



# An Efficient Algorithm for Power Dominating Set

Thomas Bläsius  

Karlsruhe Institute of Technology (KIT), Germany

Max Göttlicher  

Karlsruhe Institute of Technology (KIT), Germany

---

## Abstract

The problem POWER DOMINATING SET (PDS) is motivated by the placement of phasor measurement units to monitor electrical networks. It asks for a minimum set of vertices in a graph that observes all remaining vertices by exhaustively applying two observation rules. Our contribution is twofold. First, we determine the parameterized complexity of PDS by proving it is  $W[P]$ -complete when parameterized with respect to the solution size. We note that it was only known to be  $W[2]$ -hard before. Our second and main contribution is a new algorithm for PDS that efficiently solves practical instances.

Our algorithm consists of two complementary parts. The first is a set of reduction rules for PDS that can also be used in conjunction with previously existing algorithms. The second is an algorithm for solving the remaining kernel based on the implicit hitting set approach. Our evaluation on a set of power grid instances from the literature shows that our solver outperforms previous state-of-the-art solvers for PDS by more than one order of magnitude on average. Furthermore, our algorithm can solve previously unsolved instances of continental scale within a few minutes.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms

**Keywords and phrases** Power Dominating Set, Implicit Hitting Set, Parameterized Complexity, Reduction Rules

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2023.21

**Related Version** *Full Version:* <https://arxiv.org/abs/2306.09870> [4]

**Supplementary Material** *Software (Source Code):* <https://gitlab.com/Aldorn/pds-code>  
archived at `swh:1:rev:121131be58bc1b6f91a4863bd01fe7cbd0439ab9`

**Funding** *Max Göttlicher:* German Research Foundation (DFG) as part of the Research Training Group GRK 2153: Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation.

## 1 Introduction

Monitoring power voltages and currents in electric grids is vital for maintaining their stability and for cost-effective operation. The sensors required to obtain high-resolution measurements, so-called phasor measurement units, are expensive pieces of equipment. The goal to place as few of those sensors as possible to minimize cost is called the POWER DOMINATING SET problem (PDS). It was first posed by Mili, Baldwin and Adapa. [16] and formalized by Baldwin et al. [2]. In its basic form, the problem asks whether the graph of a power grid can be observed by exhaustively applying two observation rules [7]: First, every sensor observes its vertex and all neighbors. Secondly, if a vertex is observed and has only one unobserved neighbor, that neighbor becomes observed, too.

PDS is unfortunately NP-complete [7, 11, 14], i.e., we cannot expect there to be an algorithm that performs reasonably on all inputs. Moreover, the problem remains hard for a wide range of different graph classes [7, 8, 11, 14, 20, 9, 15]. In terms of parameterized complexity, PDS is known to be  $W[2]$ -hard [9] when parameterized by solution size.



© Thomas Bläsius and Max Göttlicher;  
licensed under Creative Commons License CC-BY 4.0  
31st Annual European Symposium on Algorithms (ESA 2023).

Editors: Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman; Article No. 21;  
pp. 21:1–21:15



Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On the positive side, various approaches for solving PDS have been proposed. Theoretic results show that PDS can be solved in linear time in graphs with fixed tree-width [14, 9]. However, those algorithms have, to the best of our knowledge, never been implemented and are probably infeasible in practice due to their bad scaling with respect to the tree-width. Several exponential-time algorithms have been presented [7, 3] but those algorithms have not been implemented and evaluated either.

Practically feasible approaches using an MILP formulation have been proposed by Aazami [1]. This formulation was later improved upon by Brimkov, Mikesell, and Smith [6] and most recently Jovanovic and Voss [13]. A different approach is to reduce PDS to the hitting set problem [5, 18]. This approach is based on the observation that one can determine so-called *forts*, which are subsets of vertices that prevent propagation if none of them is selected. A set of vertices is a valid solution for PDS if and only if at least one vertex is selected for each fort, i.e., if it is a hitting set for the collection of all forts. Graphs may contain an exponential number of forts, so this hitting set instance is not computed explicitly. Instead, one can use the so-called *implicit hitting set* approach, where one starts with a subset of all forts, computes a hitting set for this subset, and then validates whether this is already a solution for the PDS instance. If not, one obtains at least one new fort that can be added to the set of considered forts. This is iterated until a solution is found. This implicit hitting set approach has been used for other problems, e.g., for MAXSAT [17] and TQBF [12]. For PDS, it has been introduced by Bozeman et al. [5]. The strategy of finding forts has been later improved by Smith and Hicks [18], providing the current state-of-the-art for solving PDS in practice.

Our contribution is threefold. First, we study the parameterized complexity of PDS parameterized by the solution size. Though it is known to be  $W[2]$ -hard [9], it was unknown whether PDS is also contained in  $W[2]$ . We show that PDS is  $W[P]$ -complete via a reduction from WEIGHTED CIRCUIT SATISFIABILITY for circuits of arbitrary weft. This completely determines its parameterized complexity and in particular shows that it is not in  $W[2]$  unless  $W[2] = W[P]$ . In our second contribution, we propose a set of reduction rules for pre-processing PDS instances. Our reduction rules aim to produce equivalent instances that are smaller and annotated with partial decisions, i.e., some vertices are marked as selected or as forbidden-to-select. Though these annotations lead to a more general problem than the basic PDS, we show that existing approaches for solving PDS can be easily adapted to solve the annotated instances. Moreover, we show that their performance greatly benefits from our reduction rules. Finally, our third contribution is an improved heuristic for finding forts for the implicit hitting set formulation. This improved heuristic together with our reduction rules beats the current state of the art solvers by more than one order of magnitude. Moreover, our approach can solve previously unsolved instances of continental scale.

