



Parameterized Complexity of Min-Power Asymmetric Connectivity

Matthias Bentert¹ · Roman Haag¹ · Christian Hofer¹ · Tomohiro Koana¹ · André Nichterlein¹

Published online: 23 May 2020
© The Author(s) 2020, corrected publication 2021

Abstract

We investigate parameterized algorithms for the NP-hard problem MIN-POWER ASYMMETRIC CONNECTIVITY (MINPAC) that has applications in wireless sensor networks. Given a directed arc-weighted graph, MinPAC asks for a strongly connected spanning subgraph minimizing the summed vertex costs. Here, the cost of each vertex is the weight of its heaviest outgoing arc in the chosen subgraph. We present linear-time algorithms for the cases where the number of strongly connected components in a so-called obligatory subgraph or the feedback edge number in the underlying undirected graph is constant. Complementing these results, we prove that the problem is $W[2]$ -hard with respect to the solution cost, even on restricted graphs with one feedback arc and binary arc weights.

Keywords Monitoring areas · Reconnecting sensor networks · Parameterized algorithmics

This article belongs to the Topical Collection: *Special Issue on International Workshop on Combinatorial Algorithms (IWOCA 2019)*

Guest Editors: Charles Colbourn, Roberto Grossi, Nadia Pisanti

✉ Matthias Bentert
matthias.bentert@tu-berlin.de

Roman Haag
roman.haag@campus.tu-berlin.de

Christian Hofer
hofer@campus.tu-berlin.de

Tomohiro Koana
tomohiro.koana@tu-berlin.de

André Nichterlein
andre.nichterlein@tu-berlin.de

¹ Algorithmics and Computational Complexity, Faculty IV, TU Berlin, Berlin, Germany

1 Introduction

In wireless ad-hoc networks, nodes equipped with limited power supply transmit data using a multi-hop path. We study the problem of minimizing the overall power consumption while maintaining full network connectivity, that is, each node can send messages to each other node using some (multi-hop) route through the network. Formally, we study the following optimization problem.

MIN-POWER ASYMMETRIC CONNECTIVITY (MINPAC)

Input: A strongly connected directed graph G and a weight function (cost function) $w: A(G) \rightarrow \mathbb{N}$.

Task: Find a strongly connected spanning subgraph H of G minimizing

$$\sum_{v \in V} \max_{vu \in A(H)} w(vu).$$

Related work This problem was initially formalized and shown to be NP-complete by Chen and Huang [9]. Since then, there have been numerous publications on polynomial-time approximation algorithms (an asymptotically optimal $O(\log n)$ approximation [6], a constant approximation factor with symmetric arc weights [4, 9], approximation algorithms for special cases [5, 7, 8]), and hardness results for special cases [7, 10]. To the best of our knowledge, the parameterized complexity of MINPAC has not been investigated yet.

In previous work, we investigated the parameterized complexity of the symmetric version of our problem [2]; the difference to MinPAC is that an undirected graph is given and for every undirected edge in the solution subgraph H both endpoints pay at least the weight of the edge. The asymmetric case turns out to be more involved on a technical level. However, comparable results (to the symmetric case) are achievable.

Our contribution We show algorithmic results for grid-like and tree-like input graphs as well as parameterized hardness for very restricted cases. Table 1 summarizes our results. We discuss the different parameters subsequently.

It is known that the alignment of nodes in some regular grid-like patterns is optimal to fully cover a plane. In such cases, we can assume that the *obligatory arcs*, arcs that are in any optimal solution, induce a small number c of strongly connected components as there are many arcs of minimum weight. In Section 2, we define obligatory arcs, discuss the connection to grid-like graphs, and present an algorithm that solves MINPAC in linear time when c is a constant.

In Section 3, we study MINPAC parameterized by the number of different arc weights and the vertex cover number. For this combined parameter, we present an exponential-size kernel.

In Section 4, we describe a linear-time algorithm which reduces any input instance to an equivalent instance with at most $20g - 20$ vertices and $42g - 42$ arcs, where g is the feedback edge number of the underlying undirected graph. The parameter is also motivated by real world applications in which the feedback edge number is small; for

Table 1 Overview of our results, using the following notation: n —number of vertices, m —number of arcs, c —number of strongly connected components in the obligatory subgraph (see Section 2), q —number of different arc weights, x —size of a minimum vertex cover in the underlying undirected graph (see Section 3), g —size of a minimum feedback edge set of the underlying undirected graph (see Section 4), h —size of a minimum feedback arc set (see Section 5), PAC—the decision version of MINPAC asking for a solution of cost at most k

	Result	Reference
Section 2	Dynamic programming solving MINPAC in $O(c^2 \cdot 2^c \cdot n + m + 4^c \cdot c^{2c-3/2})$ time.	Theorem 1
Section 3	An $O(\min(\{x, \log_q n\}) \cdot n + m)$ -time data reduction resulting in an equivalent MINPAC instance with at most $(q + 1)^{2x} + x$ vertices.	Theorem 2
	An exponential-size kernel with respect to $x + q$.	Corollary 1
Section 4	Linear-time data reduction resulting in an equivalent MINPAC instance with at most $20g - 20$ vertices and $42g - 42$ arcs.	Theorem 4
	A polynomial-size kernel with respect to g .	Corollary 3
Section 5	PAC is NP-hard for any $h \geq 1$.	Theorem 5
	PAC is W[2]-hard parameterized by k , even if the arcs have only cost zero or one and $h = 1$.	
	PAC is not solvable in $2^{o(n)}$ time (assuming ETH).	

instance, sensor networks along waterways (including canals) are expected to have a small number of feedback edges. It follows from our result that the problem can be solved in polynomial time for $g \in O(\log n)$, that is, for very tree-like input graphs. In terms of parameterized complexity, this gives us a partial (weights left unbounded) kernelization of MINPAC with respect to the feedback edge number. Using an existing weight-shrinking technique [3], we also provide a “full” polynomial-size kernel with respect to the feedback edge number.

Finally, in Section 5 we derive hardness results for PAC, the decision version of MINPAC. We show that even if the input graph has only binary weights and is almost a DAG (a directed acyclic graph with one additional arc), PAC parameterized by the solution cost is W[2]-hard. This is in sharp contrast to the FPT result for the parameter feedback edge number.

Preliminaries For $a \in \mathbb{N}$, we abbreviate $\{1, \dots, a\}$ by $[a]$. Throughout this work, we assume that a graph is directed unless stated otherwise. For a graph $G = (V, A)$, we write $V(G)$ to denote V and $A(G)$ to denote A . We abbreviate arcs $(u, v) \in A$ by uv . We denote by $G[V']$ the subgraph induced by $V' \subseteq V(G)$. We use $G + vu$ to denote $(V(G) \cup \{v, u\}, A(G) \cup \{vu\})$ and $G - vu$ to denote $(V(G), A(G) \setminus \{vu\})$. For a vertex $v \in V(G)$, we write $N_G^+(v) = \{u \mid vu \in A\}$ and $N_G^-(v) = \{u \mid uv \in A\}$ to denote the out- and in-neighborhood of v . We define the degree of v as $\deg_G(v) = |N_G^+(v) \cup N_G^-(v)|$. We say that $S \subseteq V(G)$ is a strongly connected component if there exists a path from each vertex $u \in S$ to every other vertex $v \in S$ in $G[S]$. We

write \mathcal{S}_G to denote the set of strongly connected components. We use U_G to denote the underlying undirected graph of G . We denote the optimal cost of an instance of MINPAC I by $\text{OPT}(I)$. The cost of a vertex subset $V' \subseteq V(G)$ in a solution with arcs $A' \subseteq A(G)$ is denoted by $\text{Cost}(V', A', w) = \sum_{v \in V'} \max_{vu \in A'} w(vu)$. For ease of notation, we write $w(vu) = \infty$ if $vu \notin A$.

A parameterized problem Π is a set of pairs (I, k) , where I denotes the problem instance and k is the parameter. The problem Π is *fixed-parameter tractable* (FPT) if there exists an algorithm solving any instance of Π in $f(k) \cdot |I|^c$ time, where f is some computable function and c is some constant. A *reduction to a problem kernel* is a polynomial-time algorithm that, given an instance (I, k) of Π , returns an equivalent instance (I', k') , such that $|I'| + k' \leq g(k)$ for some computable function g . Problem kernels are usually achieved by applying *data reduction rules*. Given an instance (I, k) for MINPAC, our data reduction rules compute in polynomial time a new instance (I', k') of MINPAC and a number d . We call a data reduction rule *correct*, if $\text{OPT}(I) = \text{OPT}(I') + d$.

2 Parameterization by the Number of Strongly Connected Components Induced by the Obligatory Arcs

In this section we present a fixed-parameter algorithm with respect to the number c of strongly connected components (SCCs) induced by *obligatory arcs*—arcs that can be included into any optimal solution with no additional cost. We find the obligatory arcs by means of lower bounds on costs paid by each vertex.

Definition 1 A *vertex lower bound* is a function $\ell: V(G) \rightarrow \mathbb{N}$ such that for any optimal solution H and any vertex $v \in V(G)$, it holds that

$$\ell(v) \leq \max_{vu \in A(H)} w(vu).$$

Observe that each vertex $v \in V(G)$ has at least one outgoing arc in any optimal solution. Hence, the cost paid by v in any optimal solution is at least $\min_{vu \in A(G)} w(vu)$. Thus, $\ell(v) \geq \min_{vu \in A(G)} w(vu)$. Moreover, if a vertex v has only one incoming arc uv , then the cost for the vertex u is at least $w(uv)$, and thus $\ell(u) \geq w(uv)$. Clearly, finding more effective but still efficiently computable vertex lower bounds is challenging on its own.

Definition 2 The *obligatory subgraph* G_ℓ induced by a vertex lower bound ℓ for G is a subgraph $(V(G), A_\ell)$, where $A_\ell = \{vu \mid w(vu) \leq \ell(v)\}$.

It has been shown that sensors are optimally placed for fully covering an area when sensors are deployed in a triangular lattice pattern [18] or a strip based-pattern [1, 15]. In such cases, there are many arcs of minimum weight. Taking these arcs usually suffices to (almost) achieve strong connectivity. So even the obligatory subgraph induced by the trivial vertex lower bound described above yields a small number of SCCs.

