Parameterized Algorithms for Power-Efficiently Connecting Wireless Sensor Networks: Theory and Experiments*

Matthias Bentert[1], René van Bevern[2], André Nichterlein[1], Rolf Niedermeier[1], and Pavel V. Smirnov[2]

[1]Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany,
{matthias.bentert, andre.nichterlein, rolf.niedermeier}@tu-berlin.de
[2]Department of Mechanics and Mathematics, Novosibirsk State University, Novosibirsk, Russian Federation,
rvb@nsu.ru, p.smirnov@g.nsu.ru

September 4, 2020

Abstract

We study an NP-hard problem motivated by energy-efficiently maintaining the connectivity of a symmetric wireless communication network: Given an edge-weighted n-vertex graph, find a connected spanning subgraph of minimum cost, where the cost is determined by letting each vertex pay the most expensive edge incident to it in the subgraph. On the negative side, we show that $o(\log n)$-approximating the difference d between the optimal solution cost and a natural lower bound is NP-hard and that, under the Exponential Time Hypothesis, there are no exact algorithms running in $2^{o(n)}$ time or in $f(d) \cdot n^{O(1)}$ time for any computable function f. Moreover, we show that the special case of connecting c network components with minimum additional cost generally cannot be polynomial-time reduced to instances of size $c^{O(1)}$ unless the polynomial-time hierarchy collapses. On the positive side, we provide an algorithm that reconnects $O(\log n)$ connected components with minimum additional cost in polynomial time. These algorithms are motivated by application scenarios of monitoring areas or where an existing sensor network may fall apart into several connected components due to sensor faults. In experiments, the algorithm outperforms CPLEX with known ILP formulations when n is sufficiently large compared to c.

Keywords: monitoring areas, reconnecting sensor networks, parameterized complexity analysis, approximation hardness, parameterization above lower bounds, color-coding, experimental comparison

1 Introduction

We consider a well-studied problem arising in the context of power-efficiently maintaining the connectivity of symmetric wireless sensor communication networks. It is formally defined as follows (see Figure 1 for an example).

Problem 1.1 (Min-Power Symmetric Connectivity (MinPSC)).
Input: A connected undirected graph $G = (V, E)$ with n vertices, m edges, and edge weights $w\colon E \to \mathbb{N}$.
Goal: Find a connected spanning subgraph $T = (V, F), F \subseteq E$, of G that minimizes

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

*A preliminary version of this article appeared in the *Proceedings of the 13th International Symposium on Algorithms and Experiments for Wireless Networks (ALGOSENSORS'17), Vienna, Austria* [5]. This extended version is enhanced by an experimental evaluation and by results translated from the last author's Master's thesis [44]: it contains stronger lower-bound results and a corrected and accelerated algorithm.

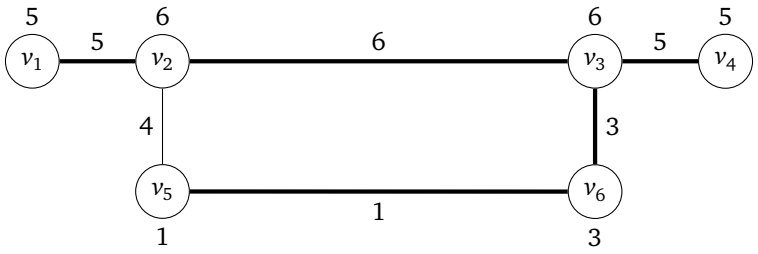


Figure 1: A graph with positive edge weights and an optimal solution (bold edges). Each vertex pays the most expensive edge incident to it in the solution (the numbers next to the vertices). The cost of the solution is the sum of the costs paid by the vertices. Note that the optimal solution has cost 26 while a minimum spanning tree (using edge $\{v_2, v_5\}$ instead of edge $\{v_2, v_3\}$) has cost 27 (as a MinPSC solution).

Table 1: Overview on our results, using the following terminology: n—number of vertices, m—number of edges, d—difference between the optimal solution cost and a lower bound (see Problem 3.3), c—number of connected components of the subgraph consisting of obligatory edges (see Definition 4.1). MinPSC-ALB is the problem of computing the minimum value of d (Problem 3.3), d-PSC-ALB is the corresponding decision problem.

	Problem	Result	Reference
Sec. 3	MinPSC-ALB	NP-hard to approximate within a factor of $o(\log n)$	Theorem 3.4(i)
	d-PSC-ALB	$W[2]$-hard when parameterized by d	Theorem 3.4(ii)
	k-PSC	not solvable in $2^{o(n)}$ time unless ETH fails	Theorem 3.4(iii)
Sec. 4	MinPSC	solvable in $\ln 1/\varepsilon \cdot (4e^2/\sqrt{2\pi})^c \cdot 1/\sqrt{c} \cdot O(9^c m + 4^c nm + nm \log n)$ time with error probability at most ε	Theorem 4.3
	MinPSC	solvable in $c^{O(c \log c)} \cdot n^{O(1)}$ time	Theorem 4.3
	MinPSC	solvable in $O(3^n \cdot m)$ time	Proposition 4.5
	k-PSC	WK[1]-hard parameterized by c	Theorem 4.19

Herein, a *spanning* subgraph of a graph $G = (V, E)$ is a subgraph that contains all vertices V. We denote the minimum cost of a solution to an MinPSC instance $I = (G, w)$ by $\mathrm{Opt}(I)$. Throughout this work, *weights* always refer to edges and *cost* refers to vertices or subgraphs. For proving computational hardness results, we will also consider the *decision version* of MinPSC, which we call k-PSC. Herein, given a natural number k, the problem is to decide whether a MinPSC instance $I = (G, w)$ satisfies $\mathrm{Opt}(I) \leq k$.

MinPSC falls into the category of survivable network design [37]. We refer to Clementi et al. [15] and Calinescu et al. [12] for a survey on different application scenarios and some results to related problems. Notably, Figure 1 reveals that computing a minimum-cost spanning tree typically does not yield an optimal solution for MinPSC (Erzin et al. [22] and de Graaf et al. [26] provide further discussions concerning the relationship to minimum-cost spanning trees). Indeed, MinPSC is NP-hard [16, 32]. In this work, we provide a refined computational complexity analysis of MinPSC in terms of parameterized complexity theory. In this way, we complement previous findings mostly concerning polynomial-time approximability [3, 16, 22, 26, 32], heuristics and integer linear programming [3, 21, 35, 39], and computational complexity analysis for special cases [13, 16, 22, 29]. We also complement recent positive results on provable effective data reduction for MinPSC [4] by negative ones.

Our contributions. Our work, which is summarized in Table 1, is driven by the question when small input-specific parameter values allow for fast (exact) solutions in practically relevant special cases. Our "use case scenarios"

herein are monitoring areas and restoring connectivity in a sensor network after sensor faults. In both scenarios, one naturally obtains lower bounds on the cost paid by each vertex: In the scenario of sensor faults, one might want to reconnect the sensor network without changing its topology too much, that is, so that each vertex pays at least the edges in an old solution. In the scenario of monitoring areas, we will notice that taking the cheapest edges incident to each vertex into a solution already yields a spanning subgraph with few connected components. The cost of this spanning subgraph coincides with the trivial lower bound

$$L := \sum_{v \in V} \min_{\{u,v\} \in E} w(\{u,v\})$$

and it only remains to connect its components to get a solution. When there is a solution whose cost coincides with the lower bound L, then it is easy to find: simply take for each vertex the incident edges of minimum weight. Naturally, the question arises whether we can efficiently find a solution if its cost is only little more than L?

In Section 3, we show that the answer to this question is "no". We show that $o(\log n)$-approximating the difference $d := \mathrm{Opt}(G,w) - L$ between the minimum solution cost and L in an n-vertex graph is NP-hard. Assuming the Exponential Time Hypothesis (ETH) [30], also we show that there is no exact $2^{o(n)}$-time algorithm for MinPSC. Going even further, we prove W[2]-hardness with respect to the parameter d, and thus, under ETH, there is no exact algorithm solving MinPSC in $f(d) \cdot n^{O(1)}$ time for any computable function f.

In Section 4, we provide an (exact) algorithm for MinPSC that exploits the above described lower bound L in a different way. More precisely, we present an algorithm that works in polynomial time if one already has a sensor network with $O(\log n)$ connected components or if one can find a set of *obligatory* edges that can be added to any optimal solution and yield a spanning subgraph with $O(\log n)$ connected components. In particular, this means that we show fixed-parameter tractability for MinPSC with respect to the parameter "number c of connected components in the spanning subgraph consisting of obligatory edges". Cases with small c occur, for example, in grid-like sensor arrangements, which minimize sensing area overlap when monitoring areas [48, 49] or when about 10 % of all sensors drop out of a triangular grid network (as we will see in Section 5). We also show negative results with respect to the parameter c: we show that the decision problem k-PSC is WK[1]-hard parameterized by c; problems that are WK[1]-hard parameterized by some parameter c are conjectured not to be reducible to solving instances of size $c^{O(1)}$ [28].

In Section 5, we conduct an experimental evaluation of our algorithm and compare it to CPLEX on state-of-the art ILP models for MinPSC, one of which also exploits the connected components of the spanning subgraph consisting of obligatory edges. We observe that our algorithms significantly outperform CPLEX with the known ILP models when n is sufficiently compared to the fixed c.

2 Preliminaries

2.1 Notation

We use $\mathbb{N}$ to denote the natural numbers including zero. By convention, the maximum of the empty set is $\max \emptyset = -\infty$ and the minimum of the empty set is $\min \emptyset = \infty$, since these are neutral elements with respect to taking the maximum and the minimum, respectively.

Graph theory. We consider undirected, finite, simple graphs $G = (V, E)$, where V is the set of *vertices* and $E \subseteq \{\{v,w\} \mid v \neq w \text{ and } v,w \in V\}$ is the set of *edges*. The *(open) neighborhood* $N_G(v) := \{u \in V \mid \{u,v\} \in E\}$ of a vertex $v \in V$ is the set of vertices *adjacent to* v in G. The *closed neighborhood* of v is $N_G[v] := N_G(v) \cup \{v\}$. For a vertex set $U \subseteq V$, $G - U$ is the graph obtained from G by deleting the vertices in U and their incident edges. A *clique* is a graph where each pair of vertices is adjacent.

2.2 Computational model and complexity

Computational model. We use the computational model of the *random access machine* (RAM) with unit costs, which acts on an array of rationals and carries out memory accesses, addition, subtraction, multiplication, and division in constant time [42, Section 4.2]. This assumption is justified since our algorithms will not operate on numbers significantly larger than the numbers given in the input.

Exponential time hypothesis. The Exponential Time Hypothesis (ETH) was introduced by Impagliazzo and Paturi [30] and it states that 3-SAT, the problem of deciding the satisfiability of a formula in 3-conjunctive normal form, cannot be solved in $2^{o(n+m)}$ time, where n and m are the number of variables and clauses in the input formula, respectively. ETH implies FPT $\neq$ W[1] [14].

Inapproximability. In order to transfer inapproximability results of some optimization problem Π to an optimization problem Π', we will use L-reductions [47]: An *L-reduction* with parameters α and β from a minimization problem Π to a minimization problem Π' is a pair of polynomial-time algorithms A_1 and A_2 such that

(i) A_1 turns any instance I of Π into an instance I' of Π' such that $\mathrm{Opt}(I') \leq \alpha\,\mathrm{Opt}(I)$, and

(ii) A_2 turns any solution of value ρ' for I' into a solution of value ρ for I such that $|\mathrm{Opt}(I)-\rho| \leq \beta|\mathrm{Opt}(I')-\rho'|$.

In particular, if there is an L-reduction with $\alpha = \beta = 1$ from Π to Π', then any γ-approximation for Π' transfers to a γ-approximation for Π. Thus, if Π is not γ-approximable in polynomial time, then neither is Π'.

2.3 Fixed-parameter tractability

Fixed-parameter algorithms. The essential idea behind fixed-parameter algorithms is to accept exponential running times, which are seemingly inevitable when solving NP-hard problems, but to restrict them to one aspect of the problem, the *parameter* [17, 20, 23, 36]. Thus, formally, an instance of a *parameterized problem* $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is a pair (x,k) consisting of the *input* x and the *parameter* k. A parameterized problem Π is *fixed-parameter tractable* with respect to a parameter k if there is an algorithm deciding $(x,k) \in \Pi$ in $f(k)\cdot n^{O(1)}$ time for some computable function f and $n = |x|$. Such an algorithm is called a *fixed-parameter algorithm*. It is potentially efficient for small values of k, in contrast to an algorithm that is merely running in polynomial time for each fixed k. FPT is the complexity class of fixed-parameter tractable parameterized problems. Observe that, for constant parameter values k, fixed-parameter tractability implies polynomial-time solvability, where the degree of the polynomial is *independent* of k. XP is the complexity class of parameterized problems solvable in polynomial time if the parameter is a constant, thus allowing for parameter dependencies in the degree of the running-time polynomial.

Parameterized intractability. The parameterized analog of P $\subseteq$ NP is a hierarchy of complexity classes FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \ldots \subseteq$ XP, where all inclusions are assumed to be proper. A parameterized problem Π with parameter k is *W[t]-hard* for some $t \in \mathbb{N}$ if every problem in W[t] has a parameterized reduction to Π: a *parameterized reduction* from a parameterized problem Π_1 to a parameterized problem Π_2 is an algorithm mapping an instance (x,k) to an instance (x',k') in $f(k)\cdot|x|^{O(1)}$ time such that $k' \leq g(k)$ and $(x,k) \in \Pi_1 \iff (x',k') \in \Pi_2$, where f and g are arbitrary computable functions. If g is a polynomial, then the parameterized reduction is called a *polynomial parameter transformation (PPT)*. By definition, no W[t]-hard problem is fixed-parameter tractable unless FPT = W[t].

2.4 (Hardness of) provably effective data reduction

Kernelization. Kernelization is the main formalization of data reduction with provable performance guarantees [24]. It has also proven effective in experimental studies [1, 7, 10, 34].

A *kernelization* for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial-time algorithm that maps any instance $(x,k) \in \Sigma^* \times \mathbb{N}$ to an instance $(x',k') \in \Sigma^* \times \mathbb{N}$ such that $(x,k) \in \Pi \iff (x',k') \in \Pi$, and $|x'|+k' \leq g(k)$ for some computable function g. We call (x',k') the *problem kernel* and g its *size*. A generalization of problem kernels are *Turing kernels*, where one is allowed to generate multiple reduced instances instead of a single one: A *Turing kernelization* for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that decides $(x,k) \in \Pi$ in polynomial time given access to an oracle that answers $(x',k') \in \Pi$ in constant time for any $(x',k') \in \Sigma^* \times \mathbb{N}$ with $|x'|+k' \leq g(k)$, where g is an arbitrary function called the *size* of the Turing kernel.