The remainder of this paper is organized as follows. Section 2 provides an overview of the basic concepts and notation used throughout this paper. In Section 3, we show that PDS is  $W[P]$ -complete. Our reduction rules and the heuristic for extending the hitting set instance are presented in Section 4. Section 5 contains our experimental evaluation of the new method using a set of benchmark instances.

## 2 Preliminaries

**Graphs and Neighborhoods.** Let  $G = (V, E)$  be an undirected graph with vertices  $V$  and edges  $E$ . For  $v \in V$ , let  $N(v) = \{u \in V \mid uv \in E\}$  be the *open neighborhood* of  $v$ . Similarly,  $N[v] = N(v) \cup \{v\}$  is the *closed neighborhood* of  $v$ . Given a set  $S \subseteq V$  we denote by  $N(S)$  and  $N[S]$  the union of all open and closed neighborhoods of the vertices in  $S$ .

**Power Dominating Set.** For a given graph  $G$ , the problem *POWER DOMINATING SET (PDS)* is to find a minimum vertex set  $S \subseteq V$  of *selected* vertices such that all vertices of the graph are observed. We call such set a *power dominating set*. The size of a minimum power dominating set of a given graph  $G$  is called the *power dominating number*  $\gamma_P(G)$ . Whether a vertex is *observed* is determined by the following rules, which are applied iteratively. We note that for the second rule, vertices can be marked as *propagating*, i.e., the input of PDS is not just a graph but a graph together with a set of propagating vertices.

**Domination rule.** A vertex is observed if it is in the closed neighborhood of a selected vertex.

**Propagation rule.** Let  $u \in V$  be a propagating vertex. If  $u$  is observed and  $v \in N(u)$  is the only neighbor of  $u$  that is not yet observed, then  $v$  becomes observed<sup>1</sup>. If the propagation rule is applied to an observed vertex  $u$ , we say it *propagates* its observation status.

The special case where we have no propagating vertices yields the well known *DOMINATING SET (DS)* problem. Moreover, the special case where all vertices are propagating is called *SIMPLE-PDS*. In addition to the above *DOMINATING SET* variants, we also consider the extension variant *DOMINATING SET EXTENSION*. For *DS-EXTENSION*, the input consists of the graph  $G = (V, E)$ , a set  $X \subseteq V$  of *pre-selected* and a set  $Y$  of *excluded* vertices; vertices in  $V \setminus X \setminus Y$  are called *undecided*. *DS-EXTENSION* asks whether there exists a solution  $S \subseteq V$  such that  $S$  includes all selected and excludes all excluded vertices, i.e.,  $X \subseteq S$  and  $Y \cap S = \emptyset$ . The problems *PDS-EXTENSION* and *SIMPLE-IPDS-EXTENSION* are defined analogously.

**Hitting Set.** Let  $V$  be a set and let  $\mathcal{F} \subseteq 2^V$  be a family of subsets. A set  $H \subseteq V$  is a *hitting set* if it *hits* every set  $F \in \mathcal{F}$ , i.e.,  $F \cap H \neq \emptyset$  for all  $F \in \mathcal{F}$ . The problem *HITTING SET* is to find a hitting set of minimum size. Note that the extension variant of *HITTING SET* reduces to an instance of *HITTING SET* itself, as one can simply remove excluded elements and remove the sets containing pre-selected elements.

### 3 Power Dominating Set is $W[P]$ -Complete

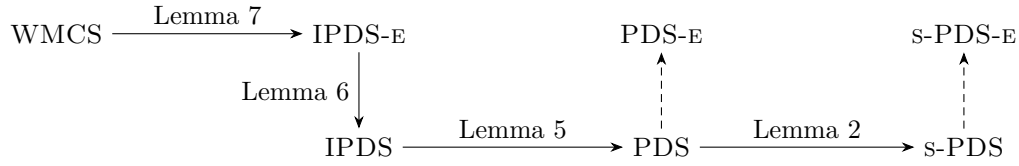
We prove  $W[P]$ -completeness via a chain of parameterized reductions from the *WEIGHTED MONOTONE CIRCUIT SATISFIABILITY (WMCS)* problem. *WMCS* has a monotone Boolean circuit as input and asks whether it can be satisfied by setting at most  $k$  inputs to *TRUE*, where  $k$  is the parameter. We assume familiarity with the  $W$ -hierarchy and parameterized reductions. We start by introducing a variant of the PDS problem that we use as an intermediate problem in our chain of reductions. For brevity, we only sketch an outline of the proof in this section; for the full proof see [4].

The input of the problem *IMPLICATING POWER DOMINATING SET (IPDS)* is an instance of PDS with the following additional information. First, edges of the graph  $G = (V, E)$  can be marked as *booster edges*. Secondly, we are given a set of *implication arcs*  $A \subseteq V \times V$ . We interpret  $A$  as a set of directed edges on  $V$  but perceive them as separate from the graph  $G$ , i.e., they do not affect the neighborhood. In addition to the domination and propagation rule introduced in Section 2, we define the following to observation rules.