Let ℓ be a vertex lower bound for a graph G . We denote the number of SCCs of the obligatory subgraph G_ℓ by $c = |\mathcal{S}_{G_\ell}|$. The number c of (strongly) connected components in the obligatory subgraph has recently been used as parameter to obtain FPT results [2, 17]. In this section, we also provide an FPT result with respect to this parameter. More specifically, we will present an algorithm for MINPAC that runs in $O(2^c \cdot c^2 \cdot n + m + 4^c \cdot c^{2c-3/2})$ time. Our algorithm runs in three phases. In the first phase, it shrinks the graph to a *relevant* subgraph in which each vertex v has at most one arc towards each SCC that does not contain v (Algorithm 1). In the second phase, it uses a dynamic programming algorithm to compute the minimum cost to connect each SCC to each subset of other SCCs (Algorithm 2). In the last phase, it exhaustively tries all combinations of connecting SCCs to find an optimal solution (Algorithm 3).

Phase 1 The following lemma specifies the conditions under which we can remove arcs. It plays a central role in this phase. The basic idea herein is to remove, for each vertex $v \in V(G)$ and each SCC S , all but the cheapest arc from v to vertices in S .

Lemma 1 *Let (G, w) be an instance of MINPAC and let ℓ be a vertex lower bound. Let $S_v, S_u \in \mathcal{S}_{G_\ell}$ be two distinct SCCs and let $v \in S_v$ and $u, u' \in S_u$ be vertices of G with $w(vu) \leq w(vu')$. Then, it holds that $OPT((G, w)) = OPT((G - vu', w))$.*

Proof Observe that the removal of an arc does not decrease the cost of an optimal solution. Thus, $OPT((G, w)) \leq OPT((G - vu', w))$. Let H be an optimal solution of (G, w) . Suppose that H contained vu' . The cost paid by v in H is then at least $w(vu')$. Since u and u' both belong to S_u , the subgraph $H' = H + vu - vu'$ is a strongly connected spanning subgraph. Since $w(vu) \leq w(vu')$, it follows that H' is optimal. Moreover, H' is also a solution of $(G - vu', w)$, and thus $OPT((G, w)) \geq OPT((G - vu', w))$. \square

Algorithm 1 exhaustively removes all arcs vu' which satisfy the preconditions of Lemma 1: The algorithm iterates over each arc in G twice. It finds a minimum-weight arc from each vertex to each SCC in the first iteration. In the second iteration, it removes all but one minimum-weight arc that share the initial vertex and the SCC the terminal vertex belongs to.

We show subsequently that the resulting instance of MINPAC satisfies the properties listed in the next definition.

Definition 3 Let (G, w) be an instance of MINPAC and let ℓ be a lower bound. We say that a graph G_ℓ^{rel} and a weight function w_ℓ^{rel} with $V(G) = V(G_\ell^{\text{rel}})$, $A(G_\ell^{\text{rel}}) \subseteq A(G)$, and $w_\ell^{\text{rel}}: A(G_\ell^{\text{rel}}) \rightarrow \mathbb{N}$ are a *relevant subgraph* and *relevant weight function* induced by ℓ , respectively, if they satisfy the following properties:

- (i) $OPT((G, w)) = OPT((G_\ell^{\text{rel}}, w_\ell^{\text{rel}}))$.
- (ii) For any SCC $S \in \mathcal{S}_{G_\ell}$, it holds that $G[S] = G_\ell^{\text{rel}}[S]$.
- (iii) For any SCC $S \in \mathcal{S}_{G_\ell}$ and any vertex $v \notin S$, it holds that $|\{vu \in A(G_\ell^{\text{rel}}) \mid u \in S\}| \leq 1$.

Algorithm 1 A reduction procedure for the first phase

```

1 Function Reduction ( $G, w, \ell$ )
   //  $S_u \in \mathcal{S}_{G_\ell}$  denotes the SCC to which  $u$  belongs
2   foreach  $vu \in A(G)$  do  $M(v, S_u) \leftarrow \text{null}$ 
3   foreach  $vu \in A(G)$  do
4     if  $M(v, S_u) = \text{null}$  or  $w(vu) < w(M(v, S_u))$  then  $M(v, S_u) \leftarrow vu$ 
5   foreach  $vu \in A(G)$  do
6     if  $S_v = S_u$  then continue
7     if  $vu \neq M(v, S_u)$  then remove  $vu$  from  $G$ 
8   return ( $G, w$ )
  
```

Since it follows from property (ii) that $\mathcal{S}_{G_\ell} = \mathcal{S}_{G_\ell^{\text{rel}}}$, we will use them interchangeably.

Lemma 2 Let (G, w) be an instance of MINPAC and let ℓ be a vertex lower bound. Algorithm 1 computes in $O(m)$ time a relevant subgraph G_ℓ^{rel} and a relevant weight function w_ℓ^{rel} induced by ℓ .

Proof For any $v \in V(G)$ and any $S \in \mathcal{S}_{G_\ell}$, Algorithm 1 sets $M(v, S)$ to a minimum-weight arc from v to a vertex in S in the first iteration. In the second iteration, every arc vu is removed unless v and u belong to the same SCC or $M(v, S_u)$ is set to vu . Recall that the optimal cost remains the same after arc removals as shown in Lemma 1. Thus, Algorithm 1 returns an instance of MINPAC $(G_\ell^{\text{rel}}, w_\ell^{\text{rel}})$ that satisfies the aforementioned properties. The algorithm spends $O(m)$ time on initializing M (Line 2) and $O(m)$ time on the iteration over the arcs (Line 3 and Line 7). \square

Phase 2. In this phase, we aim to compute an optimal set of arcs to connect each SCC to all other SCCs. We start with some notation.

Definition 4 Let G_ℓ^{rel} be a relevant subgraph. For any $S \in \mathcal{S}_{G_\ell}$, we define the set of SCCs reachable from S via an arc as

$$\mathcal{S}_{G_\ell, \ell}^{\text{reach}}(S) = \{S' \in \mathcal{S}_{G_\ell} \setminus \{S\} \mid \exists vu \in A(G_\ell^{\text{rel}}), v \in S \wedge u \in S'\}.$$

We say that an arc set B is a *connector* if B connects some SCC S to some set $\mathcal{T} \subseteq \mathcal{S}_{G_\ell, \ell}^{\text{reach}}(S)$ of SCCs reachable from S . Then, our goal is to find a connector of minimum cost for each $S \in \mathcal{S}_{G_\ell}$ and each subset $\mathcal{T} \subseteq \mathcal{S}_{G_\ell, \ell}^{\text{reach}}(S)$. This allows us to compute an optimal solution with exhaustive search on connections between SCCs in the last phase

Definition 5 Let (G, w) be an instance of MINPAC and let ℓ be a vertex lower bound. A minimum-cost connector is a function MCC: $\mathcal{S}_{G_\ell} \times 2^{\mathcal{S}_{G_\ell}} \rightarrow 2^{A(G_\ell^{\text{rel}})}$ such that for any $S \in \mathcal{S}_{G_\ell}$ and any $\mathcal{T} \subseteq \mathcal{S}_{G_\ell, \ell}^{\text{reach}}(S)$ the following properties are satisfied:

Algorithm 2 A dynamic-programming procedure for the second phase.

```

1 Function DP ( $G_\ell^{rel}, w_\ell^{rel}$ )
   //  $S_v \in \mathcal{S}_{G_\ell}$  denotes the SCC to which  $v \in V(G_\ell)$  belongs
   //  $\mathcal{T}_B \subseteq \mathcal{S}_{G_\ell}$  denotes  $\{S_u \mid \exists v. vu \in B\}$  for any  $B \subseteq A(G)$ 
2 foreach  $S = \{v_1, \dots, v_{n_S}\} \in \mathcal{S}_{G_\ell}$  do
   // Initialization phase
3    $B_0 \leftarrow \emptyset, D_0(\emptyset) \leftarrow \emptyset$ 
4   foreach  $i \in [n_S]$  do
5      $B_i \leftarrow B_{i-1} \cup \{v_i u \in A(G_\ell^{rel}) \mid u \notin S, S_u \notin \mathcal{T}_{B_{i-1}}\}$ 
6     foreach  $\mathcal{T} \subseteq \mathcal{T}_{B_i}$  do  $D_i(\mathcal{T}) \leftarrow \{vu' \in B_i \mid S_{u'} \in \mathcal{T}\}$ 
   // Update phase
7   foreach  $i \in [n_S]$  do
8     foreach  $u \in \{u' \mid u' \notin S \wedge v_i u' \in A(G_\ell^{rel})\}$  do
9        $B_{i,u} \leftarrow \{v_i u' \in A(G_\ell^{rel}) \mid u' \notin S, w_\ell^{rel}(v_i u') \leq w_\ell^{rel}(v_i u)\}$ 
10      foreach  $\mathcal{T} \subseteq \mathcal{S}_{G_\ell}^{reach}(S)$  do
11        if  $\mathcal{T} \not\subseteq \mathcal{T}_{B_{i-1}} \cup \mathcal{T}_{B_{i,u}}$  then continue
12        if  $\mathcal{T} \subseteq \mathcal{T}_{B_{i-1}}$  then  $D_i(\mathcal{T}) \leftarrow D_{i-1}(\mathcal{T})$ 
13        if  $Cost(S, D_{i-1}(\mathcal{T} \setminus \mathcal{T}_{B_{i,u}}), w_\ell^{rel}) + w_\ell^{rel}(v_i u) <$ 
14           $Cost(S, D_i(\mathcal{T}), w_\ell^{rel})$  then
15           $D_i(\mathcal{T}) \leftarrow D_{i-1}(\mathcal{T} \setminus \mathcal{T}_{B_{i,u}}) \cup \{v_i u' \in B_{i,u} \mid S_{u'} \in \mathcal{T}\}$ 
16       $MCC(S, \cdot) \leftarrow D_{n_S}$ 
return MCC

```

1. For any $S' \in \mathcal{T}$, there exist vertices $v \in S$ and $u \in S'$ with $vu \in MCC(S, \mathcal{T})$.
2. There is no subset $B \subseteq A(G_\ell^{rel})$ that satisfies the above property and that satisfies $Cost(S, B, w_\ell^{rel}) < Cost(S, MCC(S, \mathcal{T}), w_\ell^{rel})$.

Algorithm 2 computes a minimum-cost connector. For each SCC $S \in \mathcal{S}_{G_\ell}$, we employ dynamic programming over vertices in S and subsets of \mathcal{S}_{G_ℓ} . This gives us a significant speed-up compared to the naïve approach of branching into at worst c different neighbors on each vertex: from $n^{\theta(c)}$ time to $O(2^c \cdot c^2 \cdot n)$ time.