Kernelization hardness. *WK[1]-hard* parameterized problems with parameter k do not have problem kernels of size polynomial in k unless the polynomial-time hierarchy collapses and are conjectured not to have Turing kernels of polynomial size in k either [28]. Herein, a problem Π is *WK[1]-hard* if every problem in WK[1] has a polynomial parameter transformation to Π. An example for a WK[1]-hard problem is Set Cover parameterized by the size of the universe.

3 Parameterization above lower bound on total cost

In this section, we show that lower bounds on the total costs are hard to exploit algorithmically. To this end, we formalize the idea of lower bounds as follows.

Definition 3.1 (vertex lower bounds)**.** *Vertex lower bounds* are given by a function $\ell : V \to \mathbb{N}$ such that there is an optimal solution $T = (V,F)$ to MinPSC satisfying

$$\max_{\{u,v\} \in F} w(\{u,v\}) \geq \ell(v) \quad \text{for every vertex } v \in V.$$

Example 3.2. A trivial vertex lower bound $\ell(v)$ is given by the weight of the minimum-weight edge incident to v, because v has to be connected to some vertex in any solution. A more sophisticated vertex lower bound is given by

$$\ell(v) = \max_{G' \in C} \min_{u \in V(G')} w(\{u,v\}), \quad \text{where } C \text{ is the set of connected components of } G - \{v\}.$$

In the rest of this section, we concentrate on the trivial lower bound given by the minimum-weight edge. Thus, the overall cost of a solution is at least

$$L = \sum_{v \in V} \ell(v) = \sum_{v \in V} \min_{\{u,v\} \in E} w(\{u,v\}).$$

If the weights $w : E \to \mathbb{N}$ of the edges in a graph $G = (V,E)$ are at least one, then this immediately yields a "large" lower bound $L \geq n$ on the cost of an optimal solution. This implies that even constant-factor approximation algorithms (e. g., the one by Althaus et al. [3]) can return solutions that are, in absolute terms, quite far away from the optimum. Furthermore, fixed-parameter tractability for k-PSC parameterized by the solution cost immediately follows from Proposition 4.5 (in Section 4.2).

A more desirable and stronger result would be about the difference d between the optimal solution cost and L: for example a polynomial-time constant-factor approximation of d or a fixed-parameter tractability result with respect to the parameter d [8, 18, 25, 33]. However, under the Exponential Time Hypothesis, we show that such algorithms do not exist. To formally state our hardness results, we use the following problem variant, which incorporates the lower bound.

Problem 3.3 (MinPSC Above Lower Bound (MinPSC-ALB))**.**
Input: A connected undirected graph $G = (V,E)$ and edge weights $w : E \to \mathbb{N}$.
Goal: Find a connected spanning subgraph $T = (V,F)$ of G that minimizes

$$\sum_{v \in V} \max_{\{u,v\} \in F} w(\{u,v\}) - \sum_{v \in V} \min_{\{u,v\} \in E} w(\{u,v\}). \tag{1}$$

For a MinPSC-ALB instance $I = (G, w)$, we denote by $\mathrm{Opt}_{\mathrm{mar}}(I)$ the minimum value of (1) (we also refer to $\mathrm{Opt}_{\mathrm{mar}}(I)$ as the *margin* of I). For showing hardness results, we will also consider the *decision version* of the problem: By d-PSC-ALB, we denote the problem of deciding whether an MinPSC-ALB instance $I = (G, w)$ satisfies $\mathrm{Opt}_{\mathrm{mar}}(I) \leq d$.

Theorem 3.4. The following properties hold even in graphs with edge weights one and two:

(i) MinPSC-ALB is NP-hard to approximate within a factor of $o(\log n)$,

(ii) d-PSC-ALB is W[2]-hard when parameterized by d, and

(iii) unless ETH fails, k-PSC and d-PSC-ALB are not solvable in $2^{o(n)}$ time.

Moreover, d-PSC-ALB is solvable in in $O(2^d \cdot n^{d+2})$ time and (i) and (ii) also hold in complete graphs with metric edge weights.

Theorem 3.4(ii) implies that, under ETH, d-PSC-ALB is not solvable in $f(d) \cdot n^{O(1)}$ time [14]. In contrast, Theorem 3.4 also shows that it is solvable in polynomial time for any fixed d. We prove Theorem 3.4 using a modified reduction from Minimum Set Cover to MinPSC due to Erzin et al. [22].

Problem 3.5 (Minimum Set Cover)**.**
Input: A universe U and a family $\mathscr{F}$ of subsets of U.
Goal: Find a *set cover* $\mathscr{F}' \subseteq \mathscr{F}$ (that is, $\bigcup_{S \in \mathscr{F}'} S = U$) of minimum size.

We now present the reduction of Erzin et al. [22] and then show that a modification yields Theorem 3.4. Note that Erzin et al. used edge weights zero and one in their reduction, whereas we use positive integers, namely one and two.

Transformation 1. Given an instance $(U, \mathscr{F})$ of Minimum Set Cover, construct an instance (G, w) of MinPSC-ALB as follows. The vertex set of the graph $G = (V, E)$ is $V := \{s\} \uplus U \uplus \mathscr{F}$ for some new vertex s. There is an edge $\{u, S\}$ of weight two in G for each $u \in U$ and $S \in \mathscr{F}$ such that $u \in S$. Moreover, there is an edge $\{s, S\}$ of weight one for each $S \in \mathscr{F}$.

Transformation 1 creates graphs with edge weights one and two. The following transformation completes it to satisfy the triangle inequality.

Transformation 2. Given an instance $(U, \mathscr{F})$ of Minimum Set Cover, first apply Transformation 1 and, in the resulting instance (G, w) of MinPSC, turn G into a complete graph, assigning all newly added edges $\{u, v\}$ a weight equal to the length of a shortest weighted u-v-path.

Transformation 2 introduces edges of weight three and four. To prove the correctness of the reduction, we first prove that it is never required for these edges to be used in a solution.

Lemma 3.6. Let $(U, \mathscr{F})$ be an instance of Set Cover and (G, w) be an MinPSC instance generated from it by Transformation 1 or Transformation 2, where $G = (V, E)$ and $V = \{s\} \uplus U \uplus \mathscr{F}$. Finally, for a solution $T = (V, F)$ to (G, w), let

$$p_T(v) := \max_{\{u,v\} \in F} w(\{u, v\})$$

be the cost of vertex $v \in V$ paid in T. Then, the cost of T is at least $1 + |\mathscr{F}| + 2|U|$ and, in polynomial time without increasing its cost, one can transform T so that

(i) $p_T(s) = 1$,

(ii) $p_T(S) \in \{1, 2\}$ for all $S \in \mathscr{F}$,

(iii) $p_T(u) = 2$ for all $u \in U$.

Proof. Take an arbitrary solution $T = (V, F)$ for (G, w). Obviously, $p_T(s) \geq 1$, $p_T(u) \geq 2$ for any $u \in U$, and $p_T(S) \geq 1$ for any $S \in \mathscr{F}$, yielding the lower bound of $1 + |\mathscr{F}| + 2|U|$ on the cost of T.

Since the rest of the lemma obviously holds for Transformation 1, we prove it for Transformation 2. To this end, we partition $\mathscr{F} = \mathscr{F}_1 \uplus \mathscr{F}_2$ so that $p_T(S) = 1$ for all $S \in \mathscr{F}_1$ and $p_T(S) \geq 2$ for all $S \in \mathscr{F}_2$. Moreover, consider $U_2 = \bigcup_{S \in \mathscr{F}_2} S$ and $U_1 = U \setminus U_2$. Observe that $p_T(u) \geq 3$ for each $u \in U_1$, since there are no edges of weight one incident to u and each of u's neighbors along an edge of weight two pays only cost one. Thus, the cost of T is

$$\begin{aligned}\sum_{v \in V} p_T(v) &\geq |\{s\}| + |\mathscr{F}_1| + 2|\mathscr{F}_2| + 2|U_2| + 3|U_1| \\ &= 1 + |\mathscr{F}| + 2|U| + |\mathscr{F}_2| + |U_1|. \end{aligned} \tag{2}$$

We now describe a solution T' satisfying Lemma 3.6(i)–(iii) that can be computed in polynomial time from T and whose cost is exactly (2).

- T' contains all edges of weight one incident to s. These connect all vertices $S \in \mathscr{F}$ to s. Note that $1 = p_{T'}(s) \leq p_T(s)$.
- T' contains all edges of weight two incident to some $S \in \mathscr{F}_2$. These connect all vertices $u \in U_2$ to $\mathscr{F}_2$ and, hence, to s. Note that $2 = p_{T'}(S) \leq p_T(S)$ for $S \in \mathscr{F}_2$. Since T' will not contain any other edges incident to U_2, $2 = p_{T'}(u) \leq p_T(u)$ for any $u \in U_2$.
- It remains to connect each vertex $u \in U_1$ to the rest of the graph. To this end, T' contains an arbitrary edge $\{u, S\}$ such that $u \in S$, which has weight two. Note that $2 = p_{T'}(u) < 3 \leq p_T(u)$ for any $u \in U_1$.

Let $K \subseteq V$ be the set of vertices u such that $p_{T'}(u) > p_T(u)$. That is, actually, $K \subseteq \mathscr{F}_1$ and $|K| \leq |U_1|$, since $K \subseteq \mathscr{F}_1$ consists exactly of the end points of the edges in T' we added for each $u \in U_1$. Moreover, $p_{T'}(S) = 2$ for each $S \in K$. Thus, the cost of this solution T' is

$$\sum_{v \in V} p_{T'}(v) = |\{s\}| + |\mathscr{F}_1 \setminus K| + 2|K| + 2|\mathscr{F}_2| + 2|U| \leq 1 + |\mathscr{F}_1| + |U_1| + 2|\mathscr{F}_2| + 2|U| = (2). \qquad \square$$

The following lemma, together with the approximation hardness of Minimum Set Cover, will yield Theorem 3.4(i).

Lemma 3.7. Transformation 1 yields an L-reduction with parameters $\alpha = \beta = 1$ from Minimum Set Cover to MinPSC-ALB in graphs with edge weights one and two, whereas Transformation 2 yields such an L-reduction to MinPSC-ALB in complete graphs with metric edge weights.

Proof. If is obvious that Transformations 1 and 2 create graphs with edge weights one and two and metric complete graphs, respectively. We verify that they satisfy properties (i) and (ii) of L-reductions (cf. Section 2.2).

(i) Both transformations work in polynomial time. It remains to show that the margin of the optimal solution for the instance $I' = (G, w)$ of MinPSC-ALB is at most the cost of the optimal solution for the instance $I = (U, \mathscr{F})$ of Minimum Set Cover. To this end, let $\mathscr{F}' \subseteq \mathscr{F}$ be a set cover of minimum size and thus $\mathrm{Opt}(I) = |\mathscr{F}'|$. Consider the spanning subgraph T of $G = (\{s\} \uplus U \uplus \mathscr{F}, E)$ that contains all edges of weight one incident to s and all edges $\{u, S\}$ (of weight two) such that $S \in \mathscr{F}'$ and $u \in S$. Then T is a connected spanning subgraph: All vertices in $\mathscr{F}$ are connected via s by the edges with weight one. Furthermore, each vertex in U is connected to at least one vertex in $\mathscr{F}'$ since $\mathscr{F}'$ is a set cover. We analyze its margin to show $\mathrm{Opt}_{\mathrm{mar}}(I') \leq \mathrm{Opt}(I)$.

By Lemma 3.6, the lower bound of the cost of any connected spanning subgraph in G is $2|U| + |\mathscr{F}| + 1$. Yet the overall cost of T is at most $2|U| + |\mathscr{F}| + 1 + \mathrm{Opt}(I)$ and thus the margin ρ' of T is $\mathrm{Opt}(I)$: Compared to the lower bound $2|U| + |\mathscr{F}| + 1$, each vertex S with $S \in \mathscr{F}'$ pays one additionally (in total two) since it is incident to a weight-two edge in T that has the other endpoint in U. Thus, the overall cost is $2|U| + |\mathscr{F}| + 1 + \mathrm{Opt}(I)$ and $\mathrm{Opt}_{\mathrm{mar}}(I') \leq \mathrm{Opt}(I)$.

(ii) We show how to transform solutions for the instance $I' = (G, w)$ of MinPSC-ALB into set covers for $I = (U, \mathscr{F})$. To this end, let T be a connected spanning subgraph of $G = (\{s\} \uplus U \uplus \mathscr{F}, E)$. Without loss of generality,

T is of the form described by Lemma 3.6: s pays one, each vertex $u \in U$ pays two, and each vertex $S \in \mathscr{F}$ pays one or two. We show that the subset $\mathscr{F}' \subseteq \mathscr{F}$ of vertices paying two is a set cover for I. To this end, consider any element u in U. Since T is connected, it follows that at least one edge e is incident to u is in T. Since $w(e) = 2$, by construction, the second endpoint of e is in $\mathscr{F}$ and thus, $e = \{u, S\}$ for some $S \in \mathscr{F}$ with $u \in S$. By definition of $\mathscr{F}'$, we have $S \in \mathscr{F}'$. Thus, u is covered by $\mathscr{F}'$.

Observe that the margin of T is $\rho = |\mathscr{F}'|$, which is equal to the size ρ' of the set cover $\mathscr{F}'$. Since this argument holds for any connected spanning subgraph of G, we have $\mathrm{Opt}(I) \leq \mathrm{Opt}_{\mathrm{mar}}(I')$. Since we already showed $\mathrm{Opt}_{\mathrm{mar}}(I') \leq \mathrm{Opt}(I)$ it follows that $\mathrm{Opt}_{\mathrm{mar}}(I') = \mathrm{Opt}(I)$. Hence, we arrive at $|\mathrm{Opt}(I) - \rho| = |\mathrm{Opt}_{\mathrm{mar}}(I') - \rho'|$. □

The following lemma, together with the W[2]-hardness of k-Set Cover, will yield Theorem 3.4(ii).

Lemma 3.8. Transformations 1 and 2 yield parameterized reductions from k-Set Cover parameterized by the solution size k to d-MinPSC-ALB parameterized by d in graphs with edge weights one and two or in complete graphs with triangle inequality, respectively. The graph constructed by Transformation 1 has $O(|U| + |\mathscr{F}|)$ vertices.

Proof. We slightly enhance Transformations 1 and 2 using the decision versions of the problems and setting $d := k$. Note that, in the proof of Lemma 3.7, we showed that there is a set cover of size k in the given Minimum Set Cover instance if and only if there is a connected spanning subgraph T with cost $2|U| + |\mathscr{F}| + 1 + d$ in the constructed d-MinPSC-ALB-instance. Finally, observe that Transformation 1 creates a graph where the number of vertices is $O(|U| + |\mathscr{F}|)$. □

Combining the known intractability of k-Set Cover with Lemmas 3.7 and 3.8, we can finally prove Theorem 3.4.

Proof of Theorem 3.4. (i) follows from Lemma 3.7 and the fact that Minimum Set Cover is NP-hard to approximate within a factor of $o(\log n)$ [40].