**Booster rule.** Let  $uv \in E$  be a booster edge. If  $u$  is observed, then  $v$  becomes observed and vice versa.

**Implication rule.** Let  $(u, v) \in A$  be an implication arc and let  $u$  be observed. Then  $v$  also becomes observed.

<sup>1</sup> The propagation rule is motivated by Kirchoff's law and Ohm's law in electric networks. Propagating vertices are also called *zero-injection vertices*. In electric networks, they refer to buses in substations that have no attached loads.



■ **Figure 1** Reduction steps to show that PDS and its variants are  $W[P]$  hard. The solid arrows indicate our parameterized reductions described in this section. The hardness of the extension problems follows from the hardness of their basic problem, indicated by the dashed arrows.

We note that IPDS is a generalization of PDS in the sense that every PDS instances is an instance of IPDS with no booster edges and an empty set of implication arcs. The extension variant IPDS-EXTENSION is defined analogously to PDS-EXTENSION. Proving containment of IPDS-EXTENSION in  $W[P]$  is straight forward by giving an appropriate non-deterministic Turing machine. We note that the analogous statement has been observed before for PDS by Kneis et al. [14]. Note that this implies containment in  $W[P]$  for all other problem variants we defined.

► **Lemma 1.** *IMPLICATING POWER DOMINATING SET EXTENSION is in  $W[P]$ .*

As all other variants of the power dominating set problem we consider are special cases of IPDS-EXTENSION, this also proves containment in  $W[P]$  for the other variants.

**Power Dominating Set to Simple Power Dominating Set.** Our chain of reductions to prove  $W[P]$ -hardness is illustrated in Figure 1. We start with the reduction from PDS to SIMPLE PDS, which is similar to the proof of  $W[2]$  hardness of PDS [14, 9]. The core idea is to simulate a non-propagating vertex with a propagating vertex with an additional leaf attached.

► **Lemma 2.** *There is a parameterized reduction from POWER DOMINATING SET to SIMPLE POWER DOMINATING SET.*

**Proof Sketch.** Similar to the proof of  $W[2]$  hardness of POWER DOMINATING SET [14, 9], we can attach a leaf to each non-propagating vertex. Selecting the leaf as part of a solution is never optimal: one can instead choose its neighbor. Then, a vertex with an attached leaf can never propagate to any vertex except the leaf. ◀

**Implicating Power Dominating Set to Power Dominating Set.** The reduction from IPDS to PDS, works in two steps. First, we show that we can eliminate implicating arcs by replacing each of them with the small gadget show in Figure 2a. Using another gadget, we eliminate booster edges in a similar way, yielding the reduction.

► **Lemma 3.** *Every instance of IMPLICATING POWER DOMINATING SET can be reduced to an equivalent instance with no implication arcs without changing the parameter.*

**Proof Sketch.** Given an IPDS-instance  $G$ , we replace all implication arcs  $a = (x, y)$  with the gadget depicted in Figure 2a. To see why the gadget works as desired, consider the implication gadget and first assume that  $x$  is observed. By applying the booster and the propagation rules, one can verify that all vertices introduced in the gadget and  $y$  become observed. Conversely, if only  $y$  is observed,  $c$  becomes observed by the booster rule but cannot propagate due to its two unobserved neighbors. Thus, the gadget mimics the behavior of an implication arc. ◀



(a) Gadget simulating an implication arc from  $x$  to  $y$  using booster edges (marked with green diamonds).

(b) A gadget for simulating an **and** gate. The bottom  $\wedge$ -node is observed iff all input nodes are observed.

■ **Figure 2** Gadgets for implication arcs and **and** gates.

Our gadget for simulating booster edges requires adding a globally unique non-propagating vertex  $b$  to which all such gadgets are connected. The gadget in turn replaces a booster edge between  $x$  and  $y$  with a new vertex  $v_{xy}$  which is connected to  $x$ ,  $y$  and  $b$ . We enforce that  $b$  is selected by attaching a leaf.

► **Lemma 4.** *Every IPDS-instance with booster edges can be reduced to an equivalent instance without booster edges.*

**Proof Sketch.** One can verify that the booster gadget works as intended as follows: by the domination rule,  $b$  observes  $v_{xy}$ . If now either of  $x$  or  $y$  becomes observed, we can apply the propagation rule on  $v_{xy}$ , observing the other. Note that the inserted vertex  $b$  is the same for all booster gadgets. ◀

► **Lemma 5.** *There is a parameterized reduction from IMPLICATING POWER DOMINATING SET to POWER DOMINATING SET.*

**Extension to Non-Extension (for IPDS).** For the IPDS-EXTENSION to IPDS, the core difficulty comes from enforcing the excluded vertices to not be selected. We already saw how to enforce the selection of vertices in the construction of the booster gadget. The basic idea for excluding vertices from the solution is to make the selection of certain vertices very expensive.

► **Lemma 6.** *There is a parameterized reduction from IMPLICATING POWER DOMINATING SET EXTENSION to IMPLICATING POWER DOMINATING SET.*

**Proof Sketch (Vertex Exclusion).** Let  $G = (V, E)$  be an IPDS-EXTENSION-instance where the vertices  $Y \subseteq V$  are excluded from a solution. We construct an equivalent instance without excluded vertices. We achieve this by creating a new graph  $G'$ , consisting of  $|V| + 1$  copies of  $G$  and a clique  $C$  of  $|V \setminus Y|$  non-propagating vertices. Each copy  $G^{(i)}$  has a fresh set of vertices representing the vertices in  $G$  and there is an edge between two vertices in a copy if there is an edge between their counterparts in  $G$ . There are no edges between vertices in different copies. The vertices in  $C$  represent the vertices not excluded from a solution. For each vertex in  $C$  we add edges to the closed neighborhoods of their counterparts in each of the copies.