Lemma 3 Given a relevant subgraph G_ℓ^{rel} and a relevant weight function w_ℓ^{rel} , Algorithm 2 computes a minimum-cost connector MCC in $O(2^c \cdot c^2 \cdot n)$ time.

Proof We fix some $S = \{v_1, \dots, v_{n_S}\}$. We use S_v to denote the SCC to which the vertex $v \in V(G_\ell^{rel})$ belongs and $\mathcal{T}_B = \{S_u \mid \exists w. wu \in B\}$ to denote the SCCs containing a terminal vertex of at least one arc in $B \subseteq A(G_\ell^{rel})$. Note that in the subsequent proof the arcs in B will all have their initial vertex in S . Moreover, the arc set B_i contains only arcs having a initial vertex in $\{v_1, \dots, v_i\} \subseteq S$ and we have $\mathcal{T}_{B_i} = \{S_u \mid \exists j \leq i. v_j u \in A(G_\ell^{rel})\}$. (See Line 5 of Algorithm 2 for the computation of the arc set B_i .) □

In order to show that the output of Algorithm 2 satisfies the property of Definition 5, 1, we prove the following stronger claim.

Claim For any $i \in \{0, \dots, n_S\}$ and any set $\mathcal{T} \subseteq \mathcal{T}_{B_i}$ of SCC, the arc set $D_i(\mathcal{T})$ contains exactly one arc that starts in $\{v_1, \dots, v_i\}$ and ends inside S' for each $S' \in \mathcal{T}$ and contains no other arc.

Proof of claim We first show by induction that the claim holds after the initialization phase (Line 3 to 6 holds for the base case $i = 0$ because we have $D_0(\emptyset) = \emptyset$. When $i \geq 1$, since G_ℓ^{rel} is a relevant subgraph (see Definition 3), it follows that the set $\{v_i u \in A(G_\ell^{\text{rel}}) \mid u \notin S, S_u \notin \mathcal{T}_{B_{i-1}}\}$ on Line 5 contains exactly one arc whose terminal vertex lies in S' for any $S' \in \mathcal{T}_{B_i} \setminus \mathcal{T}_{B_{i-1}}$. Since the claim holds for $i - 1$ by induction hypothesis, B_i contains exactly one arc that ends inside S' for each $S' \in \mathcal{T}_{B_i}$ and no other arc. In Line 6, the algorithm computes from this set B_i of arcs all arcs that end in \mathcal{T} and assigns this to $D_i(\mathcal{T})$. Hence $D_i(\mathcal{T})$ contains exactly one arc for each $S' \in \mathcal{T}$ and thus the the claim holds for i as well.

We now show—again by induction—that the claim holds after the update phase. Again the claim clearly holds for the base case $i = 0$. When $i \geq 1$, note that each iteration step (Line 9 to 14) corresponds to the case in which the cost for v_i is exactly $w(v_i u)$. The algorithm finds the set $B_{i,u}$ of arcs that v_i can cover with cost $w(v_i u)$. We verify that the claimed property is maintained after the assignment on Line 12 and Line 14. The assignment on Line 12 is clearly correct by induction hypothesis. Observe that $B_{i,u}$ contains exactly one arc whose terminal vertex lies in S' for any $S' \in \mathcal{T}_{B_{i,u}}$ since G_ℓ^{rel} is a relevant subgraph. We can assume from the induction hypothesis that $D_{i-1}(\mathcal{T} \setminus \mathcal{T}_{B_{i,u}})$ contains exactly one arc that ends inside S'' for each $S'' \in \mathcal{T} \setminus \mathcal{T}_{B_{i,u}}$ and no further arc. Thus, the assignment on Line 14 ensures that the claim is correct. □

For the second property, we prove by induction over $i \in \{0, \dots, n_S\}$ that for any $\mathcal{T} \in \mathcal{S}_{G,\ell}^{\text{reach}}(S)$, the cost of S associated with $D_i(\mathcal{T})$ is minimized when S is restricted to the vertices $\{v_1, \dots, v_i\}$. It holds in the base case $i = 0$ because we have $D_0(\emptyset) = \emptyset$. When $i \geq 1$, we assume from the induction hypothesis that values $\text{Cost}(S, D_{i-1}(\mathcal{T}), w_\ell^{\text{rel}})$ are minimum for any $\mathcal{T} \subseteq \mathcal{S}_{G,\ell}^{\text{reach}}(S)$. Assume towards a contradiction that there exists an arc set B' that satisfies $\text{Cost}(S, D_i(\mathcal{T}_{B'}), w_\ell^{\text{rel}}) > \text{Cost}(S, B', w_\ell^{\text{rel}})$. Let $B'_i \subseteq B'$ denote the set of arcs that have v_i as their initial vertex. We distinguish two cases depending on whether B'_i is empty.

Case 1. If $B'_i = \emptyset$, then B' consists of arcs whose initial vertices are in $\{v_1, \dots, v_{i-1}\}$. Then, we have

$$\text{Cost}(S, B', w_\ell^{\text{rel}}) \geq \text{Cost}(S, D_{i-1}(\mathcal{T}_{B'}), w_\ell^{\text{rel}}) \geq \text{Cost}(S, D_i(\mathcal{T}_{B'}), w_\ell^{\text{rel}}).$$

Here the first inequality follows from the induction hypothesis, and the second inequality follows from the assignment on Line 12 because $\mathcal{T}_{B'} \subseteq \mathcal{T}_{B_{i-1}}$.

Algorithm 3 An exhaustive search algorithm for MINPAC.

```

1 Function Search ( $G_\ell^{rel}, w_\ell^{rel}, MCC$ )
2    $OptCost \leftarrow \infty, \mathcal{C} \leftarrow (\mathcal{S}_{G_\ell}(S_1), \dots, \mathcal{S}_{G_\ell}(S_c))$ 
3   for  $k \leftarrow c$  to  $2c - 2$  do
4     foreach  $k_1, \dots, k_c \in \mathbb{N}$  such that  $k_1, \dots, k_c \geq 1$  and  $\sum_{i=1}^c k_i = k$  do
5       foreach  $\mathcal{T}_{S_1}, \dots, \mathcal{T}_{S_c}$  such that  $\mathcal{T}_{S_i} \subseteq \mathcal{S}_{G_\ell}^{reach}(S_i)$  and  $|\mathcal{T}_{S_i}| = k_i$  for
6         any  $i \in [c]$  do
7            $H^{aux} \leftarrow (\{v_1, \dots, v_c\}, \{v_i v_j \mid S_j \in \mathcal{T}_{S_i}\})$ 
8           if  $|\mathcal{S}_{H^{aux}}| > 1$  then continue
9            $Cost \leftarrow 0$ 
10          foreach  $S \in \mathcal{S}_{G_\ell}$  do
11             $Cost \leftarrow Cost + Cost(S, MCC(S, \mathcal{T}_S), w_\ell^{rel})$ 
12            // We assume that  $Cost(S, MCC(S, \mathcal{T}_S), w_\ell^{rel})$  is
13              computed for any  $S \in \mathcal{S}_{G_\ell}, \mathcal{T} \subseteq \mathcal{S}_{G_\ell}^{reach}(S)$  in
14              Algorithm 2
15            if  $Cost < OptCost$  then
16               $OptCost \leftarrow Cost, \mathcal{C} \leftarrow (\mathcal{T}_{S_1}, \dots, \mathcal{T}_{S_c})$ 
17          return  $(V(G), A(G_\ell) \cup \bigcup_{i=1}^c MCC(S_i, \mathcal{C}[i]))$ 

```

Case 2. If $B'_i \neq \emptyset$, then the cost of S associated with B' is

$$\begin{aligned}
 Cost(S, B', w_\ell^{rel}) &\geq Cost(S, \mathcal{D}_{i-1}(\mathcal{T}_{B'} \setminus \mathcal{T}_{B'_i}), w_\ell^{rel}) + \max_{v_i u \in B'_i} w_\ell^{rel}(v_i u) \\
 &\geq Cost(S, \mathcal{D}_i(\mathcal{T}_{B'}), w_\ell^{rel}).
 \end{aligned}$$

Here the first inequality follows from the induction hypothesis, and the second inequality follows from the assignment on Line 14. In both cases, we have $Cost(S, B', w_\ell^{rel}) \geq Cost(S, \mathcal{D}_i(\mathcal{T}_{B'}), w_\ell^{rel})$, and thus a contradiction is reached.

It remains to analyze the running time. In the initialization phase, we iterate over all vertices (Line 4) and all subsets of \mathcal{T}_{B_i} (Line 6). Note that $|\mathcal{T}_{B_i}| \leq |\mathcal{S}_{G_\ell}| = c$ as shown in the above claim and hence there are at most 2^c subsets. Each iteration takes $O(c)$ time because B_i contains at most c arcs for any $i \in [n_S]$. In the update phase, we iterate over all vertices (Line 7), at most c neighbors (Line 9), and all subsets of $\mathcal{S}_{G_\ell}^{reach}(S)$ (Line 10). Each iteration takes $O(c)$ time because $\mathcal{D}_i(\mathcal{T})$ contains at most c arcs for any $i \in [n_S]$ and any $\mathcal{T} \subseteq \mathcal{S}_{G_\ell}^{reach}(S)$. Thus, the overall running time is $O(2^c \cdot c^2 \cdot n)$.

Phase 3. We finally present the search tree algorithm for MINPAC in Algorithm 3. The algorithm “guesses” the connections between SCCs of G_ℓ to obtain an optimal solution. To this end, we first try all possible numbers of outgoing arcs from each SCC. The array \mathcal{C} contains after Line 10 for each SCC S_i in the i^{th} entry all SCCs that S_i has an arc to in the solution.

Lemma 4 Given a relevant subgraph G_ℓ^{rel} , a relevant weight function w_ℓ^{rel} , and a minimum-cost connector $MCC: \mathcal{S}_{G_\ell} \times 2^{\mathcal{S}_{G_\ell}} \rightarrow 2^{A(G_\ell^{rel})}$, Algorithm 3 computes an optimal solution of $(G_\ell^{rel}, w_\ell^{rel})$ in $O(n + m + 4^c \cdot c^{2c-3/2})$ time.

Proof We first show the correctness of Algorithm 3. In each iteration step (Line 6 and 10), the algorithm constructs an auxiliary graph H^{aux} , which is basically a graph obtained from $(V(G), \bigcup_{i=1}^c MCC(S_i, \mathcal{T}_{S_i}))$ by contracting each SCC of G_ℓ into a single vertex. Since our algorithm performs an exhaustive search, it finds a graph of cost $OPT((G_\ell^{rel}, w_\ell^{rel}))$.