(ii) follows from Lemma 3.8 and the fact that k-Set Cover is W[2]-complete parameterized by the solution size k [20].

(iii) follows from Lemma 3.8, the observation that Transformation 1 runs in polynomial time, and the fact that k-Set Cover cannot be solved in $2^{o(|U|+|\mathscr{F}|)}$ time unless the ETH fails [31].

Finally, MinPSC can be solved in in $O(2^d \cdot n^{d+2})$ time as follows: At most d vertices can pay more than their vertex lower bound. We can try all possibilities for choosing $i \leq d$ vertices, all $\binom{d}{i}$ possibilities to increase their total cost by at most d, and check whether the graph of the "paid" edges is connected. The algorithm runs in $\sum_{i=1}^{d} \binom{n}{i}\binom{d}{i} \cdot O(n+m) \subseteq O(2^d \cdot n^{d+2})$ time. □

4 Parameterizing by the number of connected components induced by obligatory edges

Complementing our hardness results in Section 3, we show in this section how vertex lower bounds (see Definition 3.1) can be algorithmically exploited. To this end, in Section 4.1, we describe how vertex lower bounds induce, what we call, an *obligatory subgraph*, which is part of at least one optimal solution.

In Section 4.2, we present the algorithm that solves MinPSC efficiently if the obligatory subgraph has few connected components. In Sections 4.3 to 4.5, we prove its correctness and running time.

Finally, in Section 4.6, we prove that it is hard to reduce arbitrary instances of MinPSC to equivalent instances with a size polynomial in the number of the connected components of the obligatory subgraph.

Notably, the obligatory subgraph does not necessarily have to be given by vertex lower bounds. It can also arise as part of an application scenario and given as input, for example, when reconnecting a sensor network that has lost connectivity due to sensor faults, as studied by Rodoplu and Meng [41].

4.1 Finding obligatory edges

To find obligatory edges, we use vertex lower bounds (see Definition 3.1). Once we have vertex lower bounds, we can compute an *obligatory subgraph*, whose edges are contained in at least one optimal solution.

Definition 4.1 (obligatory subgraph)**.** The *obligatory subgraph* G_ℓ of a graph $G = (V, E)$ induced by vertex lower bounds $\ell : V \to \mathbb{N}$ consists of all vertices of G and all *obligatory edges* $\{u, v\}$ with

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

The better the vertex lower bounds ℓ are, the more obligatory edges they potentially induce, thus reducing the number c of connected components of G_ℓ. Clearly, coming up with good vertex lower bounds is a challenge on its own. Yet already the simple vertex lower bounds in Example 3.2 may yield obligatory subgraphs with few connected components in applications:

Example 4.2. Consider the vertex lower bounds ℓ from Example 3.2. If we arrange sensors in a grid, which is the most energy-efficient arrangement of sensors for monitoring areas [48, 49], then G_ℓ has only one connected component. The number of connected components may increase due to sensor defects that disconnect the grid or due to varying sensor distances within the grid. The worst case is if the sensors have pairwise distinct distances. Then, G_ℓ might have only one edge, joining the closest pair, and $n-1$ connected components.

Alternatively, an obligatory subgraph may arise as the result of a sensor network that lost connectivity due to faulty sensors and has to be reconnected at minimum extra cost.

4.2 A fixed-parameter algorithm

The number c of connected components in G_ℓ can easily be exploited in an exact $O(n^{2c})$-time algorithm for MinPSC,[1] which runs in polynomial time for constant c, yet is inefficient already for small values of c. We develop, among others, a randomized algorithm that runs in polynomial time even for $c \in O(\log n)$:

Theorem 4.3. MinPSC with vertex lower bounds ℓ is solvable

(i) in $O(\ln 1/\varepsilon \cdot (4e^2/\sqrt{2\pi})^c \cdot (9^c m + 4^c nm + nm \log n))$ time by a randomized algorithm with an error probability at most ε for any given ε with $0 < \varepsilon < 1$, and

(ii) in $c^{O(c \log c)} \cdot n^{O(1)}$ time by a deterministic algorithm,

where c is the number of connected components of the obligatory subgraph G_ℓ.

Remark 4.4. The algorithms behind Theorem 4.3 work for *any* vertex lower bounds ℓ. Thus, any (heuristical) approach improving on the vertex lower bounds will directly (and provably) improve the performance of the algorithms.

The deterministic algorithm in Theorem 4.3(ii) is primarily of theoretical interest because it classifies MinPSC as *fixed-parameter tractable* parameterized by c. Practically, the randomized algorithm in Theorem 4.3(i) is much easier to implement and it is the one that we experimentally evaluate in Section 5.

Note that the parameter c is upper-bounded by n. Thus, Theorem 3.4(iii) also implies that, under ETH, there is no $2^{o(c)}(n+m)^{O(1)}$-time algorithm for MinPSC, whereas our randomized algorithm in Theorem 4.3(i) has running time $2^{O(c)}n^{O(1)}$ for constant error probability ε.

The number of connected components of obligatory subgraphs has recently also been exploited in fixed-parameter algorithms for problems of servicing links in transportation networks [7, 27, 45, 46], which led to practical results [7, 9].

[1] To connect the c components of G_ℓ, one has to add $c-1$ edges. These have at most $2c-2$ endpoints. One can try all n^{2c-2} possibilities for choosing these endpoints and check each resulting graph for connectivity in $O(n+m) \subseteq O(n^2)$ time. Altogether, this proves containment of MinPSC parameterized by c in the class XP.

We will now prove Theorem 4.3. The proof also yields the following deterministic algorithm for MinPSC. It is much faster than the trivial algorithm enumerating all of the possibly n^{n-2} spanning trees. Moreover, Theorem 3.4(iii) shows that it is asymptotically optimal:

Proposition 4.5. MinPSC can be solved in $O(3^n \cdot m)$ time.

Like some known approximation algorithms for MinPSC [3, 29], our algorithms in Theorem 4.3 work by adding edges to G_ℓ in order to connect its c connected components. In contrast to the known approximation algorithms, our algorithms will find an *optimal* set of edges to add. We describe our algorithm in terms of a *padded* version $G_\ell^\bullet$ of the input graph G, in which each connected component of G_ℓ is turned into a clique. In the padded graph, it is sufficient to search for connected subgraphs of $G_\ell^\bullet$ that contain at least one vertex of each connected component of G_ℓ: We can then add the edges in G_ℓ to such subgraphs in order to obtain a connected spanning subgraph of G. This simplifies the problem.

Definition 4.6 (padded graph, components)**.** Let $\ell\colon V \to \mathbb{N}$ be vertex lower bounds for a graph $G = (V, E)$ with edge weights $w\colon E \to \mathbb{N}$. We denote the c connected components of the obligatory subgraph G_ℓ by $G_\ell^1, G_\ell^2, \ldots, G_\ell^c$.

The *padded graph* $G_\ell^\bullet = (V, E_\ell^\bullet)$ is a supergraph of G with the edge set $E_\ell^\bullet = E \cup E_\ell^\circ$, where

$$E_\ell^\circ := \{\{u, v\} \subseteq V \mid u \text{ and } v \text{ are in the same component of } G_\ell\},$$

and edge weights

$$w_\ell^\bullet\colon E_\ell^\bullet \to \mathbb{N}, \{u, v\} \mapsto \begin{cases} 0 & \text{if } \{u, v\} \in E_\ell^\circ \text{ and} \\ w(\{u, v\}) & \text{otherwise.} \end{cases}$$

Algorithm outline. To solve a MinPSC instance (G, w) with vertex lower bounds $\ell\colon V \to \mathbb{N}$, we have to add $c-1$ edges to G_ℓ in order to connect its c connected components. These edges have at most $2c-2$ endpoints. Thus, we need to find a connected subgraph in $G_\ell^\bullet$ that

- contains at most $2c-2$ vertices,
- contains at least one vertex of each connected component of G_ℓ, and
- minimizes the total cost increase compared to the vertex lower bounds $\ell\colon V \to \mathbb{N}$.

We will do this using the color-coding technique introduced by Alon et al. [2]: randomly color the vertices of $G_\ell^\bullet$ using at most $2c-2$ colors and then search for connected subgraphs of $G_\ell^\bullet$ that each contain exactly one vertex of each color and in which the total cost increase compared to the vertex lower bounds $\ell\colon V \to \mathbb{N}$ is minimized. Formally, we will solve the following problem on $G_\ell^\bullet$.

Problem 4.7 (Min-Power Increment Colorful Connected Subgraph (MinPICCS))**.**
Input: A connected undirected graph $G = (V, E)$, edge weights $w\colon E \to \mathbb{N}$, vertex colors $\mathrm{col}\colon V \to \mathbb{N}$, a function $\ell\colon V \to \mathbb{N}$, and a color subset $C \subseteq \mathbb{N}$.
Goal: Compute a connected subgraph $T = (W, F)$ of G such that col is a bijection between W and C and such that T minimizes

$$\sum_{v \in W} \max\Big\{0, \max_{\{u,v\} \in F} w(\{u, v\}) - \ell(v)\Big\}. \tag{3}$$

Note that, in order to connect the components of G_ℓ using MinPICCS, we cannot simply color the vertices of the input graph G *completely* randomly: One component of G_ℓ could contain all colors and, thus, a connected subgraph containing all colors does not necessarily connect the components of G_ℓ. Instead, we employ a trick that was previously applied mainly heuristically in algorithm engineering in order to increase the success probability of color-coding algorithms [6, 11, 19]: Since we know that our sought subgraph contains at least one vertex of each connected component of G_ℓ, we color the connected components of G_ℓ using pairwise disjoint color sets. Herein, we first "guess" the number z_i of vertices that the sought subgraph will contain of each connected component G_ℓ^i of G_ℓ and use z_i colors to color each component G_ℓ^i. We thus arrive at the following algorithm for MinPSC:

1. construct $G_\ell^\bullet$ (in which each connected component of G_ℓ is a clique)
2. Repeat a certain number of times:
   (a) color the vertices randomly so that the connected components of G_ℓ get pairwise disjoint colors.
   (b) solve MinPICCS on the colored graph.

It is formalized as follows:

Algorithm 1 (for MinPSC).
Input: A MinPSC instance $I = (G, w)$, vertex lower bounds $\ell : V \to \mathbb{N}$ for $G = (V, E)$, and an upper bound ε on the error probability, where $0 < \varepsilon < 1$.
Output: With probability at least $1 - \varepsilon$ an optimal solution for I.

1. $c \leftarrow$ number of connected components of the obligatory subgraph G_ℓ.
2. **for each** $z_1, z_2, \ldots, z_c \in \mathbb{N} \setminus \{0\}$ such that $\sum_{i=1}^{c} z_i \leq 2c-2$ **do**:
3. choose pairwise disjoint $C_i \subseteq \{1, \ldots, 2c-2\}$ with $|C_i| = z_i$ for $i \in \{1, \ldots, c\}$.
4. **repeat** $t := \lceil \ln \varepsilon / \ln(1 - \prod_{i=1}^{c} z_i!/z_i^{z_i}) \rceil$ **times**:[2]
5. **for each** $i \in \{1, \ldots, c\}$, randomly color the vertices of component G_ℓ^i of G_ℓ using colors from C_i, let the resulting coloring be $\mathrm{col} : V \to \mathbb{N}$.
6. Solve MinPICCS instance $I^\bullet := (G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$ as described in Section 4.4.
7. let $T = (W, F)$ be the best MinPICCS solution found in any of the repetitions.
8. **return** $T' = (V, (F \setminus E_\ell^\circ) \cup E_\ell)$.

We prove the correctness of Algorithm 1 in Section 4.3. Then, in Section 4.4, we show how to solve the MinPICCS instance in line 6 of Algorithm 1. Finally, in Section 4.5, we analyze the running time of Algorithm 1 and also show how to derandomize it to complete the proof of Theorem 4.3.

4.3 Correctness of Algorithm 1

We will now prove the correctness of Algorithm 1. First, with the following lemma, we prove that, if Algorithm 1 chooses a suitable coloring in line 5, then the MinPICCS instance $I^\bullet$ solved in line 6 has a solution of cost at most $\mathrm{Opt}(I) - \sum_{v \in V} \ell(v)$.

Lemma 4.8. Let $I = (G, w)$ be a MinPSC instance, let $\ell : V \to \mathbb{N}$ be vertex lower bounds for $G = (V, E)$, and let c be the number of connected components of $G_\ell = (V, E_\ell)$. Then, there is an optimal solution $T = (V, F)$ for I such that

(i) the set $W \subseteq V$ of vertices incident to an edge in $F \setminus E_\ell$ has at most $2c-2$ vertices, and

(ii) for $C = \{1, \ldots, |W|\}$ and any coloring $\mathrm{col} : V \to C$ inducing a bijection between W and C, there is a solution $T' = (W, F')$ to the MinPICCS instance $(G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$ with cost at most

$$\mathrm{Opt}(I) - \sum_{v \in V} \ell(v).$$

Proof. (i) Let $T = (V, F)$ be an optimal solution for (G, w) that contains all edges of $G_\ell = (V, E_\ell)$ and a minimum number of edges of $E \setminus E_\ell$. In order to connect the c connected components of G_ℓ, the graph T contains $c-1$ edges in $E \setminus E_\ell$. These can have at most $2c-2$ endpoints. Thus, $|W| \leq 2c-2$.

(ii) Consider the graph $T' = (W, F')$ with the edge set

$$F' := \{\{u, v\} \subseteq W \mid \{u, v\} \in F\} \cup \{\{u, v\} \subseteq W \mid \{u, v\} \in E_\ell^\bullet \text{ and } w_\ell^\bullet(\{u, v\}) = 0\}. \tag{4}$$

[2] Repeat once if the logarithm is undefined, that is, if $z_i = 1$ for all $i \in \{1, \ldots, c\}$.