Given a power dominating set  $S$  of  $G$ , selecting the corresponding vertices in  $C$  yields a power dominating set for  $G'$ . Conversely, the construction ensures that vertices selected in one of the copies never observe any vertices in another copy. Thus, and because all copies are identical, if a minimum power dominating set contains a vertex in one of the copies, it must contain a vertex in all other copies, too. Hence, a minimum power dominating set of size less than  $|V|$  cannot contain any vertices outside  $C$ . As the vertices in  $C$  correspond to the selectable vertices in  $G$ , we obtain a power dominating set of  $G$ . ◀

**WMCS to IPDS-Extension.** Finally, for the reduction from WMCS to IPDS-EXTENSION, the core idea is to replace the arcs in the directed acyclic graph describing the circuit with implication arcs and to model **and**-gates as show in Figure 2b.

► **Lemma 7.** *There is a parameterized reduction from WEIGHTED MONOTONE CIRCUIT SATISFIABILITY to IMPLICATING POWER DOMINATING SET EXTENSION.*

**Proof Sketch.** We construct an equivalent IPDS-EXTENSION-instance  $G$  from a given monotone circuit  $C = (V, E)$  as follows. In this construction, we interpret TRUE values in the circuit as a node being observed. We thus interpret all directed edges in the circuit as implication arcs and add further implication arcs from the output to every input. The input nodes become propagating vertices, all other vertices in  $G$  are non-propagating. The **or**-gates are simulated by the implication rule without further adaptation.

To simulate the **and**-gates, we use the gadget in Figure 2b. We replace every **and**-gate  $v$  by two new connected vertices  $x$  and  $y$  where all outgoing edges of  $v$  are instead outgoing implication arcs of  $y$ . For every incoming edge of  $v$  from  $u$ , we place a new proxy vertex  $x_u$  and add an edge  $x_u x$  and an implication arc  $(u, x_u)$ . Then, the gate output  $y$  becomes observed by the propagation rule from  $x$  if and only if all proxy vertices  $x_u$  are observed, i.e. if all inputs of the **and**-gate are TRUE.

We need the implication arcs back from the output to the inputs to ensure all vertices become observed if the corresponding truth assignment is satisfying. ◀

► **Corollary 8.** *POWER DOMINATING SET is  $W[P]$  complete.*

## 4 Solving Power Dominating Set

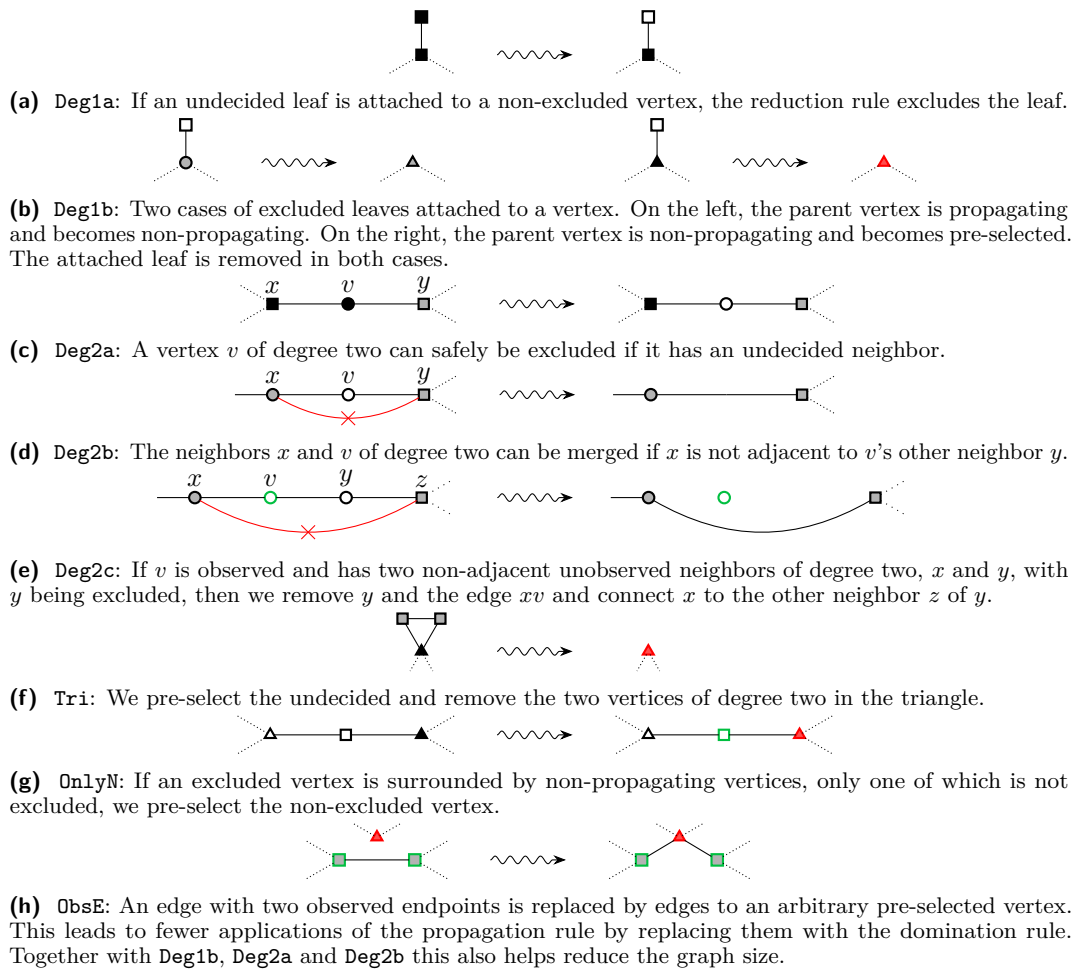
In this section, we give an algorithm for solving PDS-EXTENSION. Our algorithm consists of different phases. In the first phase, we apply the reduction rules described in Section 4.1. Each rule either shrinks the graph or decides for a vertex that it should be pre-selected or excluded. We prove that the rules are safe, i.e., they yield equivalent instances. Afterwards, in Section 4.2 we split the instance into several components that can be solved independently. Finally, each of the subinstances is solved exactly using the implicit hitting set approach [5] with our improved strategy for finding new sets that need to be hit; see Section 4.3.