We now analyze the running time of the algorithm. For each $c \leq k \leq 2c - 2$, the number of sets of integers $\{k_1, \dots, k_c\}$ that satisfy $k_i \geq 1$ for all $i \in [c]$ and $\sum_{i=1}^c k_i = k$ is

$$\binom{k-1}{c-1} \leq \binom{2c-3}{c-1} = \frac{(2c-3)!}{(c-1)!(c-2)!}$$

$$\in O\left(\frac{\sqrt{2c-3}}{\sqrt{c-1}\sqrt{c-2}} \cdot \frac{(2c-3)^{2c-3}}{(c-1)^{c-1}(c-2)^{c-2}}\right) = O\left(\frac{4^c}{\sqrt{c}}\right),$$

where the membership is due to Stirling's approximation. For each fixed set of integers $\{k_1, \dots, k_c\}$, the number of sets $\{\mathcal{T}_{S_1}, \dots, \mathcal{T}_{S_c}\}$ the algorithm generates is

$$\prod_{i=1}^c \binom{c}{k_i} \leq \prod_{i=1}^c c^{k_i} = c^{\sum_{i=1}^c k_i} = c^k.$$

So the total number of iterations (Line 6 to 10) is

$$O\left(\frac{4^c}{\sqrt{c}} \cdot \sum_{k=c}^{2c-2} c^k\right) = O(4^c \cdot c^{2c-5/2}).$$

We claim that each iteration step runs in $O(c)$ time. Computing the SCCs of H^{aux} takes $O(c)$ time because H^{aux} contains c vertices and at most $2c - 2$ arcs and the algorithm spends $O(c)$ time to compute the cost of H and update $OptCost$ and connection \mathcal{C} . Constructing the output graph takes $O(n + m)$ time. Thus, the overall running time is $O(n + m + 4^c \cdot c^{2c-3/2})$. \square

Combining Algorithm 1 to 3 we arrive at our main theorem of this section.

Theorem 1 MINPAC can be solved in $O(c^2 \cdot 2^c \cdot n + m + 4^c \cdot c^{2c-3/2})$ time.

Proof Let (G, w) be an instance of MINPAC. We run Algorithm 1 to 3 sequentially to obtain an optimal solution H of $(G_\ell^{rel}, w_\ell^{rel})$. Since G_ℓ^{rel} is a relevant subgraph of G it follows from Lemma 1 that the graph H is also an optimal solution of (G, w) .

The overall running time is then

$$O(cn + m) + O(2^c \cdot c^2 \cdot n) + O(n + m + 4^c \cdot c^{2c-3/2})$$

$$= O(c^2 \cdot 2^c \cdot n + m + 4^c \cdot c^{2c-3/2}).$$

This concludes the proof of the theorem. \square

3 Parameterization by the Number of Power Levels

It is fair to assume that the nodes cannot transmit signals with arbitrary power levels due to practical limitations [7]. In fact, many researchers have studied approximation algorithms for the MINPAC problem when only two power levels are available [4, 5, 7, 16]. In this section, we consider the case $w: A(G) \rightarrow Q$, where the set of integers $Q = \{p_1, \dots, p_q\}$ represents available power levels. The parameter q —“the number of numbers”—has been advocated by [12]. The problem remains NP-hard even when $q = 2$ [9], as also can be seen in our hardness result (Theorem 5). Thus, fixed-parameter tractability is unlikely with this parameter alone. However, using an additional parameter may alleviate this problem. We consider the *vertex cover number*, as many problems are known to become tractable when this parameter is bounded. Here we define the vertex cover number for a directed graph as the vertex cover number of the underlying undirected graph. Recall that the vertex cover number for an undirected graph is the minimum number of vertices that have to be removed to make it edgeless. Computing a minimum-cardinality vertex cover is NP-hard but any maximal matching (which can be found in linear time) gives a factor-2 approximation. We present a partial kernelization (unbounded weights) with respect to $q + x$, where x is the size of a given vertex cover. Afterwards, we strengthen this result to a proper polynomial kernel (with a worse but still polynomial running time).

Theorem 2 *Let $I = (G, w)$ be a MINPAC-instance where $w: A(G) \rightarrow Q$ and $Q \in \mathbb{N}^q$. Given I and a vertex cover X for G of size x , one can compute an instance I' of MINPAC with at most $(q + 1)^{2x} + x$ vertices and a value $d \in \mathbb{N}$ such that $OPT(I) = OPT(I') + d$ in $O(\min(\{x, \log_q n\}) \cdot n + m)$ time.*

In order to prove Theorem 2, we first observe that there are some conditions under which a vertex can be included or removed without losing the strong connectivity. Notably, we use Observation 1 to remove “twin” vertices.

Observation 1 Let G be a strongly connected graph with $u \in V(G)$. If there exists a vertex $u' \in V(G)$ with $N_G^+(u) \subseteq N_G^+(u')$ and $N_G^-(u) \subseteq N_G^-(u')$, the graph $G[V(G) \setminus \{u\}]$ is strongly connected.

Observation 2 Let G be a strongly connected graph. For any vertices $v, v' \in V(G)$, $u \notin V(G)$, the graph $G + uv + v'u$ is strongly connected.

We define “types” for vertices outside the vertex cover according to the weights of their incident arcs. This helps us to reduce the number of vertices using the observations above. Recall that we write $w(vu) = \infty$ if $vu \notin A$.

Definition 6 Let (G, w) be an instance of MINPAC with $w: A(G) \rightarrow Q$ and $Q = \{p_1, \dots, p_q\} \in \mathbb{N}^q$. Let $X = \{v_1, \dots, v_x\}$ be a vertex cover of G . The *vertex cover partition* is the partition \mathcal{P} of vertices in $V(G) \setminus X$ into sets

$$\mathcal{P}_{r_1, \dots, r_{2x}} = \{u \in V(G) \setminus X \mid \forall i \in [x]. w(uv_i) = p_{r_i} \wedge w(v_i u) = p_{r_{i+x}}\},$$

for each $r_1, \dots, r_{2x} \in [q + 1]$. Here we set $p_{q+1} = \infty$.

We initialize d with 0. In our reduction rule, we remove vertices such that, after the reduction is completed, there is at most one vertex in each set of the vertex cover partition.

Reduction Rule 1 Let $\mathcal{P}_{r_1, \dots, r_{2x}}$ be a set of the vertex cover partition with $|\mathcal{P}_{r_1, \dots, r_{2x}}| > 1$. Delete an arbitrary vertex $u \in \mathcal{P}_{r_1, \dots, r_{2x}}$ and increase d by $\min_{i \in [x]} p_{r_i}$.

Note that since the input graph is strongly connected, the increase in d in Reduction Rule 1 is at most $\max_{i \in [q]} p_i < \infty$.

Lemma 5 *Reduction Rule 1 is correct.*

Proof Let $I = (G, w)$ be an instance of MINPAC and let $I' = (G', w')$ be the instance obtained by deleting vertex u in a set $\mathcal{P}_{r_1, \dots, r_{2x}}$ of the vertex cover partition for G , as specified in Reduction Rule 1. We show that $\text{OPT}(I) = \text{OPT}(I') + \min_{i \in [x]} p_{r_i}$.

Let H be an optimal solution of I . Since $|\mathcal{P}_{r_1, \dots, r_{2x}}| > 1$, there exists a vertex $u' \in \mathcal{P}_{r_1, \dots, r_{2x}} \setminus \{u\}$. We can assume without loss of generality that $\max_{uv \in A(H)} w(uv) \leq \max_{u'v \in A(H)} w(u'v)$ holds: if it does not hold, we can exchange the role of u and u' in H without changing the cost of the solution, that is, we can update H to H' with

$$A(H') := \{vv' \mid vv' \in A(H) \wedge \{v, v'\} \cap \{u, u'\} = \emptyset\} \cup \{uv \mid u'v \in A(H)\} \\ \cup \{vu \mid vu' \in A(H)\} \cup \{u'v \mid uv \in A(H)\} \cup \{vu' \mid vu \in A(H)\}.$$

Then, we can assume that $N_H^+(u) \subseteq N_H^+(u')$ and $N_H^-(u) \subseteq N_H^-(u')$ hold (otherwise we can add the missing arcs to H without additional cost). Then, it follows from Observation 1 that $G[V(G')]$ is a solution of I' . Its cost is at most $\text{OPT}(I) - \min_{i \in [x]} p_{r_i}$ because u pays at least $\min_{i \in [x]} p_{r_i}$ in H . This shows that $\text{OPT}(I) \geq \text{OPT}(I') + \min_{i \in [x]} p_{r_i}$. For the other direction, suppose that H' is an optimal solution of I' . Let $u' \in \mathcal{P}_{r_1, \dots, r_{2x}} \setminus \{u\}$ be a vertex and let v be a vertex with $w(u'v) = \min_{i \in [x]} p_{r_i}$. Since H' is strongly connected, there exists a vertex $v' \in X$ with $v'u' \in A(H')$. Due to Observation 2, $H' + uv + v'u$ is strongly connected. Observe that the cost for u is $\min_{i \in [x]} p_{r_i}$ and the cost for v' remains unchanged because v' pays at least $w(v'u') = w(v'u)$ in H' . Hence, $\text{OPT}(I) = \text{OPT}(I') + \min_{i \in [x]} p_{r_i}$. \square

We have shown that Reduction Rule 1 is correct. It remains to show that it can be applied in $O(xn + m)$ or $O(n \log_q n + m)$ time to complete the proof of Theorem 2.

Proof of Theorem 2 We present a procedure that transforms an instance of MINPAC $I = (G, w)$ into another instance $I' = (G', w')$ with at most $(q + 1)^{2x} + x$ vertices in $O(\min(\{x, \log_q n\}) \cdot n + m)$ time. In our transformation, we distinguish two cases depending on the input size.

Case 1. If $n \leq (q + 1)^{2x} + x$, then we return I as the output of the transformation with $d = 0$.