We show that T' is a solution to the MinPICCS instance $I^\bullet = (G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$. We first analyze its cost. By Definition 3.1, $\ell(v) \leq \max_{\{u,v\}\in E} w(\{u,v\})$ for all $v \in V$. By Definition 4.6, $w_\ell^\bullet(\{u,v\}) \leq w(\{u,v\})$ if $\{u,v\} \in F$ and $w_\ell^\bullet(\{u,v\}) = 0$ if $\{u,v\} \in F' \setminus F$. Thus, the cost of T' as a solution to $I^\bullet$ is

$$\begin{aligned}
&\sum_{v\in W} \max\Big\{0, \max_{\{u,v\}\in F'} w_\ell^\bullet(\{u,v\}) - \ell(v)\Big\} \leq \sum_{v\in V} \max\Big\{0, \max_{\{u,v\}\in F'} w_\ell^\bullet(\{u,v\}) - \ell(v)\Big\} \\
&\leq \sum_{v\in V} \max\Big\{0, \max_{\{u,v\}\in F} w(\{u,v\}) - \ell(v)\Big\} \leq \sum_{v\in V} \Big(\max_{\{u,v\}\in F} w(\{u,v\}) - \ell(v)\Big) = \mathrm{Opt}(I) - \sum_{v\in V} \ell(v).
\end{aligned}$$

By assumption, col is a bijection between W and C. Thus, in order to show that T' is a solution to $I^\bullet$, it remains to show that it is connected. Towards a contradiction, assume that $T' = (W, F')$ is not connected. Choose $u, v \in W$ that are disconnected in T' and have minimum distance in T, as measured as the number of edges on a shortest u-v-path p in T. By (4), all edges of $T = (V, F)$ between vertices in W are also in T'. Thus, p has no inner vertices in W. By choice of W, it follows that all edges of p are in E_ℓ and, consequently, u and v are in the same connected component of G_ℓ. By Definition 4.6, $w_\ell^\bullet(\{u,v\}) = 0$. By (4), we get $\{u,v\} \in F'$, contradicting our assumption. □

The next lemma is the converse to Lemma 4.8: we show that Algorithm 1 in line 8 recovers a solution to I of cost at least $\mathrm{Opt}(I^\bullet) + \sum_{v\in V} \ell(v)$ from the MinPICCS instance $I^\bullet$ solved in line 6.

Lemma 4.9. Let $I := (G = (V, E), w)$ be a MinPSC instance, let $\ell\colon V \to \mathbb{N}$ be vertex lower bounds, and let $\mathrm{col}\colon V \to C$ be a coloring such that, for $i \neq j$, the sets of colors of vertices in the connected components G_ℓ^i and G_ℓ^j of $G_\ell = (V, E_\ell)$ are disjoint.

If $T = (W, F)$ is an optimal solution to the MinPICCS instance $I^\bullet = (G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$, then $T' = (V, (F \setminus E_\ell^\circ) \cup E_\ell)$ is a solution for (G, w) of cost at most

$$\mathrm{Opt}(I^\bullet) + \sum_{v\in V} \ell(v).$$

Proof. As required by the definition of MinPICCS (Problem 4.7), T is connected and contains exactly one vertex of each color in C. Since the color sets of distinct connected components of $G_\ell = (V, E_\ell)$ are disjoint, T contains at least one vertex of each connected component of G_ℓ. By construction, T' contains all edges of G_ℓ and all edges of T between different connected components of G_ℓ. Thus, T' is connected.

It remains to analyze the cost of T' as a solution to the MinPSC instance (G, w). To this end, let $F' = (F \setminus E_\ell^\circ) \cup E_\ell$ be the edge set of T' and observe that, by Definitions 4.1 and 4.6,

- for all edges $\{u,v\} \in F' \cap E_\ell \supseteq F' \setminus F$, one has $w_\ell^\bullet(\{u,v\}) = 0 \leq w(\{u,v\}) \leq \min\{\ell(u), \ell(v)\}$,
- for all edges $\{u,v\} \in F' \setminus E_\ell \subseteq E \setminus E_\ell^\circ$, one has $w_\ell^\bullet(\{u,v\}) = w(\{u,v\})$,

and that no vertex $v \in V \setminus W$ is incident to edges in F. Thus, the cost of T' as solution to the MinPSC instance (G, w) is

$$\begin{aligned}
&\sum_{v\in V} \max_{\{u,v\}\in F'} w(\{u,v\}) \leq \sum_{v\in V} \max\Big\{\ell(v), \max_{\{u,v\}\in F'} w(\{u,v\})\Big\} = \sum_{v\in V} \max\Big\{\ell(v), \max_{\{u,v\}\in F'} w_\ell^\bullet(\{u,v\})\Big\} \\
&\leq \sum_{v\in V} \max\Big\{\ell(v), \max_{\{u,v\}\in F} w_\ell^\bullet(\{u,v\})\Big\} = \sum_{v\in V} \max\Big\{0, \max_{\{u,v\}\in F} w_\ell^\bullet(\{u,v\}) - \ell(v)\Big\} + \sum_{v\in V} \ell(v) \\
&= \sum_{v\in W} \max\Big\{0, \max_{\{u,v\}\in F} w_\ell^\bullet(\{u,v\}) - \ell(v)\Big\} + \sum_{v\in V} \ell(v) = \mathrm{Opt}(I^\bullet) + \sum_{v\in V} \ell(v).
\end{aligned}$$

□

In Lemmas 4.8 and 4.9, we have shown how to translate between optimal solutions of the input MinPSC instance I and the optimal solutions of the MinPICCS instance $I^\bullet$ solved in line 6, given a suitable vertex coloring. To prove the correctness of Algorithm 1, it remains to combine these lemmas and analyze the probability of a suitable coloring in order to show that Algorithm 1 returns an optimal solution with probability at least $1 - \varepsilon$.

Proposition 4.10. Algorithm 1 is correct.

Proof. By Lemma 4.8, there is an optimal solution $T = (V, F)$ to the MinPSC instance $I = (G, w)$ with vertex lower bounds $\ell : V \to \mathbb{N}$ such that the set $W \subseteq V$ of vertices incident to an edge in T that is not in G_ℓ satisfies $|W| \leq 2c-2$, where c is the number of connected components of G_ℓ. Thus, in at least one iteration of the loop in line 2, we will have that z_i is the number of vertices of the connected component G_ℓ^i that are contained in W. If any of the colorings in line 5 colors the vertices of W in $|W|$ pairwise distinct colors, then, by Lemma 4.8, the MinPICCS instance $I^\bullet$ solved in line 6, and therefore the solution selected in line 7, has cost at most $\mathrm{Opt}(I^\bullet) \leq \mathrm{Opt}(I) - \sum_{v \in V} \ell(v)$. By Lemma 4.9, the MinPSC solution returned in line 8 then has cost at most $\mathrm{Opt}(I^\bullet) + \sum_{v \in V} \ell(v) \leq \mathrm{Opt}(I)$, implying that it is optimal. It remains to analyze the probability of a suitable coloring.

Since line 3 chooses pairwise disjoint color sets for the components G_ℓ^i, line 5 colors the vertices of W in $|W|$ pairwise distinct colors if and only if, for each $i \in \{1, \dots, c\}$, the z_i vertices of W in component G_ℓ^i are colored in z_i pairwise distinct colors. Call this event A_i for $i \in \{1, \dots, c\}$. There are $z_i^{z_i}$ possibilities to color the z_i vertices of W in component G_ℓ^i. Out of these, there are $z_i!$ possibilities to color them in pairwise distinct colors. Thus, $\Pr[A_i] = z_i!/z_i^{z_i}$ for each component G_ℓ^i. Since A_i and A_j are independent, the probability that all vertices of W get pairwise distinct colors is

$$p := \Pr\Big[\bigcap_{i=1}^{c} A_i\Big] = \prod_{i=1}^{c} \Pr[A_i] = \prod_{i=1}^{c} \frac{z_i!}{z_i^{z_i}}.$$

If $p = 1$, then the only repetition of the loop in line 4 yields a suitable coloring. If $p < 1$, then the probability that none of its t iterations yields a suitable coloring is $(1-p)^t = (1-p)^{\ln \varepsilon / \ln(1-p)} = (1-p)^{\log_{1-p} \varepsilon} = \varepsilon$. □

Having shown that Algorithm 1 is correct, to prove Theorem 4.3, it remains to analyze the running time of Algorithm 1 and to derandomize it. To analyze its running time, we now show how to efficiently solve the MinPICCS instances in line 6 of Algorithm 1.

4.4 Solving MinPICCS in line 6 of Algorithm 1

In the previous section, we have shown that Algorithm 1 is correct. To carry it out efficiently, we show in this section how to solve the MinPICCS instances in line 6:

Proposition 4.11. The MinPICCS instance in line 6 of Algorithm 1 can be solved in time

$$O(3^{|C|}m + 2^{|C|}nm + nm \log n).$$

To prove Proposition 4.11, we use a dynamic programming algorithm inspired by an algorithm used for finding signalling pathways in biological networks [43]: it finds trees containing one vertex of each color in a vertex-colored graph. Our case is complicated by the non-standard goal function (3) of MinPICCS and that we do not want to compute the graph $G_\ell^\bullet$ explicitly: its size might be quadratic in that of G, which leads to prohibitively large running times when working on $G_\ell^\bullet$.[3] We will thus have to compute the optimal solution for $I^\bullet := (G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$ by looking at the input graph G only.

In the following, we will use some simplifying assumptions and conventions, which clearly do not influence the optimal solutions to MinPSC and MinPICCS.

Assumption 4.12. For each vertex $v \in V$, there is a loop $\{v\} \in E$ of weight $w(\{v\}) = 0$. Consequently, by Definition 4.6, also $\{v\} \in E_\ell^\bullet$ and $w_\ell^\bullet(\{v\}) = 0$. For any $e \notin E_\ell^\bullet$, we assume $w_\ell^\bullet(e) = \infty$.

To use dynamic programming, we formally define the subproblems that we are going to solve using a recurrence relation.

[3] Indeed, in the conference version of this article [5], we worked directly on $G_\ell^\bullet$, which led to a running time of $O(3^{|C|}n^4)$ for carrying out line 6. In experiments this turned out to be inferior to CPLEX or even brute force.

Definition 4.13 ($P(v,q,C')$, $\Phi(v,q,T)$, $D[v,q,C']$)**.** For any color set $C' \subseteq C$ and any edge $\{v,q\} \in E_\ell^\bullet$ (possibly, $v = q$), we denote by $P(v,q,C')$ the subproblem of computing a feasible solution $T = (W,F)$ to the MinPICCS instance $(G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C')$ that minimizes

$$\Phi(v,q,T) := \max\{0, w_\ell^\bullet(\{v,q\}) - \ell(v)\} + \sum_{v' \in W \setminus \{v\}} \max\Big\{0, \max_{\{u,v'\} \in F} w_\ell^\bullet(\{u,v'\}) - \ell(v')\Big\}$$

under the constraints that $v \in W$ and

$$\max\{0, w_\ell^\bullet(\{v,q\}) - \ell(v)\} \geq \max\Big\{0, \max_{\{u,v\} \in F} w_\ell^\bullet(\{u,v\}) - \ell(v)\Big\} \tag{5}$$

(such a solution might not exist for some choices of v and q). We denote the cost of an optimal solution to $P(v,q,C')$ by

$$D[v,q,C'] := \min\{\Phi(v,q,T) \mid T \text{ is a feasible solution to } P(v,q,C')\}.$$

Note that the only difference between $\Phi(v,q,T)$ and the goal function (3) of MinPICCS is that the vertex v pays exactly $\max\{0, w_\ell^\bullet(\{v,q\}) - \ell(v)\}$. However, by constraint (5), v still pays at least the heaviest edge incident to v in T.

Since we want to compute $D[v,q,C']$ without looking at $G_\ell^\bullet$, the following lemma, which exploits Assumption 4.12, will be helpful.

Lemma 4.14. For each edge $\{v,q\} \in E_\ell^\bullet$ there is an edge $\{v,q'\} \in E$ with $w_\ell^\bullet(\{v,q\}) = w_\ell^\bullet(\{v,q'\})$ and $D[v,q,C'] = D[v,q',C']$.

Proof. If $\{v,q\} \in E$, then choose $q' := q$. Otherwise, $w_\ell^\bullet(\{v,q\}) = 0 = w_\ell^\bullet(\{v\})$ by Definition 4.6 and Assumption 4.12. Since $\{v\} \in E$, one can choose $q' := v$. □

By Lemma 4.14, we can compute the cost of an optimal solution to the MinPICCS instance $I^\bullet$ as

$$\min_{\{v,q\} \in E_\ell^\bullet} D[v,q,C] = \min_{\{v,q\} \in E} D[v,q,C]. \tag{6}$$

For $|C'| \leq 1$, the value of $D[v,q,C']$ can be easily computed:

Observation 1. $D[v,q,\{\mathrm{col}(v)\}] = \max\{0, w_\ell^\bullet(\{v,q\}) - \ell(v)\}$ and $D[v,q,C'] = \infty$ if $\mathrm{col}(v) \notin C'$.

We now compute $D[v,q,C']$ for $|C'| \geq 2$. To this end, we distinguish three roles that a vertex v might play in an optimal solution T to $P(v,q,C')$, which, without loss of generality, is a tree. Vertex v might be

- a cut vertex of T (Lemma 4.15), or
- a leaf and not the only vertex of its connected component of G_ℓ in T (Lemma 4.16), or
- a leaf and the only vertex of its connected component of G_ℓ in T (Lemma 4.17).

Herein, note that all recurrence relations that we prove for $D[v,q,C']$ will not refer to $G_\ell^\bullet$, but to the input graph G only, since we want to avoid the expensive construction of $G_\ell^\bullet$.

Lemma 4.15. For any $\{v,q\} \in E$ and $C' \subseteq C$ with $|C'| \geq 2$,

$$D[v,q,C'] \leq D_1[v,q,C'] := \min \left\{ \begin{array}{l} D[v,q,C_1] + D[v,q,C_2] - \max\{0, w_\ell^\bullet(\{v,q\}) - \ell(v)\} \\ \quad \text{for all } C_1 \subsetneq C' \text{ and } C_2 \subsetneq C' \\ \quad\quad \text{such that } C_1 \cup C_2 = C' \text{ and } C_1 \cap C_2 = \{\mathrm{col}(v)\} \end{array} \right\}. \tag{7}$$

If there is an optimal solution T to $P(v,q,C')$ with cut vertex v, then $D[v,q,C'] = D_1[v,q,C']$.

Proof. ($\leq$) By taking the unions of the vertex sets and edge sets of two optimal solutions for $P(v,q,C_1)$ and $P(v,q,C_2)$ with $C_1 \cup C_2 = C'$ and $C_1 \cap C_2 = \{\mathrm{col}(v)\}$, we get a feasible solution T' for $P(v,q,C')$. Since these are edge-disjoint and intersect only in v, we get

$$D[v,q,C'] \leq \Phi(v,q,T') = D[v,q,C_1] + D[v,q,C_2] - \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}.$$

($\geq$) Solution T decomposes into two proper subgraphs T_1 and T_2 only intersecting in v. For $i \in \{1,2\}$, let C_i be the set of colors of the vertices of T_i. Then, one has $C_1 \subsetneq C'$, $C_2 \subsetneq C'$, $C_1 \cup C_2 = C'$, and $C_1 \cap C_2 = \{\mathrm{col}(v)\}$. For $i \in \{1,2\}$, T_i is a feasible solution to $P(v,q,C_i)$. Thus,

$$\begin{aligned} D[v,q,C'] = \Phi(v,q,T) &= \Phi(v,q,T_1) + \Phi(v,q,T_2) - \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &\geq D(v,q,C_1) + D[v,q,C_2] - \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}. \end{aligned}$$ □

Lemma 4.16. For any $\{v,q\} \in E$ and $C' \subseteq C$ with $|C'| \geq 2$ such that the connected component of G_ℓ containing v contains a vertex with color in $C' \setminus \{\mathrm{col}(v)\}$,

$$D[v,q,C'] \leq D_2[v,q,C'] := \min_{\{u,q'\} \in E} D[u,q',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}. \tag{8}$$

If there is an optimal solution T to $P(v,q,C')$ with a leaf v, then $D[v,q,C'] = D_2[v,q,C']$.