We note that these phases are somewhat modular in the sense that one could easily add further reduction rules or that one can replace the algorithm for solving the kernel in the final step. In our experiments in Section 5, we also use an MILP for this step. This MILP formulation is based on the formulation for PDS by Jovanovic and Voss [13] with the adjustments from [4, Appendix A]. Moreover, instead of solving the subinstances optimally, one can instead use a heuristic solver. The preceding application of our reduction rules and splitting into subinstances then helps to find better solutions rather than improving the running time of exact solvers. This is used in our experiments to find upper bounds on the power domination number before we have an exact solution.

### 4.1 Reduction Rules

Many of our reduction rules are local in the sense that they transform one substructure into a different substructure. Most of our local reduction rules are illustrated in Figure 3. In the following, we specify additional reduction rules that are either non-local or otherwise difficult to illustrate. For more details and proofs of safeness, see [4, Appendix B].





■ **Figure 3** Illustrated overview of the local reduction rules. Round vertices ● are propagating, triangular vertices ▲ are non-propagating, square vertices ■ may be propagating or non-propagating. Hollow vertices □ are excluded from a solution, vertices filled black ■ are undecided, red triangular vertices ▲ are pre-selected. Vertices filled gray ■ may be undecided, excluded or pre-selected. Green vertices □ are observed but not pre-selected. The absence of an edge is indicated in red  $\rightarrow\times$ .

► **Reduction ObsNP (Observed Non-Propagating).** *Let  $v$  be an observed, non-propagating and excluded vertex. Then delete  $v$ .*

► **Reduction Iso1 (Isolated).** *Let  $v$  be an undecided isolated vertex. Then pre-select  $v$ .*

For the next two reduction rules, we introduce the concept of *observation neighborhood*. For a set of vertices  $U \subseteq V$  the observation neighborhood is the set of vertices that is observed when selecting  $U$  in addition to all pre-selected vertices and applying the observation rules exhaustively. For convenience, we define the observation neighborhood of a single vertex  $v$  to be the observation neighborhood of the single element set  $\{v\}$ .

► **Reduction Dom (Domination).** *If the closed neighborhood of some undecided vertex  $w$  is contained in the observation neighborhood of some other undecided vertex  $v$ , exclude  $w$ .*

► **Reduction NecN (Necessary Node).** *Let  $v$  be an undecided vertex. If the observation neighborhood of all undecided vertices except  $v$  does not contain all vertices in  $G$ , select  $v$ .*

We note that Binkle-Raible and Fernau [3] already introduced reduction rules for their exponential-time algorithm. We do not use them in our algorithm as they are not generally applicable but rather require a specific situation. The only exception [3, “isolated”] is superseded by our reduction rules.

#### 4.1.1 Order of Application

In a first step, use a depth first search and process the vertices in post-order to apply the rules `Deg1a`, `Deg1b`, and `Deg2a`. The order in which we process the vertices is important here, as it makes sure that attached paths are properly reduced. This is only relevant for this first application of the reduction rules and in later applications, we process the vertices and edges in arbitrary order. After this initial application, we iterate the following three steps until no reduction rules can be applied. (i) Iterate the application of the local reduction rules (`Deg1a`, `Deg1b`, `Deg2a`, `Deg2b`, `Tri`, `Deg2c`, `OnlyN`, `ObsNP`, `ObsE`) until no local reduction rule is applicable. (ii) Apply the non-local rule `Dom`. (iii) Apply the non-local rule `NecN`.

We note that applying `Dom` once to all vertices is exhaustive in the sense that it cannot be applied again immediately afterwards. It can, however, become applicable again after rerunning the other reduction rules. The same is true for `NecN`.

Our reasoning for this sequence of application is that the local rules are more efficient than the non-local ones. Thus, we first apply the cheap rules exhaustively before resorting to the expensive ones. Preliminary experiments showed that further tweaking the order of application has only minor effect on the kernel size and run time.

#### 4.1.2 Implementation Notes

The naïve implementation of the reduction rules can be very slow, in particular for the non-local rules. The costly operation in those rules is the computation of the observation neighborhood. We thus use a specialized data structure that allows us to pre-select and deselect vertices in arbitrary order. Each time we pre-select a vertex, we also update the observed vertices and keep track of which vertex propagates to which other vertex. For de-selecting vertices, we only mark vertices as unobserved, that were directly or indirectly observed by the deselected vertex. Being able to select and deselect arbitrary vertices allows a straightforward implementation of the non-local rules.