Case 2. If $n > (q + 1)^{2x} + x$, then we apply Reduction Rule 1 exhaustively. Let \mathcal{P} be the vertex cover partition. There are at most $(q + 1)^{2x}$ sets of \mathcal{P} and each set yields at most one vertex in G' . Thus, the reduced instance has at most $(q + 1)^{2x} + x$ vertices. We show that the transformation can be performed in $O(xn + m)$ time. We first build a $2x$ -dimensional table D , where for each dimension there are $q + 1$ values. All the $(q + 1)^{2x}$ entries of D are initialized as false. (This can be done in $O(n)$ time, since $n > (q + 1)^{2x} + x$.) The entry $D[r_1, \dots, r_{2x}]$ represents whether a vertex in the set $\mathcal{P}_{r_1, \dots, r_{2x}}$ has been found. We iterate through all vertices in $V(G) \setminus X$. For each vertex $u \in V(G) \setminus X$, we set the corresponding entry in D to true if it is false, and we remove u and its incident arcs if it is true. Since accessing an entry in D takes $O(x)$ time and removing u takes $O(\deg_G(u))$ time, the transformation overall takes $O(xn + m)$ time. Note that $n > (q + 1)^{2x} + x$ yields that $2x < \log_q n$ and hence the reduction can also be done in $O(n \log_q n + m)$ time. \square

Notice that Theorem 2 does not show a kernel for the parameter combination vertex cover x plus number of numbers q . In order to obtain a kernel, we will next show how to shrink the weights.

Theorem 3 *Let $I = (G, w)$ be an instance of MINPAC where G contains n vertices and m edges. There is a polynomial-time algorithm that computes a new weight function \hat{w} such that $\|\hat{w}\|_\infty < 2^{4m^3} (4nm + 1)^{m(m+2)}$ and such that any optimal solution $T = (V, F)$ of (G, w) is also an optimal solution for (G, \hat{w}) .*

Proof We use the notion of α - \mathbb{Z} -linearizability, which uses

$$\mathbb{Z}_r := \{-r, -r + 1, \dots, r - 1, r\}$$

and is defined as follows:

Definition 7 ([3]) A function $f: L \times \mathbb{Q}^d \rightarrow \mathbb{Q}$ with $L \subseteq \Sigma^*$ is α - \mathbb{Z} -linearizable, $\alpha \in \mathbb{N}$, if for all $\omega \in \mathbb{Q}^d$ and for all $x \in L$ it holds that

1. there exists $b_x \in \mathbb{Z}_\alpha^d$ such that $f(x, \omega) = b_x^\top \omega$ and
2. for all $\omega' \in \{\omega'' \in \mathbb{Q}^d \mid \forall \beta \in \mathbb{Z}_\alpha^d. \text{sign}(\beta^\top \omega) = \text{sign}(\beta^\top \omega'')\}$ it holds that $f(x, \omega) = b_x^\top \omega$ if and only if $f(x, \omega') = b_x^\top \omega'$.

To this end, observe that we can rewrite the goal function to fit their notion as follows. Let $F_v := \{vu \in F \mid u \in N_G^+(v)\}$ and $\mathcal{F} := \{F_v \mid v \in V\}$. Then

$$\text{Cost}(V, F, w) = \sum_{F_v \in \mathcal{F}} g(F_v, w), \quad \text{with } g(F, w) = \max_{e \in F} w(e).$$

Clearly, with $A = \{e_1, \dots, e_m\}$ the function $f: E \times \mathbb{Z}^m \rightarrow \mathbb{Z}$, $f(e_i, \omega) \mapsto \omega_i := w(e_i)$ is 1- \mathbb{Z} -linearizable. On the one hand, we have that $f(e_i, \omega) = \mathbf{e}_i^\top \omega$ (where \mathbf{e}_i denotes the unit vector with the i^{th} entry being one). On the other hand, for all $\omega' \in \mathbb{Z}^m$ it holds true that $f(e_i, \omega) = \mathbf{e}_i^\top \omega$ if and only if $f(e_i, \omega') = \mathbf{e}_i^\top \omega'$.

By Lemma 4.8 in [3], it follows that $\text{Cost}(V, F, w)$ is $2n$ - \mathbb{Z} -linearizable. Finally, Theorem 4.7 in [3] yields the desired weight function \hat{w} . \square

Combining Theorems 2 and 3 gives us the desired kernel.

Corollary 1 *MINPAC admits an exponential-size kernel with respect to the combined parameter vertex cover plus number of numbers.*

4 Parameterization by Feedback Edge Number

In this section we describe a kernelization for MINPAC parameterized by the *feedback edge number*. The feedback edge number for an undirected graph is the minimum number of edges that have to be removed in order to make it a forest. We define the feedback edge number for a directed graph G as the feedback edge number of its underlying undirected graph U_G . Note that a minimum feedback edge set can be computed in linear time. In Section 5, we will show that the parameter *feedback arc number*, which is the directed counterpart of the feedback edge number, does not allow the design of an FPT algorithm for MINPAC unless $P = NP$.

The feedback edge number measures how tree-like the input is. From a theoretical perspective this is interesting to analyze because any instance (G, w) of MINPAC is easy to solve if U_G is a tree. In this case all edges of U_G must correspond to arcs in both directions in G and the optimal solution is G itself. The parameter is also motivated by real world applications in which the feedback edge number is small; for instance, sensor networks along waterways (including canals) are expected to have a small number of feedback edges. In this section we first prove the following theorem which states that MINPAC admits a partial kernel with respect to feedback edge number. Afterwards, we strengthen this result to a proper polynomial kernel (with a worse but still polynomial running time).

Theorem 4 *In linear time, one can transform any instance $I = (G, w)$ of MINPAC with feedback edge number g into an instance $I' = (G', w')$ and compute a value $d \in \mathbb{N}$ such that G' has at most $20g - 20$ vertices, $42g - 42$ arcs, and $\text{OPT}(I) = \text{OPT}(I') + d$.*

Corollary 2 *MINPAC can be solved in $O(2^{O(g)} + n + m)$ time.*

We will present a set of data reduction rules which shrink any instance of MINPAC to an essentially equivalent instance whose size is bounded as specified in Theorem 4. We simultaneously compute the value d , which we initialize with 0.

Our first reduction rule reduces the weights of arcs outgoing from a vertex by the weight of its cheapest outgoing arc. This ensures that each vertex has at least one outgoing arc of weight zero.

Reduction Rule 2 Let v be a vertex with $\delta_v := \min_{vu \in A(G)} w(vu) > 0$. Update the weights and d as follows:

- (i) $w(vu) = w(vu) - \delta_v$ for each $vu \in A(G)$.
- (ii) $d := d + \delta_v$.

Lemma 6 *Reduction Rule 2 is correct.*

Proof Let $I = (G, w)$ be an instance of MINPAC and let $v \in V(G)$ be a vertex with $\delta_v = \min_{vu \in A(G)} w(vu) > 0$. Let I' be an instance with reduced weights using Reduction Rule 2. We show that $\text{OPT}(I) = \text{OPT}(I') + \delta_v$.

Let H be an optimal solution of I . Then, H is also a solution of I' , where the cost for v is decreased by δ_v and the cost for every other vertex remains identical. Thus, $\text{OPT}(I')$ is at most $\text{OPT}(I) - \delta_v$ and we obtain $\text{OPT}(I) \geq \text{OPT}(I') + \delta_v$. For the other direction, let H be an optimal solution for I' . Then H is also a solution for I , where the cost for v is increased by δ_v and the cost for every other vertex remains identical. Thus, $\text{OPT}(I)$ is at most $\text{OPT}(I') + \delta_v$ and we obtain $\text{OPT}(I) = \text{OPT}(I') + \delta_v$. □

Our next reduction rule discards all degree-one vertices.

Reduction Rule 3 Let v be a vertex with $\text{deg}_G(v) = 1$ and let u be its neighbor. Update (G, w) and d as follows:

- (i) $G := G[V(G) \setminus \{v\}]$.
- (ii) $w(uv') := \max\{0, w(uv') - w(uv)\}$ for each $uv' \in A(G) \setminus \{uv\}$.
- (iii) $d := d + w(vu) + w(uv)$.

Lemma 7 *Reduction Rule 3 is correct.*

Proof Let $I = (G, w)$ be an instance of MINPAC with an optimal solution H . Let $v \in V(G)$ be a vertex with $\text{deg}_G(v) = 1$ and u be its neighbor. (Since G is strongly connected, we have $uv \in A(G)$ and $vu \in A(G)$.) Let $I' = (G', w')$ be the instance in which v is removed according to Reduction Rule 3. Then, $H[V(G')]$ is a solution of I' . Since the cost for u decreases by $\max_{uv' \in A(H)} w(uv') - \max_{uv' \in A(H) \setminus \{uv\}} w'(uv') = w(uv)$ and the costs for other vertices remain unchanged, we have $\text{OPT}(I) \geq \text{OPT}(I') + w(vu) + w(uv)$. For the other direction, let H' be an optimal solution of I' . Then, $H' + vu + uv$ is a solution of I . The cost for v is $w(vu)$ and the cost for u is $\max_{uv' \in A(H') \cup \{uv\}} w(uv') = w(uv) + \max_{uv' \in A(H')} w'(uv')$, while the costs for other vertices remain the same. Thus, we obtain $\text{OPT}(I) \leq \text{OPT}(I') + w(vu) + w(uv)$. □

Lemma 8 *Reduction Rules 2 and 3 can be exhaustively applied in linear time.*

Proof For each vertex $v \in V(G)$, set $\ell(v) := \min_{vu \in A(G)} w(vu)$. Let L be a list of degree-1 vertices. We apply the following procedure as long as L is nonempty. Let v be the vertex taken from L and let u be its neighbor. Remove v and its incident arcs from G , set $\ell(u) := \max\{\ell(u), w(uv)\}$, and update $d := d + \max\{w(vu), \ell(v)\}$. If the degree of u becomes 1 after deleting v , then add u to L . Once L is empty, update the weight of each remaining arc $w(vu) := \max\{0, w(vu) - \ell(v)\}$. Finally,

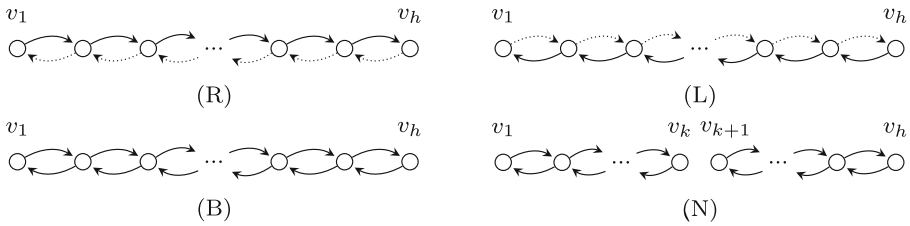


Fig. 1 Visualization of the four cases for connectivity inside maximal induced paths (see Observation 3). For cases (R) and (L), at least one of the dotted arcs is not present

update $d := d + \ell(v)$ for each remaining vertex v . It is easy to see that the algorithm runs in linear time. \square

Henceforth, we can assume that Reduction Rules 2 and 3 are exhaustively applied. Thus, the underlying undirected graph U_G will have no degree-one vertices. It remains to bound the number of vertices that have degree two in U_G . Once this is achieved, we can use standard arguments to upper-bound the size of the instance [2].