Proof. ($\geq$) $T' = T - \{v\}$ is a feasible solution with cost at least $D[u,q',C' \setminus \{\mathrm{col}(v)\}]$ for $P(u,q',C' \setminus \{\mathrm{col}(v)\})$, where u is any vertex in T' and $\{u,q'\} \in E^\bullet_\ell$ is an edge incident to u in T' that maximizes $w^\bullet_\ell(\{u,q'\})$. Thus, using Lemma 4.14, we get

$$\begin{aligned} D[v,q,C'] &= \Phi(u,q',T') + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &\geq D[u,q',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &\geq \min_{\{u',q''\} \in E} D[u',q'',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}. \end{aligned}$$

($\leq$) Let $\{u,q'\} \in E$ be an edge minimizing $D[u,q',C' \setminus \{\mathrm{col}(v)\}]$, let T' be an optimal solution to $P(u,q',C \setminus \{\mathrm{col}(u)\})$, and u^* be a vertex in T' such that $\mathrm{col}(u^*) \in C' \setminus \{\mathrm{col}(v)\}$ and u^* is in the same connected component of G_ℓ as v. By Definition 4.6, $\{u^*,v\} \in E^\bullet_\ell$ and $w^\bullet_\ell(\{u^*,v\}) = 0$. Thus, adding this edge to T', we get a connected subgraph $T^* = (W,F)$ of $G^\bullet_\ell$ containing all colors of C'. It is a feasible solution for $P(v,q,C')$ since T^* satisfies constraint (5), that is,

$$\max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \geq 0 = \max\{0, w^\bullet_\ell(\{u^*,v\}) - \ell(v)\} = \max\Big\{0, \max_{\{u,v\} \in F} w^\bullet_\ell(\{u,v\}) - \ell(v)\Big\},$$

where the last equality is due to the fact that $\{u^*,v\}$ is the only edge incident to v in T^*. Thus,

$$\begin{aligned} D[v,q,C'] \leq \Phi(v,q,T^*) &= \Phi(u,q',T') + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &= \min_{\{u',q''\} \in E} D[u',q'',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}. \end{aligned}$$ □

Lemma 4.17. For any $\{v,q\} \in E$ and $C' \subseteq C$ with $|C'| \geq 2$ such that the connected component of G_ℓ containing v does *not* contain a vertex with color in $C' \setminus \{\mathrm{col}(v)\}$,

$$D[v,q,C'] \leq D_3[v,q,C'] := \min \left\{ \begin{array}{l} D[u,q',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ \quad \text{for all } u \in N_G(v) \text{ and } q' \in N_G(u) \\ \quad\quad \text{such that } w^\bullet_\ell(\{u,q'\}) \geq w^\bullet_\ell(\{u,v\}) \\ \quad\quad \text{and } w^\bullet_\ell(\{v,q\}) \geq w^\bullet_\ell(\{u,v\}) \end{array} \right\}. \tag{9}$$

If there is an optimal solution T to $P(v,q,C')$ with a leaf v, then $D[v,q,C'] = D_3[v,q,C']$.

Proof. ($\leq$) Let T' be an optimal solution to $P(u, q', C' \setminus \{\mathrm{col}(v)\})$ for any $u \in N_G(v)$ and $q' \in N_G(u)$ such that $w^\bullet_\ell(\{u,q'\}) \geq w^\bullet_\ell(\{u,v\})$ and $w^\bullet_\ell(\{v,q\}) \geq w^\bullet_\ell(\{u,v\})$. Adding the edge $\{u,v\}$ to T' gives a feasible solution T' for $P(v,q,C')$. Thus

$$D[v,q,C'] \leq \Phi(v,q,T') = D[u,q',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\}.$$

($\geq$) Let u be the neighbor of v in T. Since T contains no other vertices than v from the connected component of G_ℓ, by Definition 4.6 it follows that $\{u,v\} \in E$, or, equivalently, $u \in N_G(v)$. Now, let $\{u,q'\}$ be an edge in T maximizing $w^\bullet_\ell(\{u,q'\})$. Then $T - \{v\}$ is a feasible solution for $P(u,q',C' \setminus \{\mathrm{col}(v)\})$ and

$$\begin{aligned} D[v,q,C'] = \Phi(v,q,T) &= \Phi(u,q',T-\{v\}) + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &\geq D[u,q',C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \\ &= D[u,q^*,C' \setminus \{\mathrm{col}(v)\}] + \max\{0, w^\bullet_\ell(\{v,q\}) - \ell(v)\} \end{aligned}$$

for some $\{u,q^*\} \in E$ with $w^\bullet_\ell(\{u,q^*\}) = w^\bullet_\ell(\{u,q'\})$ by Lemma 4.14. Note that $q^* \in N_G(u)$ with $w^\bullet_\ell(\{u,q^*\}) = w^\bullet_\ell(\{u,q'\}) \geq w^\bullet_\ell(\{u,v\})$ since $\{u,q'\}$ maximizes $w^\bullet_\ell(\{u,q'\})$ among the edges incident to u in T. □

This completes our recurrence relations for computing $D[v,q,C']$, which we will now use to prove Proposition 4.11.

Proof of Proposition 4.11. We first compute all connected components of G in $O(n+m)$ time using depth-first search, marking each vertex with the number of the connected component it belongs to. This allows us to access $w^\bullet_\ell(\{v,q\})$ for any edge $\{v,q\} \in E$ in constant time: if v and q are in one component, then $w^\bullet_\ell(\{v,q\}) = 0$ by Definition 4.6. Otherwise, $w^\bullet_\ell(\{v,q\}) = w(\{v,q\})$.

We now sort the neighbors q of each vertex v by non-increasing edge weight $w^\bullet_\ell(\{v,q\})$ in $O(n \log n)$ total time. Then, in order to compute $D_3[v,q,C']$ in (9) quickly later, for each edge $\{v,q\} \in E$ and $u \in N_G[v]$ with $w^\bullet_\ell(\{v,q\}) \geq w^\bullet_\ell(\{u,v\})$, we precompute

$$X[v,q,u] := \max\{j \in \{1,\dots,\deg(u)\} \mid w^\bullet_\ell(\{u,q_j\}) \geq w^\bullet_\ell(\{u,v\})\},$$

where $N_G(u) = \{q_1,\dots,q_{\deg(u)}\}$ such that $w^\bullet_\ell(\{u,q_i\}) \geq w^\bullet_\ell(\{u,q_{i+1}\})$ for all $i \in \{1,\dots,j-1\}$. The computation of $X[v,q,u]$ for all $\{v,q\} \in E$ and $u \in N_G(v)$ works in $O(mn \log n)$ total time using binary search on $N_G(u)$.

Then, we compute the table entries $D[v,q,C']$ for all $\{v,q\} \in E$ and $C' \subseteq C$. The optimum of the MinPICCS instance is then given by (6). We compute the table entries for increasing cardinality of C'. While computing $D[v,q,C']$ for all $\{v,q\} \in E$ and fixed $C' \subseteq C$, we also precompute

$$Y[C'] := \min\{D[v,q,C'] \mid \{v,q\} \in E\}$$

increasing the running time by only a constant factor. Moreover, while computing $D[v,q,C']$ for fixed $C' \subseteq C$, fixed $v \in V$, and all $q \in N_G(v)$ by non-increasing edge weight, we precompute

$$Z[v,j,C'] := \min\{D[v,q_i,C' \setminus \{\mathrm{col}(v)\}] \mid i \in \{1,\dots,j\}\},$$

where $N_G(v) = \{q_1,\dots,q_{\deg(v)}\}$ such that $w^\bullet_\ell(\{v,q_i\}) \geq w^\bullet_\ell(\{v,q_{i+1}\})$ for all $i \in \{1,\dots,j-1\}$. This increases the running time only by a constant factor since $Z[v,j,C']$ is computable in constant time from $Z[v,j-1,C']$ and $D[v,q_j,C' \setminus \{\mathrm{col}(v)\}]$.

We now describe how to compute the table entries $D[v,q,C']$. By Observation 1, the $O(m)$ table entries $D[v,q,C']$ for $\{v,q\} \in E$ and $|C'| = 1$ can be computed in constant time each.

For $|C'| \geq 2$, we compute $D[v,q,C']$ under the assumption that all table entries for $C'' \subsetneq C'$ are already known. Vertex v is either a cut vertex or a leaf of an optimal solution T to $P(v,q,C')$, which, without loss of generality, is a tree. Thus, if there is a vertex u in the same component of G_ℓ as v and $\mathrm{col}(u) \in C'$, then we compute $D[v,q,C'] = \min\{D_1[v,q,C'], D_2[v,q,C']\}$ using (7) and (8). Otherwise, we compute $D[v,q,C'] = \min\{D_1[v,q,C'], D_3[v,q,C']\}$ using (7) and (9).

One can compute $D_1[v,q,C']$ from (7) in $O(3^{|C|}m)$ total time for all $\{v,q\} \in E$ and $C' \subseteq C$ since, in total, there are at most $3^{|C|}$ ways to choose $C_1 \cup C_2 = C'$ for all $C' \subseteq C$ such that $C_1 \cap C_2 = \{\mathrm{col}(v)\}$ for some $v \in V$: each element except $\mathrm{col}(v)$ is either in C_1, in C_2, or in $C \setminus (C_1 \cup C_2)$.

One can compute $D_2[v,q,C']$ from (8) in constant time for each $\{v,q\} \in E$ and $C' \subseteq C$ using $Y[C' \setminus \{v\}]$. Thus, the total time spent computing the $D_2[v,q,C']$ is $O(2^{|C|} \cdot m)$.

Finally, $D_3[v,q,C']$ from (9) can be computed in $O(n)$ time for each $\{v,q\} \in E$ and $C' \subseteq C$: we iterate over each $u \in N_G(v)$ with $w_\ell^\bullet(\{v,q\}) \geq w_\ell^\bullet(\{u,v\})$ and, for each of them, look up $Z[u,j,C' \setminus \{v\}]$ for $j = X[v,q,u]$. It follows that the total time to compute $D_3[v,q,C']$ from (9) is $O(2^{|C|}mn)$. □

Remark 4.18. Proposition 4.11 directly yields Proposition 4.5: for solving an MinPSC instance (G,w) with $G = (V,E)$, we can simply choose $\ell : V \to \mathbb{N}, v \mapsto 0$, the color set $C = \{1,\dots,n\}$, an arbitrary bijection $\mathrm{col}: V \to C$. Then, $G_\ell^\bullet = G$ and $w_\ell^\bullet = w$ and we solve the MinPICCS instance $(G_\ell^\bullet, w_\ell^\bullet, \mathrm{col}, \ell, C)$ in $O(3^n \cdot m)$ time by Proposition 4.11, which is equivalent to (G,w) by Lemmas 4.8 and 4.9.

4.5 Running time and derandomization of Algorithm 1

We can finally prove the running time, error probability, and show the derandomization of Algorithm 1, thus proving Theorem 4.3.

Proof of Theorem 4.3. To analyze the running time of Algorithm 1, note that there are $\binom{2c-2}{c}$ possibilities to enumerate all $z_1,\dots,z_c \in \mathbb{N}^+$ with $\sum_{i=1}^c z_i \leq 2c-2$ in line 2. Using Stirling's approximation

$$\sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \leq n! \leq e^{1/12n}\sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n,$$

we get that the number of iterations of the loop in line 2 is

$$\begin{aligned}\binom{2c-2}{c} &= \frac{(2c-2)!}{c! \cdot (c-2)!} \in O\left(\frac{\sqrt{2c-2}}{\sqrt{c} \cdot \sqrt{c-2}} \cdot \frac{(2c-2)^{2c-2}}{e^{2c-2}} \cdot \frac{e^c \cdot e^{c-2}}{c^c \cdot (c-2)^{c-2}}\right) = O\left(\frac{(2c-2)^{2c-2}}{\sqrt{c} \cdot c^c \cdot (c-2)^{c-2}}\right) \\ &= O\left(\frac{4^c}{\sqrt{c}} \cdot \left(\frac{c-1}{c}\right)^c \cdot \left(\frac{c-1}{c-2}\right)^{c-2}\right) = O\left(\frac{4^c}{\sqrt{c}} \cdot \left(1-\frac{1}{c}\right)^c \cdot \left(1+\frac{1}{c-2}\right)^{c-2}\right) = O\left(\frac{4^c}{\sqrt{c}}\right).\end{aligned}$$

Solving the MinPICCS instance with at most $2c-2$ colors in line 6 works in $O(3^{2c-2} \cdot m + 2^{2c-2}nm + nm \log n)$ time by Proposition 4.11. To analyze the number t of repetitions in line 4, we use $x \geq \ln(1+x)$ and again Stirling's approximation to get

$$t - 1 \leq \ln 1/\varepsilon \cdot \prod_{i=1}^c \frac{z_i^{z_i}}{z_i!} \leq \ln 1/\varepsilon \cdot \prod_{i=1}^c \frac{e^{z_i}}{\sqrt{2\pi z_i}} \leq \ln 1/\varepsilon \cdot \frac{e^{2c-2}}{\sqrt{2\pi}^c}.$$

The running time of the algorithm is thus $\ln 1/\varepsilon \cdot (4e^2/\sqrt{2\pi})^c \cdot 1/\sqrt{c} \cdot O(9^c m + 4^c nm + nm \log n)$.

(ii) To derandomize the algorithm, we use (d,k)-*perfect hash families* $\mathscr{F}$ of functions $f : \{1,\dots,d\} \to \{1,\dots,k\}$ such that, for each subset $W \subseteq \{1,\dots,d\}$ of size k, at least one of the functions in $\mathscr{F}$ is a bijection between W and $\{1,\dots,k\}$. Let n_i be the number of vertices in component G_ℓ^i. Instead of coloring the vertices in each component G_ℓ^i for $i \in \{1,\dots,c\}$ with colors from C_i randomly in line 5, we color them using all of the functions in $\mathscr{F}_i$ of an (n_i,z_i)-perfect hash family $\mathscr{F}_i$.