### 4.2 Split into Subinstances

While the propagation rule may have non-local effects within the whole graph, propagation cannot pass through selected vertices. This is formalized by the following theorem.

► **Theorem 9.** *Let  $G = (V, E)$  be the graph with pre-selected vertices  $X \subseteq V$  and let  $C_1, \dots, C_\ell \subseteq V$  be the vertices in the connected components of the sub-graph of  $G$  induced by  $V \setminus X$ . Further let  $S_1, \dots, S_\ell$  be minimum power dominating sets of the subgraphs induced by  $N[C_1], \dots, N[C_\ell]$ . Then  $S = S_1 \cup \dots \cup S_\ell$  is a minimum power dominating set of  $G$ .*

These sub-problems can be identified in linear time using a depth-first search restarting at unexplored non-active nodes while ignoring outgoing edges of active nodes.

### 4.3 Solving the Kernel Via Implicit Hitting Set

We briefly describe the implicit hitting set approach. Compared to how Bozeman et al. [5] introduced it, we allow non-propagating vertices. However, this does not change any of the proofs and thus the approach directly translates to this slightly more general setting.



In a graph  $G$ , a *fort* is a non-empty subset of vertices  $F \subseteq V(G)$  such that no propagating vertex outside  $F$  is adjacent to precisely one vertex in  $F$ . A power dominating set must be a hitting set of the family of all *fort neighborhoods* in  $G$ , i.e., if  $F$  is a fort and  $S$  is a power dominating set, then  $N[F] \cap S \neq \emptyset$ . Conversely, if a hitting set for a family  $\mathcal{F}$  of fort neighborhoods is not a power dominating set, then one can find an additional fort of  $G$  whose neighborhood is not in  $\mathcal{F}$ .

This yields the following algorithm. Start with some set  $\mathcal{F}$  of fort neighborhoods. Compute a minimum hitting set  $H$  for  $\mathcal{F}$ . If  $H$  is already a power dominating set, we have found the optimum. Otherwise, we construct at least one new fort neighborhood and add it to  $\mathcal{F}$ .

One core ingredient of this approach is the choice of which fort neighborhoods to add to  $\mathcal{F}$ . Previous approaches [18, 5] aimed at finding forts or fort neighborhoods that are as small as possible. The reasoning behind this is that the set of all fort neighborhoods can be exponentially large (even when restricted to those that are minimal with respect to inclusion) and thus it makes sense to add sets that are as restrictive as possible, hoping that only few sets suffice before the HITTING SET solution yields a PDS solution. However, finding forts of minimum size or minimum size fort neighborhoods is difficult while just finding any fort is easy. Moreover, if we add only few forts in every step, we have to potentially solve more HITTING SET instances. We thus propose to instead find multiple forts at once and to add them all to the HITTING SET instance. Our method of finding forts is based on the following lemma.

► **Lemma 10.** *Let  $G = (V, E)$  be a graph and let  $S \subseteq V$  be a set of selected vertices. Let further be  $R$  the set of vertices observed by exhaustive application of the observation rules with respect to  $S$ . Then the set of unobserved vertices  $V \setminus R$  is a fort.*

**Proof.** Assume  $V \setminus R$  is not a fort. Then there exists a propagating vertex  $v$  in  $R$  that is adjacent to precisely one vertex  $w$  in  $V \setminus R$ , i.e.,  $v$  has precisely one unobserved neighbor  $w$ . This contradicts the exhaustive application of the propagation rule and thus the set of unobserved vertices is a fort. ◀

By Lemma 10, whenever we have a candidate solution that does not yet observe all vertices, we obtain a new fort and can add its neighborhood to the HITTING SET instance  $\mathcal{F}$ . For the new forts, we have two objectives. First, we want the new fort neighborhood to actually provide new restrictions, i.e., it should not be already hit by the minimum hitting set  $H$  of  $\mathcal{F}$ . This is achieved by making sure that the candidate solution  $S$  is a superset of the hitting set  $H$ . Secondly, we want the resulting forts (i.e., the number of unobserved vertices) to be small. We achieve this heuristically by greedily considering large candidate solutions.

Specifically, we choose candidate solutions as follows. Recall, that we consider the extension problem, i.e., we have sets  $X$  and  $Y$  of pre-selected and excluded vertices, respectively. Moreover, let  $H$  be a minimum hitting set of the current set of fort neighborhoods. Then  $V$  is partitioned into the four sets  $X$ ,  $Y$ ,  $H$ , and  $U = V \setminus H \setminus X \setminus Y$ . Each candidate solution  $S$  we consider is a superset of  $H \cup X$  and a subset of  $H \cup X \cup U$ . We randomly order the vertices in  $U = \{u_1, \dots, u_\ell\}$  and define a sequence  $U_0, \dots, U_\ell \subseteq U$ . We then consider the candidate solutions  $S_i = H \cup X \cup U_i$  for  $0 \leq i \leq \ell$ . As we want to consider large candidate solutions, we start with  $U_0 = U$ , which clearly yields a solution as the instance would be invalid otherwise. We obtain the subset  $U_i$  from  $U_{i-1}$  as follows. If  $S_{i-1}$  was a solution, i.e., there were no unobserved vertices, then  $U_i = U_{i-1} \setminus \{u_i\}$ . Otherwise,  $U_i = U_{i-1} \cup \{u_{i-1}\} \setminus \{u_i\}$ . Note that this makes sure that each candidate solution  $S_i$  we consider is either a solution or barley not a solution as  $S_i \cup \{u_i\}$  is a solution.