The rough idea to bound the number of degree-two vertices is as follows: In order to upper-bound the number of degree-two vertices in U_G , we consider long paths in U_G . A path $P = (v_0, \dots, v_{h+1})$ in U_G is a *maximal induced path* of G if $\deg_G(v_0) > 2$, $\deg_G(v_{h+1}) > 2$, and $\deg_G(v_i) = 2$ for all $i \in [h]$. We call the vertices $\{v_i \mid i \in [h]\}$ the inner vertices of P . We will replace the inner vertices of each maximal induced path on at least seven vertices with a fixed gadget. The arc-weights in the gadget are chosen such that the four possible ways in which the outermost inner vertices are connected inside the path (see Fig. 1 for a visualization of the four cases) are preserved.

Observation 3 Let $I = (G, w)$ be an instance of MINPAC with an optimal solution H . Let $P = (v_0, \dots, v_{h+1})$ be a maximal induced path of G . Then, there are four cases in which v_1 and v_h are connected inside P in H (Fig. 1):

- Case (R) It holds for any $i \in [h - 1]$ that $v_i v_{i+1} \in A(H)$ and there exists $k \in [h - 1]$ such that $v_{k+1} v_k \notin A(H)$.
- Case (L) It holds for any $i \in [h - 1]$ that $v_{i+1} v_i \in A(H)$ and there exists $k \in [h - 1]$ such that $v_k v_{k+1} \notin A(H)$.
- Case (B) It holds for any $i \in [h - 1]$ that $v_i v_{i+1} \in A(H)$ and $v_{i+1} v_i \in A(H)$.
- Case (N) There exists $k \in [h - 1]$ such that $v_k v_{k+1} \notin A(H)$ and $v_{k+1} v_k \notin A(H)$, and thus it holds for any $i \in \{1, \dots, k - 1, k + 1, \dots, h - 1\}$ that $v_i v_{i+1} \in A(H)$ and $v_{i+1} v_i \in A(H)$.

Proof If $v_i v_{i+1} \in A(H)$ holds for all $i \in [h - 1]$ or $v_{i+1} v_i \in A(H)$ holds for all $i \in [h - 1]$, then we have one of the three cases (R), (L), or (B). Otherwise there exist $k, k' \in [h - 1]$ such that $v_k v_{k+1} \notin A(H)$ and $v_{k'+1} v_{k'} \notin A(H)$. We show that this corresponds to the case (N). To this end, we show that $k = k'$. If $k < k'$, then there is no outgoing arc from any vertices of $S_1 = \{v_{k+1}, \dots, v_{k'}\}$ to $V(G) \setminus S_1$. This is contradicting the assumption that H is a solution, and hence $k \geq k'$. If $k > k'$, then

there is no incoming arc to any vertices of $S_2 = \{v_{k'+1}, \dots, v_k\}$ from $V(G) \setminus S_2$. Again this is a contradiction, and hence we obtain $k \leq k'$. \square

Before giving the gadget to replace the inner vertices of a maximal induced path, we define the cost for the inner vertices in cases (R), (L), and (N).

Definition 8 Let (G, w) be an instance of MINPAC and let $P = (v_0, \dots, v_{h+1})$ be a maximal induced path of G . We define the cost for the connection inside P in the right direction, the left direction, and neither direction as follows:

$$C_R := \sum_{i=1}^{h-1} w(v_i v_{i+1}), \quad C_L := \sum_{i=1}^{h-1} w(v_{i+1} v_i),$$

$$C_N := \sum_{i=1, i \neq k}^{h-1} w(v_i v_{i+1}) + w(v_{i+1} v_i),$$

where

$$k := \operatorname{argmax}_{i \in [h-1]} (w(v_i v_{i+1}) + w(v_{i+1} v_i)).$$

Note that $C_R = \infty$ (or $C_L = \infty$) if $v_i v_{i+1} \notin A(G)$ (or $v_{i+1} v_i \notin A(G)$) for some i (recall that $w(vu) = \infty$ for $vu \notin A(G)$). We finally present the gadget to replace the inner vertices of a maximal induced path. The gadget is somewhat more involved than the gadget used in the symmetric version of MINPAC [2] because it needs to encode the four cases seen in Observation 3.

Definition 9 Let $P = (v_0, \dots, v_{h+1})$ be a maximal induced path. The path-gadget for P is a graph on 6 vertices $\{v_1, v_h, a_1, a_2, b_1, b_2\}$ and 10 arcs $\{v_1 a_1, a_1 v_1, v_h a_2, a_2 v_h, a_1 b_1, a_2 b_2, b_1 a_2, b_2 a_1, a_1 b_2, a_2 b_1\}$ with weights defined as follows:

$$w(v_1 a_1) := 0, \quad w(a_1 v_1) := 0, \quad w(v_h a_2) := 0, \quad w(a_2 v_h) := 0,$$

$$w(a_1 b_1) := C_R, \quad w(a_2 b_2) := C_L, \quad w(b_1 a_2) := 0, \quad w(b_2 a_1) := 0,$$

$$w(a_1 b_2) := \begin{cases} C_R & \text{if } C_R \leq C_N \text{ or } C_L \leq C_N, \\ \left\lceil \frac{1}{2} C_N \right\rceil & \text{otherwise,} \end{cases}$$

$$w(a_2 b_1) := \begin{cases} C_L & \text{if } C_R \leq C_N \text{ or } C_L \leq C_N, \\ \left\lceil \frac{1}{2} C_N \right\rceil & \text{otherwise.} \end{cases}$$

Observation 4 In Definition 9, it always holds that $w(a_1 b_2) \leq C_R$, $w(a_2 b_1) \leq C_L$, and $w(a_1 b_2) + w(a_2 b_1) \geq C_N$.

Reduction Rule 4 Let $P = (v_0, \dots, v_{h+1})$ be a maximal induced path of G with $h \geq 7$. Then, remove the vertices v_2, \dots, v_{h-1} of P , add a path-gadget for P with endpoints v_1 and v_h (see Fig. 2), and keep d unchanged.

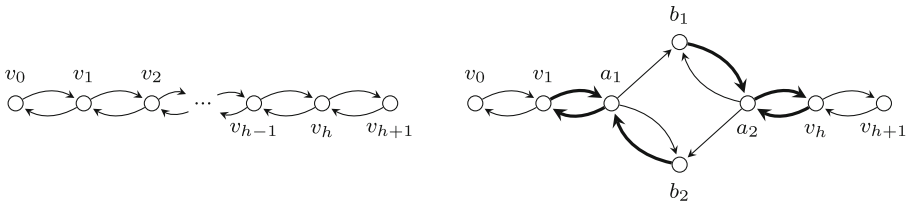


Fig. 2 Illustration of Reduction Rule 4. We replace the inner vertices of a maximal induced path with a path-gadget. Bold arcs denote arcs of weight 0. For the weights of other arcs in the path-gadget, see Definition 9. The value d remains unchanged, that is, the cost in both instances are the same

Lemma 9 *Reduction Rule 4 is correct and can be exhaustively applied in linear time.*

Proof Let $I = (G, w)$ be an instance of MINPAC and $P = (v_0, \dots, v_{h+1})$ be a maximal induced path of G with $h \geq 7$. Let $I' = (G', w')$ be the instance where the inner vertices of P are replaced by the path-gadget for P . We use V_{in} to denote the inner vertices $\{v_1, \dots, v_h\}$ of P and V_{new} to denote $\{v_1, v_h, a_1, a_2, b_1, b_2\}$ the new vertices in the path-gadget. We show that $OPT(I) = OPT(I')$.

(\geq) Let H be an optimal solution of I . Let $B = A(H) \setminus \{v_i v_{i+1}, v_{i+1} v_i \mid i \in [h - 1]\}$ be the set of arcs in H that do have the initial or terminal vertex outside V_{in} . Let $B_0 = \{v_1 a_1, a_1 v_1, v_h a_2, a_2 v_h, b_1 a_2, b_2 a_1\}$ be weight-zero arcs inside V_{new} and let $B_1 = \{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}$ be the arcs inside V_{new} that have weight at least one. We will construct a solution H' of I' such that $A(H') = B \cup B_0 \cup B'_1$ for some $B'_1 \subseteq B_1$ which we specify later (in a case distinction). Thus, in this construction the arcs outside V_{in} and V_{new} remain identical in H and in H' . Hence, it is sufficient to compare the cost for V_{in} in H and the costs for V_{new} in H' . To this end, for $X \subseteq A(G)$ we define an auxiliary function for the weight of an arc

$$w_X(vu) = \begin{cases} w(vu) & \text{if } vu \in X, \\ 0 & \text{otherwise.} \end{cases}$$

We define w'_X analogously for w' of I' . Using this notation, the cost for V_{in} reads

$$\begin{aligned} \text{Cost}(V_{in}, A(H), w) &= \sum_{i=1}^h \max\{w_{A(H)}(v_i v_{i-1}), w_{A(H)}(v_i v_{i+1})\} \\ &= \sum_{i=1}^h w_{A(H)}(v_i v_{i-1}) + w_{A(H)}(v_i v_{i+1}). \\ &= w_B(v_1 v_0) + w_B(v_h v_{h+1}) \\ &\quad + \sum_{i=1}^{h-1} w_{A(H)}(v_i v_{i+1}) + w_{A(H)}(v_{i+1} v_i). \end{aligned}$$

Here the second equality follows from the assumption that Reduction Rule 2 is applied: it holds for any $i \in [h]$ that at least one of $w(v_i v_{i-1})$ or $w(v_i v_{i+1})$ is 0. On the other hand, the cost for V_{new} reads

$$\begin{aligned} \text{Cost}(V_{\text{new}}, A(H'), w') &= \mathbf{w}_B(v_1 v_0) + \mathbf{w}_B(v_h v_{h+1}) + \max\{\mathbf{w}'_{B'_1}(a_1 b_1), \mathbf{w}'_{B'_1}(a_1 b_2)\} \\ &\quad + \max\{\mathbf{w}'_{B'_1}(a_2 b_1), \mathbf{w}'_{B'_1}(a_2 b_2)\}. \end{aligned}$$

We show that $\delta := \text{Cost}(V_{\text{in}}, A(H), w) - \text{Cost}(V_{\text{new}}, A(H'), w') \geq 0$. We rewrite δ , by canceling out the terms $\mathbf{w}_B(v_1 v_0)$ and $\mathbf{w}_B(v_h v_{h+1})$, as

$$\delta = \text{Cost}(V_{\text{in}}, A(H), w) - \text{Cost}(V_{\text{new}}, A(H'), w') = C_I - C_{I'}$$

where C_I and $C_{I'}$ are given by

$$\begin{aligned} C_I &= \sum_{i=1}^{h-1} \mathbf{w}_{A(H)}(v_i v_{i+1}) + \mathbf{w}_{A(H)}(v_{i+1} v_i) \text{ and} \\ C_{I'} &= \max\{\mathbf{w}'_{B'_1}(a_1 b_1), \mathbf{w}'_{B'_1}(a_1 b_2)\} + \max\{\mathbf{w}'_{B'_1}(a_2 b_1), \mathbf{w}'_{B'_1}(a_2 b_2)\}. \end{aligned}$$

We distinguish between the four cases shown in Observation 3.