One can construct an (n_i,z_i)-perfect hash family $\mathscr{F}_i$ with $e^{z_i}{z_i}^{O(\log z_i)} \log n_i$ functions in $e^{z_i}{z_i}^{O(\log z_i)} \cdot n_i \log n_i$ time [17, Theorem 5.18]. Thus, in each iteration of the loop in line 2, we generate the families $\mathscr{F}_i$ for all $i \in \{1,\dots,c\}$ in

$$\sum_{i=1}^c e^{z_i}{z_i}^{O(\log z_i)} n_i \log n_i \subseteq e^c c^{O(\log c)} \log n \sum_{i=1}^c n_i = e^c c^{O(\log c)} n \log n$$

time. Then, in line 5, we color the vertices of all components of G_ℓ according to

$$\prod_{i=1}^{c} e^{z_i} z_i^{O(\log z_i)} \log n_i \subseteq (\log n)^c e^{2c} c^{O(c \log c)}$$

functions. Thus, the overall running time of the deterministic algorithm is $c^{O(c \log c)} \cdot (4e)^{2c} \cdot (\log n)^c \cdot O(9^c m + 4^c nm + nm \log n)$, which is $c^{O(c \log c)} \cdot n^{O(1)}$ [17, Exercise 3.18]. □

4.6 Hardness of provably effective data reduction

In the previous sections, we have seen that the number c of connected components of the obligatory subgraph can be effectively used to obtain fixed-parameter algorithms. We will experimentally support these findings in Section 5, where we will also find data reduction to play an important role.

However, the following theorem shows that MinPSC has no problem kernel of size polynomial in c unless the polynomial-time hierarchy collapses and, as is also conjectured, does not even have Turing kernels of size polynomial in c [28].

Theorem 4.19. MinPSC is WK[1]-hard parameterized by the number c of connected components of the obligatory subgraph G_ℓ even for the trivial vertex lower bounds $\ell(v)$ equal to the least weight of any edge incident to v. This holds even in graphs of edge weights one and two or in complete graphs with metric edge weights.

Proof. Hermelin et al. [28] have shown that the problem of checking whether a Minimum Set Cover instance $(U, \mathcal{F})$ has a solution of cost at most k is WK[1]-hard parameterized by $|U|$.

As shown in Lemma 3.8, Transformations 1 and 2 correctly reduce this problem to k-PSC. It is now enough to observe that the obligatory subgraphs of the graphs $G = (\{s\} \uplus U \uplus \mathcal{F}, E)$ generated by Transformations 1 and 2 have $|U|+1$ connected components: one component consists of the vertices $v \in \{s\} \uplus \mathcal{F}$, each of which has $\ell(v) = 1$, and $|U|$ components are formed by the vertices in $u \in U$, each of which has $\ell(u) = 2$. Thus, Transformations 1 and 2 are polynomial parameter transformations of Minimum Set Cover parameterized by $|U|$ to k-PSC parameterized by c. □

In contrast to Theorem 4.19, we point out that, using an approach of van Bevern et al. [7], given any $\varepsilon > 0$, one can reduce any instance I of MinPSC with metric edge weights to an instance I' of an annotated version of the problem with $O(c/\varepsilon)$ vertices such that any α-approximation for I' can be transformed to an $(1+\varepsilon)\alpha$-approximation for I [44]: the annotations merely keep track of vertex lower bounds and which vertices in the remaining instance were connected in the input instance.

5 Experimental evaluation

In this section, we experimentally evaluate Algorithm 1 and compare it to state of the art ILP models due to Montemanni and Gambardella [35] solved by CPLEX. In Section 5.1, we describe two data reduction rules to speed up the algorithms. In Section 5.2, we describe the data we tested the algorithms on. In Section 5.3, we describe our test environment and implementation. Finally, in Section 5.4, we present our experimental results.

5.1 Data reduction

The following data reduction rules preserve the possibility to find optimal solutions. However, by Theorem 4.19, they cannot provably lead to a problem kernel for MinPSC with size polynomial in the number c of the connected components of the obligatory subgraph.

Heavy edge deletion. The first preprocessing step is inspired by Montemanni and Gambardella [35]. The weight M of a minimum spanning tree is at most twice the cost of an optimal solution to an MinPSC instance (G, w) on a graph $G = (V, E)$ [32], so no edge $e \in E$ with weight $w(e) > 2M$ can be part of an optimal solution. Montemanni and Gambardella [35] thus suggest to delete all edges e with $w(e) > 2M$ from E. We take this thought further, incorporating vertex lower bounds $\ell : E \to \mathbb{N}$ and taking a possibly tighter upper bound than $2M$.

Let $M' \leq 2M$ be the cost of a minimum spanning tree when viewed as a solution to MinPSC. By Definition 3.1, there is an optimal solution T in which each vertex v pays at least $\ell(v)$. Since the cost of T is at most M', it cannot contain any edge $\{u, v\} \in E$ satisfying

$$\sum_{x \in V \setminus \{u,v\}} \ell(x) + \max\{\ell(u), w(\{u, v\})\} + \max\{\ell(v), w(\{u, v\})\} > M'.$$

We thus delete all such edges.

Redundant vertex deletion. While the previous data reduction rule is applicable to any solution algorithm for MinPSC, the next data reduction exploits properties of Algorithm 1. In line 7, the connected components of the obligatory subgraphs G_ℓ are treated as cliques: the MinPICCS subproblem is solved on $G_\ell^\bullet$. Thus, there is always an optimal solution to the MinPICCS subproblem that does not contain vertices of $G_\ell^\bullet$ having neighbors only in their own connected component of G_ℓ. We thus delete such vertices.

5.2 Data generation

Due to the lack of openly available benchmark instances for MinPSC, experimental works on MinPSC evaluate their algorithms on random points on a plane [3, 21, 35, 39]. As sketched in Example 4.2, in this case the number of connected components in the obligatory subgraph is likely to be $\Theta(n)$. Such test instances seem artificial in several application scenarios. Moreover, they lack any input structure, whereas the aim of our algorithm is efficiently solving MinPSC by making explicit use of input structure. The latter presumably occurs in real-world monitoring systems as, in order to guarantee a long lifetime of the sensor network, the sensor layout takes energy saving aspects into account. This aspect was taken into consideration for the two instance types that we describe in Sections 5.2.1 and 5.2.2.

In all generated instances, we choose as vertex lower bound $\ell(v)$ the least weight of any edge incident to the vertex v. For vertices incident to a single edge $\{u, v\}$, we set the vertex lower bounds $\ell(u)$ and $\ell(v)$ to cover at least the weight of $\{u, v\}$.

5.2.1 The "faulty grid" data set.

This instance set simulates the sensor fault scenario described in Example 4.2. Grid-like sensor arrangements minimize sensing area overlap when monitoring areas, which minimizes the energy for sensing and thus is important for a long lifetime of the (usually battery-powered) sensor network [48, 49]. Thus, in this data set, sensors are laid out on a triangular grid, yet we assume that several sensors fail. The goal is to restore the connectivity of the network at minimum additional cost, again in order to minimize energy consumption. An example is shown in Figure 2.

More specifically, the generation of the "faulty grid" instance is governed by two parameters: the grid size N and the number of obligatory components c. An instance is generated by assuming an infinite grid of equilateral triangles with unit edge lengths in the plane and taking the nodes of the grid that fall into a $[0, N] \times [0, N]$ rectangle. These nodes are the wireless sensors that form the vertices of a complete graph G. An edge $\{v, u\}$ in G has weight equal to the squared Euclidean distance between v and u, since by the inverse-square law, signal intensity depends inversely proportionally on the squared distance. Now, we select uniformly at random a fraction of vertices from G and delete them to simulate defective sensors. We chose the fraction to be $0.1 + \frac{1}{\sqrt{N}}$, since it yields a small yet greater than one number of obligatory components. After deleting vertices, we select graphs whose obligatory subgraph has c connected components.

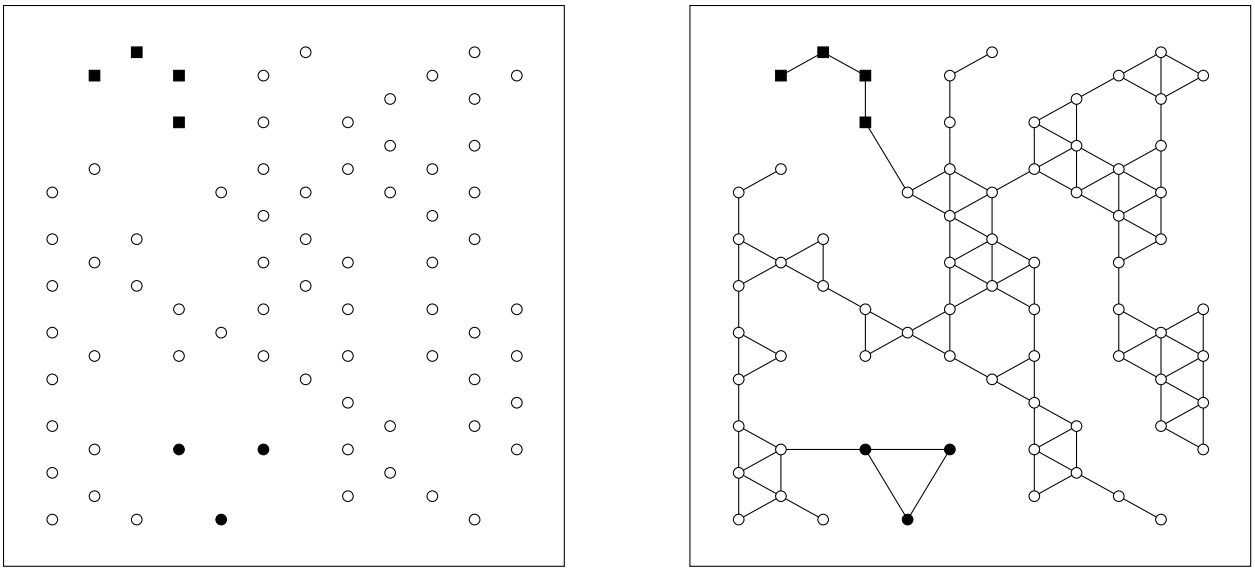


Figure 2: An instance from the "faulty grid" data set for $N = 10$ and $c = 3$ on the left and its optimal solution on the right. Sensors from distinct obligatory components are drawn using distinct marks.

We generated instances with $N \in \{10, 20, \ldots, 80\}$, and $c \in \{3, 4, 5\}$. As seen in Figure 2, these instances usually have one giant connected component and several small connected components.

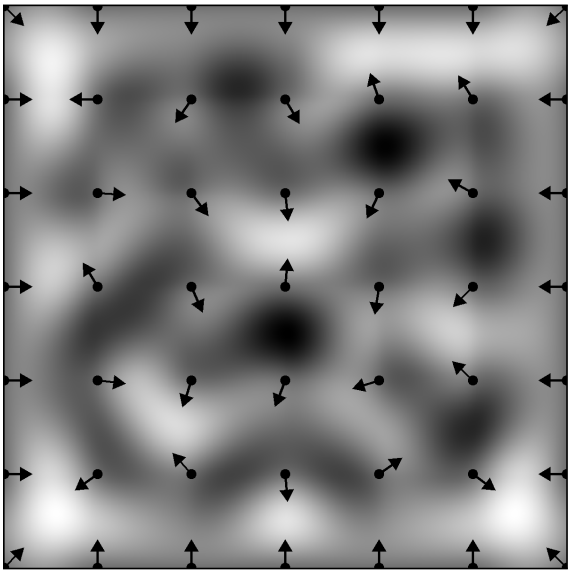
5.2.2 The "lakes" data set.

This data set corresponds to the scenario where we have to connect components of a wireless network that are already connected by vertex lower bounds (or due to constraints arising in the application). Intuitively, one can imagine the task of measuring pollution in lakes using triangular sensor grids. An instance of this type is shown in Figure 3c. The lakes are generated as follows: we generate a triangular grid and terrain data, assign a global water level to the terrain, and take the sensors lying in the lakes.

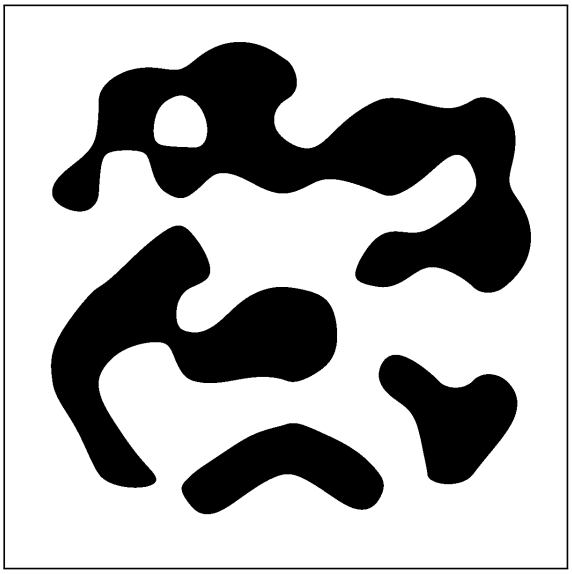
In more detail (see Figure 3) instance generation is governed by three parameters: the granularity M of the "terrain", the triangular grid size N, and the number of obligatory components c. We generate a Perlin noise function $F : [0, M-1]^2 \mapsto \mathbb{R}$ [38] over a square with edge length $M-1$. Intuitively, $F(x, y)$ gives the height of the terrain over point $(x, y) \in [0, M-1]^2$ of the square (Figure 3a). When generating the Perlin noise function F, we choose the angle of the gradient $\nabla F(x, y)$ randomly for the *inner integer* points (x, y) of the square. In order to prevent lakes from crossing the boundaries of the square, we choose $\nabla F(x, y)$ to point towards the square center for points (x, y) on the square's corners and perpendicular to the edge for the other integer points (x, y) on the square's edges, as shown in Figure 3a.

We then cover the terrain using a triangular sensor grid as in the "faulty grid" data set, yet take only those grid nodes $(x, y) \in [0, N]^2$ lying in lakes, that is, $F(x\frac{M-1}{N}, y\frac{M-1}{N}) < 0$ (Figure 3b and Figure 3c). Like in the "faulty grid" data set, all generated network nodes form a vertex set V of a complete graph G, where the weight of each edge is the squared Euclidean distance between the end points. The described procedure leaves an instance with a small number of obligatory components c. We repeat the generation process until the desired value of c is obtained.

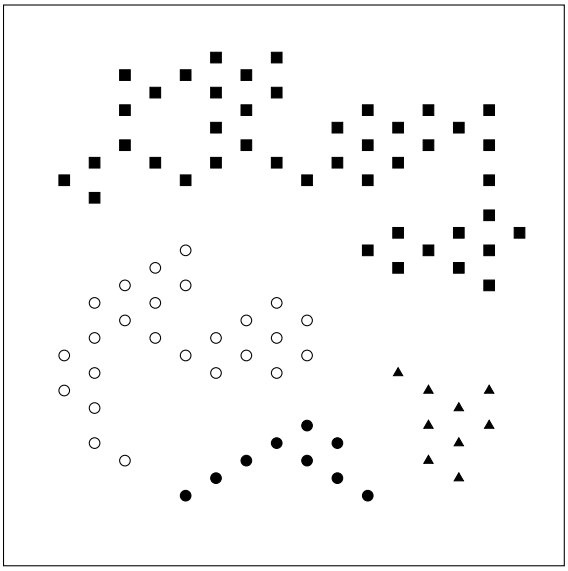
We generated instances for $M = 7$, $N \in \{10, 12, \ldots, 30\}$, and $c \in \{3, 4, 5\}$.