This gives us at least one and up to  $\ell$  new fort neighborhoods. These are not directly added to the set  $\mathcal{F}$ . Instead, we first apply a simple local search to make sure that each fort is minimal with respect to inclusion. To this end, we iteratively re-select vertices from  $U$  that had been removed before and check whether this still results in a non-empty fort.

We note that we only add sets to  $\mathcal{F}$ . Thus, we have to solve a sequence of increasing HITTING SET instances as a subroutine. To improve the performance of this, one can use lower bounds achieved in earlier iterations as lower bounds for later iterations (HITTING SET is monotone with respect to the addition of sets).

## 5 Experiments

The goal of this section is threefold. First, we evaluate the performance of our algorithm in comparison to two previous state-of-the-art approaches. Secondly, we give a more detailed view on the performance by analyzing how the upper and lower bounds found by the different algorithms converge to the optimal solution. Thirdly, we evaluate the impact of the different reduction rules.

**Experiment Setup.** We implemented our algorithm in C++ 20 and compiled it with clang 15.0.1 with the `-O3` optimization flag. Our source code will be made publicly available on publication. For the comparison with the previous state-of-the-art, we use the MILP formulation approach by Jovanovic and Voss [13]. In the following, we refer to this algorithm with MILP. The second solver by Smith and Hicks [18] and is based on the implicit hitting set approach. Unfortunately, their code is not publicly available, and the paper does not specify all implementation details. To make a fair (or rather generous) comparison, we initialized their set of forts with our fort heuristic, which, as far as we can judge, leads to better results than reported in the original publication [18]. We refer to this algorithm as MFN (abbreviation for *minimum fort neighborhood*). For the implicit hitting set approaches, we use an MILP formulation to solve the HITTING SET instances. All MILP instances are solved using Gurobi 9.5.2 [10].

The experiments were run on a machine running Ubuntu 22.04 with Linux 5.15. The machine has two Intel®Xeon®Gold 6144 CPUs clocked at 3.5 GHz with 8 single-thread cores and 192 GB of RAM.

We used a collection of instances shipped with pandapower [19]. We further use the Eastern, Western, Texas and US instances from the powersimdata set<sup>2</sup> [21] based on the US electric grids. We interpret the power grids as graphs where buses are vertices and power lines and transformers are edges. Buses without attached loads or generators yield propagating vertices. For experiments on the pandapower instances, we used a timeout of 2 h and repeated each experiment 5 times. On the powersimdata instances, we used a timeout of 10 h and only repeated the experiments using our solver. For repeated experiments, we report the median result.

**Performance Comparison.** We compare the performance of our solver to the MILP and MFN approach, each with and without preprocessing by the reduction rules. To assess the performance of our approach with reduction rules, we compute the speedup compared to the lowest run time of the previous approaches without reduction rules.

---

<sup>2</sup> <https://github.com/Breakthrough-Energy/PowerSimData>

■ **Table 1** Run times of different combinations of PDS solvers and reduction rules on the pandapower data set. Note that  $|S|$  differs from the results reported in other literature. This is to be expected because we include non-propagating vertices from the input. Further observe that some run times are given in milli- or microseconds.

instance	$ S $ #	MILP <sup>a)</sup> s	MILP+R <sup>a)</sup> s	MFN <sup>a)</sup> s	MFN+R <sup>a)</sup> s	Ours s	Ours+R s	Speedup
4gs	2	1.41m	2.75m	2.34m	1.89m	<b>513<math>\mu</math></b>	672 $\mu$	2.1
5	2	1.06m	840 $\mu$	2.24m	1.60m	265 $\mu$	<b>254<math>\mu</math></b>	4.2
6ww	1	1.19m	10 $\mu$	836 $\mu$	<b>6<math>\mu</math></b>	156 $\mu$	8 $\mu$	104.6
9	2	2.61m	1.23m	4.98m	2.14m	544 $\mu$	<b>181<math>\mu</math></b>	14.4
11_iwamoto	2	4.59m	23 $\mu$	3.49m	21 $\mu$	613 $\mu$	<b>17<math>\mu</math></b>	205.5
14	3	1.79m	1.38m	3.82m	2.01m	<b>487<math>\mu</math></b>	522 $\mu$	3.4
24_ieee_rts	6	4.19m	4.87m	5.59m	5.50m	1.30m	<b>802<math>\mu</math></b>	5.2
30	6	2.93m	52 $\mu$	5.01m	49 $\mu$	622 $\mu$	<b>45<math>\mu</math></b>	65.1
ieee30	6	3.40m	52 $\mu$	6.53m	<b>43<math>\mu</math></b>	669 $\mu$	46 $\mu$	73.9
33bw	11	1.81m	50 $\mu$	8.39m	<b>44<math>\mu</math></b>	551 $\mu$	144 $\mu$	12.6
39	9	6.60m	112 $\mu$	11.95m	<b>104<math>\mu</math></b>	1.76m	116 $\mu$	56.9
57	12	9.74m	9.78m	24.94m	13.83m	<b>1.71m</b>	1.75m	5.6
89pegase	13	22.73m	190 $\mu$	12.88m	<b>169<math>\mu</math></b>	1.74m	288 $\mu$	44.7
118	29	14.26m	12.49m	59.64m	39.27m	7.63m	<b>4.41m</b>	3.2
145	18	123.65m	275 $\mu$	19.65m	<b>269<math>\mu</math></b>	2.46m	274 $\mu$	71.7
illinois200	39	20.57m	<b>276<math>\mu</math></b>	162.35m	464 $\mu$	4.56m	412 $\mu$	49.9
300	72	29.66m	2.75m	218.41m	4.70m	7.57m	<b>1.44m</b>	20.6
1354pegase	311	105.77m	2.61m	2.10	2.93m	19.37m	<b>1.89m</b>	56.0
1888rte	375	554.12m	6.82m	6.26	5.80m	40.36m	<b>3.10m</b>	178.7
2848rte	585	603.93m	7.55m	15.77	6.20m	52.60m	<b>4.94m</b>	122.3
2869pegase	612	1.21	221.46m	16.12	1.53	165.41m	<b>96.05m</b>	12.6
3120sp	768	1.36	270.73m	40.29	1.60	310.77m	<b>113.29m</b>	12.0
6470rte	1303	2.94	42.65m	88.96	68.06m	241.73m	<b>27.58m</b>	106.7
6495rte	1314	3.52	45.85m	89.29	91.92m	256.41m	<b>27.12m</b>	129.7
6515rte	1315	3.89	45.88m	89.75	98.39m	231.60m	<b>27.83m</b>	139.9
9241pegase	2010	5.71	1.31	212.93	13.47	1.36	<b>660.18m</b>	8.6

