have only neighbours in the same connected component of G_ℓ. Hence, we can delete them (Figure 3.6b).

We could ask, how effective reduction techniques can possibly be for MinPSC. Bentert et al. [2] show that MinPSC has no problem kernel of size polynomial in the number of connected components c of G_ℓ, even when naive vertex lower bounds ℓ from Observation 1.14 are considered.

Conclusion

In this thesis we studied Min-Power Symmetric Connectivity with focus on the approach taken by Bentert et al. [2]. The goal was to further fine-tune the proposed algorithms by Bentert et al. [2].

Two new methods for vertex lower bounds incrementation were introduced. One method is suitable for arbitrary vertex lower bounds, while the other is restricted to specific vertex lower bounds but is applicable in more general cases.

We also studied the possibility of data reduction and found a new method for identifying redundant edges that can be deleted before the algorithm reaches the parts with exponential time complexity.

Furthermore, a new result on computational complexity was introduced. Min-Power Symmetric Connectivity (its decision version) is NP-hard for instances of complete graphs with three distinct edge-weights $a_1, a_2, a_3 \in \mathbb{N}$ with $a_1 < a_2 < a_3$ and $a_3 \geq 2a_2 - a_1$.

Discussion

We have provided theoretical background for possible vertex lower bounds incrementation and data reduction. It would be interesting to run simulations and to benchmark these methods. When it comes to the methods concerning incrementation of vertex lower bounds, how many vertices meet our requirements so that the methods can be applied? It would be particularly interesting to analyze specific instances of grid graphs, since these layouts are used in real-life scenarious. When it comes to identification of redundant edges, even naive methods that we described in this thesis, were able to reduce up to 75% of the edges in certain instances of triangular grid graphs. What ratio of edges can we expect to reduce using our newly proposed edge reduction technique?

This raises another question, this time from a computational complexity perspective, that might be interesting for further investigation. What specific classes of graphs result in a notable reduction in the size of the input (yielding, for example, a polynomial kernel) when these methods are applied?

Appendix A
Acronyms

ETH	Exponential Time Hypothesis
FPT	Fixed-Parameter Tractable
MST	Minimum Spanning Tree

Bibliography

1. CALINESCU, Gruia; MANDOIU, Ion; ZELIKOVSKY, A. Symmetric Connectivity with Minimum Power Consumption in Radio Networks. *Proc. 2nd IFIP International Conference on Theoretical Computer Science*. 2002, vol. 223, pp. 119–130. ISBN 978-1-4757-5275-5. Available from DOI: 10.1007/978-0-387-35608-2_11.
2. BENTERT, Matthias; BEVERN, René van; NICHTERLEIN, André; NIEDERMEIER, Rolf; SMIRNOV, Pavel V. Parameterized Algorithms for Power-Efficiently Connecting Wireless Sensor Networks: Theory and Experiments. *INFORMS Journal on Computing*. 2022, vol. 34, no. 1, pp. 55–75. ISSN 1526-5528. Available from DOI: 10.1287/ijoc.2020.1045.
3. MATOUŠEK, Jiří; NEŠETŘIL, Jaroslav. *Kapitoly z diskrétní matematiky*. Univerzita Karlova v Praze, Nakladatelství Karolinum, 2009.
4. DIESTEL, Reinhard. *Graph Theory*. 5th. Springer Publishing Company, Incorporated, 2017. ISBN 3662536218.
5. ARORA, Sanjeev; BARAK, Boaz. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
6. AARONSON, Scott. 2005. Available also from: https://complexityzoo.net/.
7. IMPAGLIAZZO, R.; PATURI, R. Complexity of k-SAT. In: *Proceedings. Fourteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference) (Cat.No.99CB36317)*. 1999, pp. 237–240. Available from DOI: 10.1109/CCC.1999.766282.
8. CYGAN, Marek; FOMIN, Fedor V; KOWALIK, Łukasz; LOKSHTANOV, Daniel; MARX, Dániel; PILIPCZUK, Marcin; PILIPCZUK, Michał; SAURABH, Saket. *Parameterized algorithms*. Vol. 5. Springer, 2015. No. 4.
9. PLOTNIKOV, Roman; ERZIN, A.; MLADENOVIC, Nenad. Approximation algorithms for the min-power symmetric connectivity problem. 2016, vol. 1776, p. 050012. Available from DOI: 10.1063/1.4965333.
10. ZALYUBOVSKIY, Vyacheslav; ERZIN, Adil; ASTRAKOV, Sergey; CHOO, Hyunseung. Energy-efficient Area Coverage by Sensors with Adjustable Ranges. *Sensors*. 2009, vol. 9, no. 4, pp. 2446–2460. ISSN 1424-8220. Available from DOI: 10.3390/s90402446.
11. KIROUSIS, Lefteris M.; KRANAKIS, Evangelos; KRIZANC, Danny; PELC, Andrzej. Power consumption in packet radio networks. *Theoretical Computer Science*. 2000, vol. 243, no. 1, pp. 289–305. ISSN 0304-3975. Available from DOI: https://doi.org/10.1016/S0304-3975(98)00223-0.

12. ERZIN, A.; PLOTNIKOV, Roman; SHAMARDIN, Yu. On some polynomially solvable cases and approximate algorithms in the optimal communication tree construction problem. *Journal of Applied and Industrial Mathematics*. 2013, vol. 7. Available from DOI: 10.1134/S1990478913020038.

13. KARP, Richard M. Reducibility Among Combinatorial Problems. In: MILLER, Raymond E.; THATCHER, James W. (eds.). *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*. Plenum Press, New York, 1972, pp. 85–103. The IBM Research Symposia Series. Available from DOI: 10.1007/978-1-4684-2001-2_9.

14. ALON, Noga; YUSTER, Raphael; ZWICK, Uri. Color-coding. *J. ACM*. 1995, vol. 42, no. 4, pp. 844–856. ISSN 0004-5411. Available from DOI: 10.1145/210332.210337.

15. CYGAN, Marek; FOMIN, Fedor V.; KOWALIK, Łukasz; LOKSHTANOV, Daniel; MARX, Dániel; PILIPCZUK, Marcin; PILIPCZUK, Michał; SAURABH, Saket. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. Available from DOI: 10.1007/978-3-319-21275-3.

16. PARK, Joongseok; SAHNI, Sartaj. Power Assignment for Symmetric Communication in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks*. 2009, vol. 5. Available from DOI: 10.1080/15501320701343992.

17. ALTHAUS, Ernst; CALINESCU, Gruia; MANDOIU, Ion; PRASAD, Sushll; TCHERVENSKI, N.; ZELIKOVSKY, A. Power Efficient Range Assignment for Symmetric Connectivity in Static Ad Hoc Wireless Networks. *Wireless Networks*. 2006, vol. 12, pp. 287–299. Available from DOI: 10.1007/s11276-005-5275-x.

18. FUCHS, Bernhard. On the Hardness of Range Assignment Problems. In: CALAMONERI, Tiziana; FINOCCHI, Irene; ITALIANO, Giuseppe F. (eds.). *Algorithms and Complexity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 127–138. ISBN 978-3-540-34378-3.