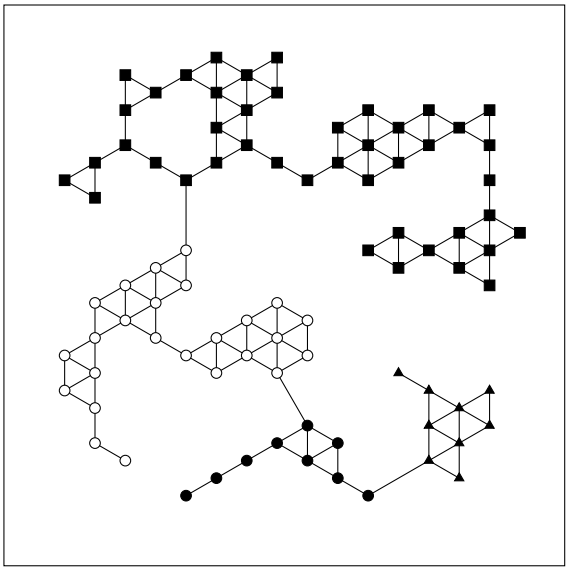
(a) Height map over a $[0, M-1]^2$ square generated from Perlin noise using randomly chosen gradient vectors in the square's integer points. A lighter shade of gray means higher.



(b) Black areas are below the "water level" and form lakes.



(c) A triangular sensor grid covers the four lakes. Distinct obligatory components are represented by distinct marks.



(d) An optimal solution.

Figure 3: Three steps of generating a "lakes" instance for $M = 7$, $N = 16$, $c = 4$, and an optimal solution.

5.3 Experimental setup.

We implemented Algorithm 1 in approximately 650 lines of C++, not including the code for the testing environment.[4] We compiled the code using GNU C++ 9.2.1 with optimization level `-O2`. The experiments were conducted on a dual core Intel Core i7-9700K CPU with 3.60 GHz and 15.6 GiB of RAM running 64-bit Ubuntu 19.10. We compared the following five algorithms.

DP1: Algorithm 1 *with* the the two data reduction rules described in Section 5.1.

DP2: Algorithm 1 *without* the the two data reduction rules described in Section 5.1.

BF: The simple $O(n^{2c})$-time brute force algorithm described in the beginning of Section 4.2.

EX1, EX2: Two state-of-the-art exact algorithms based on ILP models suggested by Montemanni and Gambardella [35], which we solve using CPLEX 12.8.[5]

For DP1 and DP2, we chose $\varepsilon = 0.1$ as an upper bound on the error probability. We implemented EX1 and EX2 of Montemanni and Gambardella [35] with the edge reduction rule from Section 5.1 and their extra inequalities (18), (19), (20), (23), (24), (25), as they were the most effective in their experiments. We point out that our implementations of algorithms EX1 and EX2 also make use of vertex lower bounds: we add ILP constraints that force edges of the obligatory subgraph into the solution.

In particular, this means that our implementation of algorithm EX2, just like our algorithms DP1 and DP2, merely has to solve the problem of connecting c components: EX2 originally consists of iteratively solving an ILP model, adding more and more connectivity constraints in each iteration. The solution to the ILP model in each iteration yields a spanning subgraph that is not necessarily connected. Its connected components are computed and, for the next iteration, a constraint is added so that at least one edge has to leave each connected component. Since our implementation in the first iteration adds the constraints that force all obligatory edges into the solution, our implementation of EX2 has all obligatory connected components available after the first iteration.

We ran the algorithms on instances described in Section 5.2. We generated 10 instances with different random seeds for each set of generation parameters.

5.4 Experimental results.

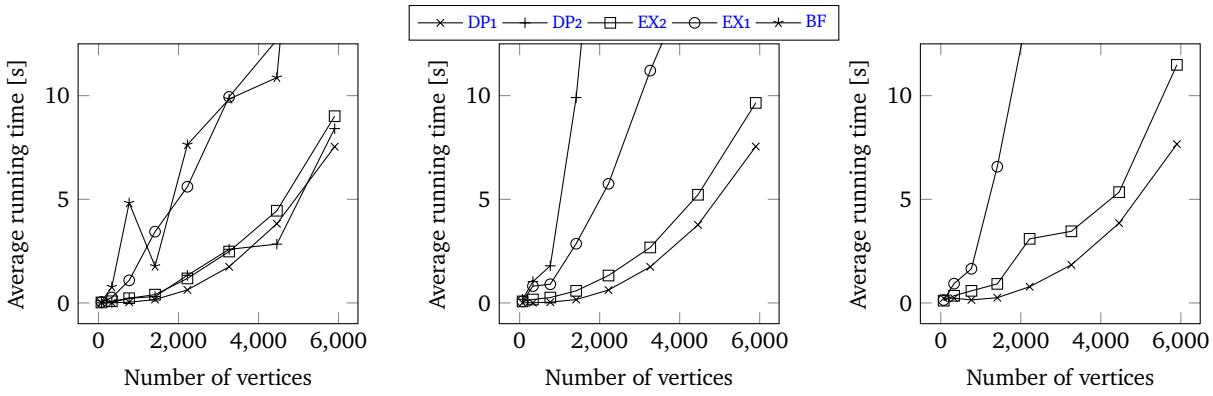
5.4.1 "Faulty grid" instances.

Figure 4 shows our experimental results on the "faulty grid" instances. The brute force algorithm BF is among the slowest already for $c = 3$ and is excluded from the plots for $c \geq 4$ since its running time on only 332 vertices varied from 45 seconds to two hours. DP2, although being one of the best algorithms for $c = 3$, is a bad choice already for $c \geq 4$, and is excluded from the plot for $c = 5$, since its running time for 2221 vertices varied between 212 seconds and 85 minutes. DP1 outperforms the integer linear programming models EX1 and EX2 of Montemanni and Gambardella [35] for all $c \in \{3, 4, 5\}$. This supports the claim that our fixed-parameter algorithm efficiently solves instances for small values of c.

5.4.2 "Lakes" instances.

Since DP1 and EX2 were the fastest algorithms in Section 5.4.1, we compare them in more detail on the "lakes" data set. Figure 5 shows the running time of DP1 and EX2 with $c \in \{3, 4, 5\}$. DP1 is at least 18.75 times faster than EX2 on instances with $c = 3$ and at least 3.75 times faster on instances with $c = 4$. For $c = 3$, most instances are solved at least 36 times faster by DP1 than by EX2. For $c = 4$, most instances are solved from 6 to 36 times faster by DP1. For $c = 5$, EX2 is faster on small instances, yet in all cases, we see that DP1 is faster on the instances that are hard to solve by both algorithms.

[4] The code and testing environment are available at https://gitlab.com/rvb/mpsc.
[5] http://www.cplex.com



(a) $c = 3$ (b) $c = 4$ (c) $c = 5$

Figure 4: Experimental evaluation of the running times of the five implementations BF, DP1, DP2, EX1, and EX2 on "faulty grid" instances with $C \in \{3, 4, 5\}$ connected components in the obligatory subgraph.

Figure 6 shows the running time of DP1 and EX2 in dependence on the number of input vertices, estimates of the mean and the standard deviation. We see not only that the running time of EX2 clearly grows superpolynomially with the number of vertices, but also has a higher variance than that of DP1. The running time of EX2 can easily vary by a factor of 100 for graphs of the same size, whereas that of DP1 varies by a factor of 10. This makes the running times of DP1 more predictable.

5.4.3 The role of data reduction.

One reason for the better performance of DP1 compared to EX2 surely is the data reduction presented in Section 5.1. On Figure 7, we see that the heavy edge deletion rule reduces the number of edges to about 25 %. Additionally removing redundant vertices reduces the instance size to about 5 %. Yet the vertex deletion rule is applicable only to DP1. Applying it to EX1 or EX2 would break the connectivity constraints and require cardinally changing the models and the additional inequalities for speeding them up. This is beyond the scope of our work, which is focused on algorithms with provable running time bounds. Moreover, data reduction is not the only reason for the better performance of DP1: Figure 6 clearly exhibits a superpolynomial running time dependency of EX2 on the number n of vertices, whereas DP1 has a proven *worst-case* running time $O(nm \log n)$ for fixed c.

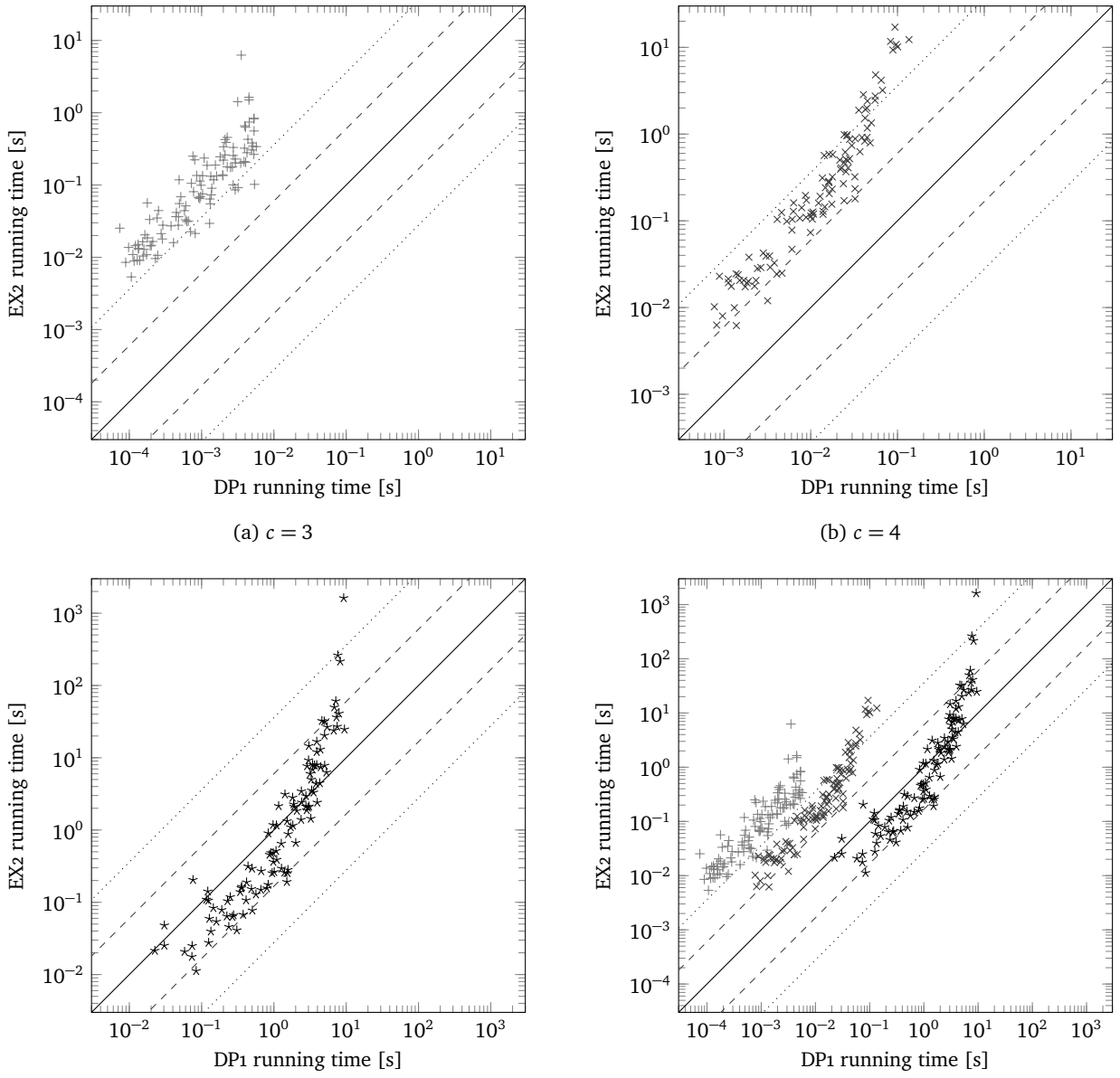
5.4.4 Error rate.

We ran the randomized algorithms DP1 and DP2 with an upper bound of $\varepsilon = 0.1$ on the error probability, yet in fact the empirical error rate was significantly lower. Neither DP1 nor DP2 gave incorrect answers on any of the "faulty grid" instances.

DP1 yielded incorrect answers on 3.6 % of 330 samples from the "lakes" data set, whereas DP2 yielded incorrect answers on 2.5 % of 320 samples (the number of conducted experiments for DP2 is lower since it was unable to finish some large instances with $c = 5$ in reasonable time). The cost of incorrect solutions was higher than the cost of an optimal solution by at most 3.5 %. We point out that, by merely doubling the running time of DP1, one can guarantee an error rate below $\varepsilon = 0.01$ and still significantly outperform EX2 on large instances.

5.4.5 Conclusion.

For small c, DP1 obviously outperforms EX2. For larger c, the advantage of DP1 over EX2 becomes smaller. Yet for large enough instances, DP1 will always outperform EX2: the running time of DP1 for constant c is merely



(a) $c = 3$

(b) $c = 4$

(c) $c = 5$

(d) All instances for $c \in \{3, 4, 5\}$ on one plot.

Figure 5: Comparison of EX2 and DP1 on the "lakes" instances. Each point is a single problem instance. DP1 is faster than EX2 if the point is above the diagonal and slower otherwise. The dashed line indicates running time difference by a factor of 6, the dotted line shows a factor of 36.

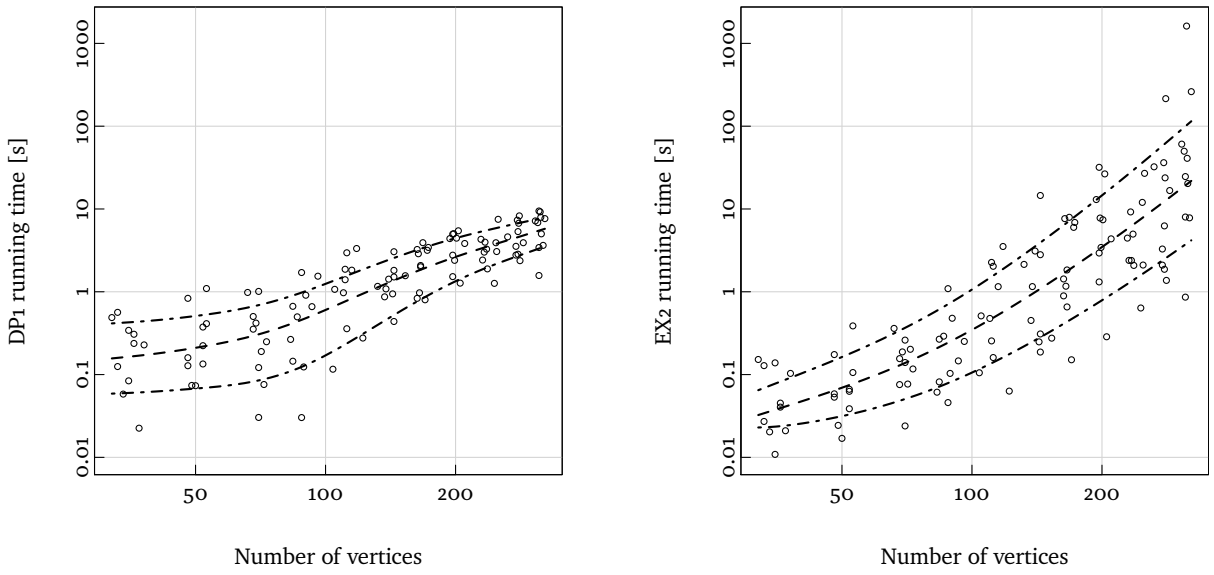


Figure 6: Running times of DP1 and DP2. Each point is an instance of "lakes" data set with $c = 5$. Estimates of the mean and the standard deviation are shown.