- Case (R). We set $B'_1 := \{a_1 b_1, a_1 b_2\}$. Then, H' is a solution because the connectivity from v_1 to v_h inside the gadget is preserved. Since H contains arcs $v_i v_{i+1}$ for all $i \in [h-1]$, we have $C_I \geq C_R$. As noted in Observation 4, we have $C_{I'} = \max\{w(a_1 b_1), w(a_1 b_2)\} \leq C_R \leq C_I$.
- Case (L). We set $B'_1 := \{a_2 b_1, a_2 b_2\}$. Then, H' is a solution because the connectivity from v_h to v_1 inside the gadget is preserved. Since H contains arcs $v_{i+1} v_i$ for all $i \in [h-1]$, we have $C_I \geq C_L$. As noted in Observation 4, we have $C_{I'} = \max\{w(a_2 b_1), w(a_2 b_2)\} \leq C_L \leq C_I$.
- Case (B). We set $B'_1 = \{a_1 b_1, a_2 b_2\}$. Then, H' is a solution because the connectivities from v_1 to v_h and from v_h to v_1 are both preserved. Since H contains arcs $v_i v_{i+1}$ and $v_{i+1} v_i$ for all $i \in [h-1]$, we have $C_I \geq C_R + C_L$. We also have $C_{I'} = C_R + C_L$ because the cost for a_1 and a_2 are C_R and C_L , respectively.
- Case (N). Here we have $C_I = C_N$. We further distinguish three subcases. If $C_R \leq C_N$, then we set $B'_1 := \{a_1 b_1, a_1 b_2\}$. Then, H' is a solution with $C_{I'} = C_R \leq C_N$. If $C_L \leq C_N$, then we set $B'_1 := \{a_2 b_1, a_2 b_2\}$. Then, H' is a solution with $C_{I'} = C_L \leq C_N$. Otherwise we set $B'_1 := \{a_1 b_2, a_2 b_1\}$. Then, H' is a solution with $C_{I'} = C_N$.
- (\leq) Let H' now be an optimal solution of I' . We can assume that H' contains all weight-zero arcs in B_0 and some of the non-zero-weight arcs in B_1 . Let $B = A(H') \setminus (B_0 \cup B_1)$. We construct a solution H such that $A(H) \supseteq$

B. We will give H by specifying $B' = A(H) \setminus B$. By the same argument as before we compare the following two quantities:

$$C_I = \sum_{i=1}^{h-1} \mathbf{w}_{B'}(v_i v_{i+1}) + \mathbf{w}_{B'}(v_{i+1} v_i) \text{ and}$$

$$C_{I'} = \max\{\mathbf{w}'_{A(H')}(a_1 b_1), \mathbf{w}'_{A(H')}(a_1 b_2)\} \\ + \max\{\mathbf{w}'_{A(H')}(a_2 b_1), \mathbf{w}'_{A(H')}(a_2 b_2)\}.$$

- Case 1. If there is a path from v_1 to v_h but no path from v_h to v_1 inside the path-gadget, then we set $B' = \{v_i v_{i+1} \mid i \in [h - 1]\}$. Then, H is a solution with $C_I = C_R$. Since H' contains $a_1 b_1$, it holds that $C_{I'} \geq C_R$.
- Case 2. If there is a path from v_h to v_1 but no path from v_1 to v_h inside the path-gadget, then we set $B' = \{v_{i+1} v_i \mid i \in [h - 1]\}$. Then, H is a solution with $C_I = C_L$. Since H' contains $a_2 b_2$, it holds that $C_{I'} \geq C_L$.
- Case 3. If there is paths from v_1 to v_h and from v_h to v_1 inside the path-gadget, then we set $B' = \{v_i v_{i+1}, v_{i+1} v_i \mid i \in [h - 1]\}$. Then, H is a solution with $C_I = C_R + C_L$. Since H' contains $a_1 b_1$ and $a_2 b_2$, it holds that $C_{I'} \geq C_R + C_L$.
- Case 4. If there is neither a path from v_1 to v_h nor from v_h to v_1 inside the path-gadget, then we set $B' = \{v_i v_{i+1}, v_{i+1} v_i \mid i \in [h - 1], i \neq k\}$, where k is $\arg \max_{i \in [h-1]} w(v_i v_{i+1}) + w(v_{i+1} v_i)$. Then, H is a solution with $C_I = C_N$. Since H' neither contains $a_1 b_1$ nor $a_2 b_2$ in this case, it must contain $a_1 b_2$ and $a_2 b_2$ so that b_1 and b_2 can be reached in H' . As noted in Observation 4, we have $C_{I'} \geq w(a_1 b_2) + w(a_2 b_2) \geq C_N$.

Running time To find maximal induced paths, we start with a degree-two vertex and traverse in both directions until a vertex with degree at least 3 is discovered. If the maximal induced path contains at least 7 inner vertices, then we replace it with a gadget with appropriate weights. The algorithm spends a constant time for each inner vertex in the maximal induced path. Since inner vertices of maximal induced paths are pairwise disjoint, this procedure applies Reduction Rule 4 exhaustively in linear time. □

Remark 1 Reduction Rule 4 cannot be applied when U_G is a large cycle because there is no vertex with degree 3 or larger. However, if U_G is a cycle, then we can easily compute a solution: Let $v \in V(G)$ be an arbitrary vertex. Then, compute costs corresponding to the cases (R), (L), and (N) with $v_1 = v_h = v$ (see Fig. 1). Take the cheapest solution found.

We have so far shown a reduction rule to remove all degree-one vertices and a gadget to replace every maximal induced path with a fixed number of vertices. As shown in previous work [2], this is sufficient to obtain a linear-size kernel.

Proposition 1 ([2]) *Any undirected graph G without degree-one vertices contains at most $2g - 2$ vertices of degree at least three, where g is the feedback edge number of G .*

Proposition 2 ([2]) *Any connected undirected graph G without degree-one vertices consists of at most $3g - 3$ maximal induced paths, where $g \geq 2$ is the feedback edge number of G .*

We use the two propositions above to prove the main theorem of this section.

Proof of Theorem 4 Let $I = (G, w)$ be an instance of MINPAC with feedback edge number g . We apply Reduction Rules 2 and 3 exhaustively to obtain $I' = (G', w')$, in which there is no degree-one vertex. We then obtain $I'' = (G'', w'')$, in which the inner vertices of each maximal induced path is replaced with a path-gadget using Reduction Rule 4. It follows from Lemmas 6, 7 and 9 that this transformation is correct and can be done in linear time.

We show that G'' has at most $20g - 20$ vertices and $42g - 42$ arcs. It follows from Propositions 1 and 2 that there are at most $2g - 2$ vertices of degree at least three and $3g - 3$ maximal induced paths in $U_{G'}$. After the exhaustive application of *Reduction Rule 4*, each maximal induced path (v_0, \dots, v_{h+1}) contains at most 6 inner vertices and 14 arcs (including $v_0v_1, v_1v_0, v_hv_{h+1}, v_{h+1}v_h$). Thus, G'' contains at most $2g - 2 + 6 \cdot (3g - 3) = 20g - 20$ vertices and at most $14 \cdot (3g - 3) = 42g - 42$ arcs. Note that we count edges between vertices of degree at least three as a maximal induced paths with no inner vertex. \square

We can finally again use Theorem 3 to bound the weights and hence arrive at the following result.

Corollary 3 *MINPAC admits a polynomial-size kernel with respect to the feedback edge number.*

5 Parameterized Hardness

In this section we present several hardness results for MINPAC. To this end, we consider the decision variant of MINPAC.

POWER ASYMMETRIC CONNECTIVITY (PAC)

Input: A strongly connected graph G , arc weights $w: A(V) \rightarrow \mathbb{N}$, and a budget $k \in \mathbb{N}$.

Question: Is there a strongly connected spanning subgraph H of G , such that $\text{Cost}(V(G), A(H), w) \leq k$?

We prove that PAC remains NP-hard even if the *feedback arc number* is 1. This complements the result in Section 4, where we showed that MINPAC parameterized by the feedback edge number admits an FPT algorithm via a kernelization. Recall that the feedback arc number for a directed graph is the minimum number of arcs that have to be removed to make it a directed acyclic graph. Furthermore, we show that PAC is W[2]-hard with respect to the solution cost k . We also show that PAC cannot be solved in subexponential time in the number of vertices assuming the Exponential

Time Hypothesis (ETH) [13], which states that 3-SAT cannot be solved in $2^{o(n+m)}$ time, where n and m are the number of variables and clauses in the input formula. Summarizing we show the following.

Theorem 5 *Even if each arc weight is either one or zero and the feedback arc number is 1,*

- (i) PAC is NP-hard,
- (ii) PAC is W[2]-hard when parameterized by the solution cost k , and
- (iii) PAC is not solvable in $2^{o(n)}$ time, unless the ETH fails.

It follows from Theorem 5 (ii) that there (presumably) is no algorithm solving PAC running in $f(k) \cdot n^{O(1)}$ time. Nonetheless, a simple brute-force algorithm solves PAC in $n^{\theta(k)}$ time, certifying that PAC is in the class XP with respect to the parameter solution cost. In order to prove the claims of Theorem 5, we use a reduction from the well-studied SET COVER problem.