$O(nm \log n)$, whereas for EX2, one has to expect the running time to depend superpolynomially on n, as witnessed by Figure 6.

In general, we thus recommend to use DP1 for exactly solving MinPSC instances with obligatory subgraphs with a small number of connected components and for large graphs.

6 Conclusion

We presented a new algorithm for MinPSC that runs in polynomial time on instances in which we can find an obligatory subgraph with logarithmically many connected components. On instances with few such connected components, it outperforms state-of-the art integer linear programming models. To achieve this, data reduction played a crucial role, yet we also saw that data reduction with *provable* effect is hard.

Our algorithms are less suited for random test data (as typically used in published work so far) because our algorithms make explicit use of structure in the input that plausibly occurs in real-world monitoring instances, where the layout of the sensors in the network has to take into account energy-efficiency in order to maximize the lifetime of the sensor network.

An important theoretical and practical challenge is to find vertex lower bounds that yield obligatory subgraphs with few connected components. This goes hand in hand with identifying scenarios where (more) obligatory edges are given by the application. We identified the scenario where a sensor network lost connectivity and has to be reconnected at minimum additional energy consumption, but this also may be the case in communication networks with designated hub nodes.

Acknowledgments. The results in Sections 4.6, 5.1 and 5.4.3 were obtained while René van Bevern and Pavel V. Smirnov were supported by the Russian Foundation for Basic Research under grant 18-501-12031 NNIO_a. The results in Sections 4.1 to 4.4, 5.2.1 and 5.4.1 were obtained while René van Bevern was supported by stipend SP-2178.2019.5 of the President of the Russian Federation and Pavel V. Smirnov was supported by Mathematical Center in Akademgorodok, agreement No. 075-15-2019-1675 with the Ministry of Science and Higher Education

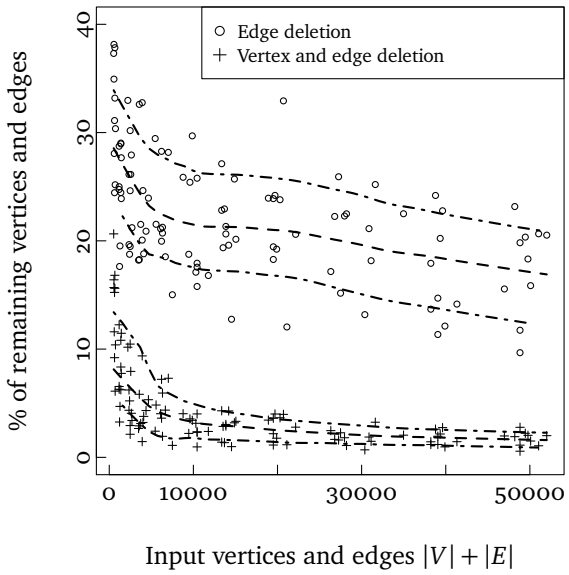


Figure 7: Effect of the data reduction described in Section 5.1 on the "lakes" instances with $c = 5$. Estimates of mean and standard deviation are shown.

of the Russian Federation. The results in Sections 3, 5.2.2 and 5.4.2 were obtained while both were supported by Mathematical Center in Akademgorodok. Both thank Oxana Yu. Tsidulko for fruitful discussions.

References

[1] Alber J, Betzler N, Niedermeier R (2006) Experiments on data reduction for optimal domination in networks. *Annals of Operations Research* 146(1):105–117. doi:10.1007/s10479-006-0045-4.

[2] Alon N, Yuster R, Zwick U (1995) Color-coding. *Journal of the ACM* 42(4):844–856. doi:10.1145/210332.210337.

[3] Althaus E, Călinescu G, Mandoiu II, Prasad SK, Tchervenski N, Zelikovsky A (2006) Power efficient range assignment for symmetric connectivity in static ad hoc wireless networks. *Wireless Networks* 12(3):287–299. doi:10.1007/s11276-005-5275-x.

[4] Bentert M, van Bevern R, Fluschnik T, Nichterlein A, Niedermeier R (2020) Polynomial-time data reduction for weighted problems beyond additive goal functions. Preprint, arXiv:1910.00277. URL https://arxiv.org/abs/1910.00277.

[5] Bentert M, van Bevern R, Nichterlein A, Niedermeier R (2017) Parameterized algorithms for power-efficient connected symmetric wireless sensor networks. Anta AF, Jurdzinski T, Mosteiro MA, Zhang Y, eds., *ALGOSENSORS 2017: Algorithms for Sensor Systems*, volume 10718 of *Lecture Notes in Computer Science*, 26–40 (Springer). doi:10.1007/978-3-319-72751-6_3.

[6] Betzler N, van Bevern R, Fellows MR, Komusiewicz C, Niedermeier R (2011) Parameterized algorithmics for finding connected motifs in biological networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8(5):1296–1308. doi:10.1109/tcbb.2011.19.

[7] van Bevern R, Fluschnik T, Tsidulko OYu (in press) On approximate data reduction for the Rural Postman Problem: Theory and experiments. *Networks* doi:10.1002/net.21985.

[8] van Bevern R, Froese V, Komusiewicz C (2018) Parameterizing edge modification problems above lower bounds. *Theory of Computing Systems* 62(3):739–770. doi:10.1007/s00224-016-9746-5.

[9] van Bevern R, Komusiewicz C, Sorge M (2017) A parameterized approximation algorithm for the mixed and windy capacitated arc routing problem: Theory and experiments. *Networks* 70(3):262–278. doi:10.1002/net.21742.

[10] van Bevern R, Smirnov PV (2020) Optimal-size problem kernels for d-hitting set in linear time and space. *Information Processing Letters* 105998. doi:10.1016/j.ipl.2020.105998.

[11] Bruckner S, Hüffner F, Karp RM, Shamir R, Sharan R (2010) Topology-free querying of protein interaction networks. *Journal of Computational Biology* 17(3):237–252. doi:10.1089/cmb.2009.0170.

[12] Calinescu G, Mandoiu II, Zelikovsky A (2002) Symmetric connectivity with minimum power consumption in radio networks. Baeza-Yates R, Montanari U, Santoro N, eds., *Foundations of Information Technology in the Era of Network and Mobile Computing*, volume 223 of *IFIP — The International Federation for Information Processing book series*, 279–294 (Springer). doi:10.1007/978-0-387-35608-2_11.

[13] Carmi P, Katz MJ (2007) Power assignment in radio networks with two power levels. *Algorithmica* 47(2):183–201. doi:10.1007/s00453-006-1230-1.

[14] Chen J, Huang X, Kanj IA, Xia G (2006) Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences* 72(8):1346–1367. doi:http://dx.doi.org/10.1016/j.jcss.2006.04.007.

[15] Clementi AEF, Huiban G, Penna P, Rossi G, Verhoeven YC (2002) Some recent theoretical advances and open questions on energy consumption in ad-hoc wireless networks. *Proceedings of the 3rd Workshop on Approximation and Randomization Algorithms in Communication Networks (ARACNE'02)*, volume 15 of *Proceedings in Informatics*, 23–38 (Carleton Scientific).

[16] Clementi AE, Penna P, Silvestri R (2004) On the power assignment problem in radio networks. *Mobile Networks and Applications* 9(2):125–140. doi:10.1023/B:MONE.0000013624.32948.87.

[17] Cygan M, Fomin FV, Kowalik L, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015) *Parameterized Algorithms* (Springer). doi:10.1007/978-3-319-21275-3.

[18] Cygan M, Pilipczuk M, Pilipczuk M, Wojtaszczyk JO (2013) On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory* 5(1):3. doi:10.1145/2462896.2462899.

[19] Dost B, Shlomi T, Gupta N, Ruppin E, Bafna V, Sharan R (2008) Qnet: A tool for querying protein interaction networks. *Journal of Computational Biology* 15(7):913–925. doi:10.1089/cmb.2007.0172.

[20] Downey RG, Fellows MR (2013) *Fundamentals of Parameterized Complexity* (Springer). doi:10.1007/978-1-4471-5559-1.

[21] Erzin AI, Mladenovic N, Plotnikov RV (2017) Variable neighborhood search variants for min-power symmetric connectivity problem. *Computers & Operations Research* 78:557–563. doi:10.1016/j.cor.2016.05.010.

[22] Erzin AI, Plotnikov RV, Shamardin YV (2013) On some polynomially solvable cases and approximate algorithms in the optimal communication tree construction problem. *Journal of Applied and Industrial Mathematics* 7(2):142–152. doi:10.1134/s1990478913020038.

[23] Flum J, Grohe M (2006) *Parameterized Complexity Theory*. Texts in Theoretical Computer Science, An EATCS Series (Springer). doi:10.1007/3-540-29953-X.

[24] Fomin FV, Lokshtanov D, Saurabh S, Zehavi M (2019) *Kernelization* (Cambridge University Press). doi:10.1017/9781107415157.

[25] Garg S, Philip G (2016) Raising the bar for vertex cover: Fixed-parameter tractability above a higher guarantee. Kraughgamer R, ed., *SODA '16: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, 1152–1166 (SIAM). doi:10.1137/1.9781611974331.ch80.

[26] de Graaf M, Boucherie RJ, Hurink JL, van Ommeren JCW (2019) An average case analysis of the minimum spanning tree heuristic for the power assignment problem. *Random Structures and Algorithms* 55(1):89–103. doi:10.1002/rsa.20831.

[27] Gutin G, Wahlström M, Yeo A (2017) Rural Postman parameterized by the number of components of required edges. *Journal of Computer and System Sciences* 83(1):121–131. doi:10.1016/j.jcss.2016.06.001.

[28] Hermelin D, Kratsch S, Sołtys K, Wahlström M, Wu X (2015) A completeness theory for polynomial (Turing) kernelization. *Algorithmica* 71(3):702–730. doi:10.1007/s00453-014-9910-8.

[29] Hoffmann S, Wanke E (2016) Minimum power range assignment for symmetric connectivity in sensor networks with two power levels. ArXiv:1605.01752.

[30] Impagliazzo R, Paturi R (2001) On the complexity of k-SAT. *Journal of Computer and System Sciences* 62(2):367–375. doi:10.1006/jcss.2000.1727.

[31] Impagliazzo R, Paturi R, Zane F (2001) Which problems have strongly exponential complexity? *Journal of Computer and System Sciences* 63(4):512–530. doi:10.1006/jcss.2001.1774.

[32] Kirousis LM, Kranakis E, Krizanc D, Pelc A (2000) Power consumption in packet radio networks. *Theoretical Computer Science* 243(1):289–305. doi:10.1016/S0304-3975(98)00223-0.

[33] Mahajan M, Raman V (1999) Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms* 31(2):335–354. doi:10.1006/jagm.1998.0996.

[34] Mellor D, Prieto E, Mathieson L, Moscato P (2010) A kernelisation approach for multiple d-hitting set and its application in optimal multi-drug therapeutic combinations. *PLOS ONE* 5(10). doi:10.1371/journal.pone.0013055.

[35] Montemanni R, Gambardella L (2005) Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. *Computers & Operations Research* 32(11):2891–2904. doi:10.1016/j.cor.2004.04.017.

[36] Niedermeier R (2006) *Invitation to Fixed-Parameter Algorithms* (Oxford University Press). doi:10.1093/acprof:oso/9780198566076.001.0001.

[37] Panigrahi D (2011) Survivable network design problems in wireless networks. Randall D, ed., *SODA '11: Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, 1014–1027 (SIAM). doi:10.1137/1.9781611973082.78.

[38] Perlin K, Hoffert EM (1989) Hypertexture. *ACM SIGGRAPH Computer Graphics* 23(3):253–262. doi:10.1145/74334.74359.

[39] Plotnikov R, Erzin A, Mladenovic N (2019) VNDS for the min-power symmetric connectivity problem. *Optimization Letters* 13(8):1897–1911. doi:10.1007/s11590-018-1324-0.

[40] Raz R, Safra S (1997) A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. Leighton FT, Shor PW, eds., *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 475–484 (ACM). doi:10.1145/258533.258641.

[41] Rodoplu V, Meng TH (1999) Minimum energy mobile wireless networks. *IEEE Journal on selected areas in communications* 17(8):1333–1344. doi:10.1109/49.779917.

[42] Schrijver A (2003) *Combinatorial optimization: polyhedra and efficiency*, volume 24 of *Algorithms and Combinatorics* (Springer).

[43] Scott J, Ideker T, Karp RM, Sharan R (2006) Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology* 13(2):133–144. doi:10.1089/cmb.2006.13.133.

[44] Smirnov PV (2020) *Razrabotka algoritmov i programmnogo obespecheniya dlya resheniya zadach analiza setevykh struktur*. Master's thesis, Novosibirsk State University, Novosibirsk, Russian Federation. URL http://rvb.su/pdf/Smi20.pdf.

[45] Sorge M, van Bevern R, Niedermeier R, Weller M (2011) From few components to an Eulerian graph by adding arcs. Kolman P, Kratochvíl J, eds., *WG 2011: Graph-Theoretic Concepts in Computer Science*, volume 6986 of *Lecture Notes in Computer Science*, 307–318 (Springer). doi:10.1007/978-3-642-25870-1_28.

[46] Sorge M, van Bevern R, Niedermeier R, Weller M (2012) A new view on Rural Postman based on Eulerian Extension and Matching. *Journal of Discrete Algorithms* 16:12–33. doi:10.1016/j.jda.2012.04.007.

[47] Williamson DP, Shmoys DB (2011) *The Design of Approximation Algorithms* (Cambridge University Press). doi:10.1017/CBO9780511921735.

[48] Zalyubovskiy VV, Erzin AI, Astrakov SN, Choo H (2009) Energy-efficient area coverage by sensors with adjustable ranges. *Sensors* 9(4):2446–2460. doi:10.3390/s90402446.

[49] Zhang H, Hou JC (2005) Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc & Sensor Wireless Networks* 1(1-2):89–124.