SET COVER

Input: A universe $U = \{u_1, \dots, u_n\}$, a set family $\mathcal{F} = \{S_1, \dots, S_m\}$ containing sets $S_i \subseteq U$, and $\ell \in \mathbb{N}$.

Question: Is there a size- ℓ set cover $\mathcal{F}' \subseteq \mathcal{F}$ (that is, $\bigcup_{S \in \mathcal{F}'} S = U$)?

SET COVER is NP-hard and W[2]-hard with respect to the solution size ℓ [11] and is not solvable in $2^{o(|U|+|\mathcal{F}|)}$ time unless the ETH fails [14].

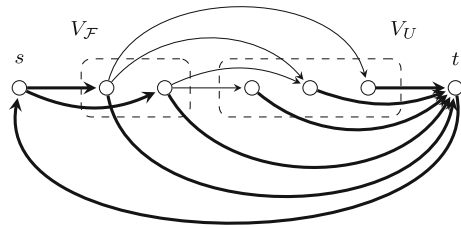
For the reduction, we use one vertex for each element and each subset and one arc to represent the membership of an element in a subset. The construction resembles the one used in MIN-POWER SYMMETRIC CONNECTIVITY [2].

Reduction 1 Given an instance $I = (U, \mathcal{F}, \ell)$ of SET COVER, we construct an instance $I' = (G, w, k = \ell)$ of PAC as follows. We introduce a vertex v_u for every $u \in U$, a vertex v_S for every $S \in \mathcal{F}$, and two additional vertices s and t . We construct a graph such that $V(G) = \{s, t\} \cup V_U \cup V_{\mathcal{F}}$ where $V_U = \{v_u \mid u \in U\}$ and $V_{\mathcal{F}} = \{v_S \mid S \in \mathcal{F}\}$. For the arcs we first add an arc ts of weight 0. We then add arcs sv_S and $v_S t$ of weight 0 for every $S \in \mathcal{F}$ and an arc $v_u t$ of weight 0 for every $u \in U$. For every $S \in \mathcal{F}$ and every $u \in S$ we finally add an arc $v_S v_u$ of weight 1.

Figure 3 illustrates the reduction to PAC. We can assume that arcs of weight zero (bold arcs in the figure) are part of the solution. The idea is that in order to obtain a strongly connected subgraph, one has to select at least one incoming arc for each vertex in V_U such that only k vertices in $V_{\mathcal{F}}$ have outgoing arcs that are selected.

To prove the W[2]-hardness, we have to verify that the given reduction is indeed a *parameterized reduction*.

Fig. 3 Illustration of Reduction 1 on a SET COVER instance with universe $U = \{1, 2, 3\}$ and set family $\mathcal{F} = \{\{2, 3\}, \{1, 2\}\}$. Bold arcs denote arcs of weight 1 and other arcs have weight 1



Definition 10 A *parameterized reduction* from a parameterized problem $\Pi \subseteq \Sigma^* \times \Sigma^*$ to a parameterized problem $\Pi' \subseteq \Sigma^* \times \Sigma^*$ is a function which maps any instance $(I, p) \in \Sigma^* \times \Sigma^*$ to another instance (I', p') such that

- (i) (I', p') can be computed from (I, p) in $f(p) \cdot |I|^{O(1)}$ time for some computable function f ,
- (ii) $p' \leq g(p)$ for some computable function g , and
- (iii) $(I, p) \in \Pi$ if and only if $(I', p') \in \Pi'$.

Lemma 10 *Reduction 1 is a parameterized reduction from SET COVER parameterized by the solution size to PAC parameterized by the solution cost.*

Proof To prove that Reduction 1 is a parameterized reduction, we verify Definition 10 (i) to (iii). Observe that Reduction 1 can be done in $O(|U| + |\mathcal{F}|)$ time, which satisfies Definition 10 (i). Definition 10 (ii) is clearly satisfied. For Definition 10 (iii), we show that I has a set cover of size at most ℓ if and only if G has a strongly connected subgraph H of cost is at most ℓ .

- (\Rightarrow) Let $\mathcal{F}' \subseteq \mathcal{F}$ be a set cover of size at most ℓ . Let $B_0 = \{s v_S, v_S t \mid S \in \mathcal{F}\} \cup \{v_u t \mid u \in U\}$ be the arcs of weight 0. We claim that $H = (V(G), B_0 \cup \{v_{S'} v_{u'} \mid S' \in \mathcal{F}', u' \in S'\})$ is a solution with cost ℓ . Since \mathcal{F}' is a set cover, there exists at least one incoming arc in H for any vertex in V_U . Thus, H is strongly connected. Since the cost for $v_{S'}$ is 1 for any $S' \in \mathcal{F}'$ and the costs for other vertices are 0, the cost of H is at most ℓ .
- (\Leftarrow) Let H be a strongly connected subgraph of cost at most ℓ . Let $\mathcal{F}' = \{S \mid \exists u. v_S v_u \in A(H)\}$. Then, \mathcal{F}' is a set cover because there is at least one incoming arc in H for any $v_u \in V_U$. Since the cost of H is at most ℓ , we have $|\mathcal{F}'| \leq \ell$. □

Now we can prove the statements of the theorem.

Proof of Theorem 5 Theorem 5 (ii) follows from Lemma 10 because SET COVER is $W[2]$ -hard when parameterized by the solution size [11]. For Theorem 5 (i), observe that Reduction 1 is a polynomial-time reduction from SET COVER and the constructed graph has a feedback arc set of size 1. For Theorem 5 (iii), observe that the constructed graph of Reduction 1 has $O(|U| + |\mathcal{F}|)$ vertices. Since SET COVER cannot be solved in $2^{o(|U|+|\mathcal{F}|)}$ time assuming ETH [14], Theorem 5 (iii) follows. □

Remark 2 We remark that having arcs of weight zero is essential for the $W[2]$ -hardness in Theorem 5 (ii): If $\min_{vu \in A(G)} w(vu) \geq 1$ for any $v \in V(G)$, then PAC is trivially FPT with respect to the solution cost (as the cost is at least n). However, even if $\min_{vu \in A(G)} w(vu) \geq 1$, PAC is still $W[2]$ -hard with respect to the *above lower bound* $k - \sum_{v \in V(G)} \min_{vu \in A(G)} w(vu)$ (this follows from a modification to Reduction 1 where every arc weight is increased by one).

6 Conclusion

We started the investigation of the parameterized complexity of MINPAC, leading to first tractability and intractability results. We remark that our algorithms run in linear time when the respective parameters are bounded. Thus we believe that our results are worthwhile for empirical experiments. There are also several theoretical challenges for future work: Can the running time of the parameterized algorithm with respect to the number c of SCCs in the obligatory subgraph be improved to single-exponential? Resolving the parameterized complexity of MINPAC with respect to the single parameter vertex cover number is another task for future work. Finally, problem variants where the solution graph is not only required to be strongly connected but needs to have at most a certain diameter might be interesting (theoretically and from an application point of view where the number of hops for communication should be limited).

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Bai, X., Kumar, S., Xuan, D., Yun, Z., Lai, T.-H.: Deploying wireless sensors to achieve both coverage and connectivity. In: Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 06), pp. 131–142. ACM (2006)
2. Bentert, M., van Bevern, R., Nichterlein, A., Niedermeier, R.: Parameterized algorithms for power-efficient connected symmetric wireless sensor networks. In: Proceedings of the 13th International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS '17), volume 10718 of LNCS, pp. 26–40. Springer (2017)
3. Bentert, M., Van Bevern, R., Fluschnik, T., Nichterlein, A., Niedermeier, R.: Polynomial-time preprocessing for weighted problems beyond additive goal functions. CoRR, arXiv:[abs/1910.00277](https://arxiv.org/abs/1910.00277) (2019)
4. Călinescu, G.: Approximate min-power strong connectivity. SIAM J. Discrete Math. **27**(3), 1527–1543 (2013)
5. Călinescu, G.: 1.61-approximation for min-power strong connectivity with two power levels. J. Comb. Optim. **31**(1), 239–259 (2016)

6. Călinescu, G., Kapoor, S., Olshevsky, A., Zelikovsky, A.: Network lifetime and power assignment in ad hoc wireless networks. In: Proceedings of the 11th Annual European Symposium on Algorithms (ESA '03), volume 2832 of LNCS, pp. 114–126. Springer (2003)
7. Carmi, P., Katz, M.J.: Power assignment in radio networks with two power levels. *Algorithmica* **47**(2), 183–201 (2007)
8. Chen, J.-J., Lu, H.-I., Kuo, T.-W., Yang, C.-Y., Pang, A.-C.: Dual power assignment for network connectivity in wireless sensor networks. In: Proceedings of the Global Telecommunications Conference (GLOBECOM T'05), pp. 5. IEEE (2005)
9. Chen, W.-T., Huang, N.-F.: The strongly connecting problem on multihop packet radio networks. *IEEE Trans. Commun.* **37**(3), 293–295 (1989)
10. Clementi, A.E.F., Penna, P., Silvestri, R.: On the power assignment problem in radio networks. *Mobile Netw. Appl.* **9**(2), 125–140 (2004)
11. Downey, R.G., Fellows, M.R.: Fundamentals of parameterized complexity. Springer (2013)
12. Fellows, M.R., Gaspers, S., Rosamond, F.A.: Parameterizing by the number of numbers. *Theor. Comput. Syst.* **50**(4), 675–693 (2012)
13. Impagliazzo, R., Paturi, R.: On the complexity of k -SAT. *J. Comput. Syst. Sci.* **62**(2), 21 (2001)
14. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.* **63**(4), 512–530 (2001)
15. Iyengar, R., Kar, K., Banerjee, S.: Low-coordination wake-up algorithms for multiple connected-covered topologies in sensor nets. *Int. J. Sens. Netw.* **5**(1), 33–47 (2009)
16. Rong, Y., Choi, H., Choi, H.-A.: Dual power management for network connectivity in wireless sensor networks. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS '04), IEEE (2004)
17. Sorge, M., Van Bevern, R., Niedermeier, R., Weller, M.: A new view on rural postman based on eulerian extension and matching. *J. Discrete Algorithms* **16**, 12–33 (2012)
18. Zhang, H., Hou, J.C.: Maintaining sensing coverage and connectivity in large sensor networks. In: Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-To-Peer Networks, pp. 453–474. CRC Press / Taylor & Francis (2005)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.