a) numbers here were obtained from our interpretation of the respective approach

Table 1 shows the run times of the solvers on the smaller pandapower instances. Preprocessing significantly reduced the running times of all solvers in most cases, especially for the larger instances. We found that our solver with reduction rules performs best in 17 out of 26 instances. In 3 further instances, our solver without reduction rules performed best and the version with reduction rules came second. In particular, our solver performs best on all instances of more than 300 vertices and all instances that took more than one millisecond to solve for any solvers. Even in the six instances where our solver was not the fastest, the other approaches could only compete when combined with the reduction rules.

For the larger powersimdata instances, neither MILP nor MFN were able to compute an optimal solution without using our reduction rules. Thus, for these instances, we only compare our solver with MFN+R and MILP+R. Table 2 shows the results. Observe that for **Eastern**, our algorithm finished after 16 min while MFN did not finish after more than 6 h, with a lower bound that was still more than 100 vertices below the optimal solution. Further observe that the number of fort neighborhoods  $|\mathcal{F}|$  is lower for MFN. This is to be expected as minimizing their number is basically the main goal of MFN when finding new fort neighborhoods. However, this clearly does not show any benefit in the resulting run time.

■ **Table 2** Comparison between our algorithm and MFN on the larger powersimdata US instances preprocessed with our reduction rules.  $n$  is the number of vertices,  $|Z|$  is the number of non-propagating vertices and  $|\mathcal{F}|$  is the size of the arising hitting set instance. For the solvers, we report the power dominating number  $\gamma_P$  (or the best found lower bound) as well as the number of fort neighborhoods  $\mathcal{F}$  and the run time.

Instance	Input		Our Solver			MFN+R			MILP+R	
	$n$	$ Z $	$\gamma_P$	$ \mathcal{F} $	$t$ (s)	$\gamma_P$	$ \mathcal{F} $	$t$ (s)	$\gamma_P$	$t$ (s)
Texas	2000	376	411	838	0.98	411	659	17.73	411	1.81
Western	10024	4106	1825	2618	1.55	1825	2010	158.51	1825	2.16
Eastern	70047	30332	12895	27019	552.46	>12789	>15043	>10 h	>12890	>10 h
USA	82071	34814	15131	30357	728.62	>14124	>16391	>10 h	>15126	>10 h

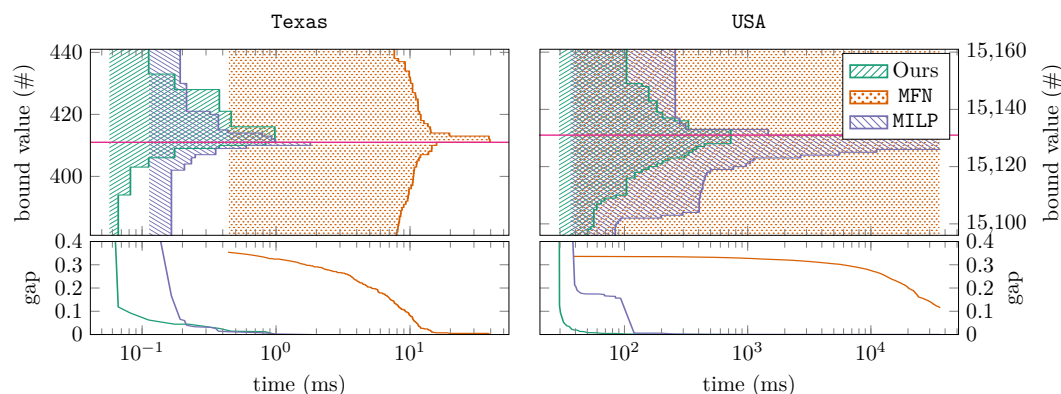
**Lower and Upper Bounds.** We note that all three approaches find lower bounds while solving the instances. In case of the implicit hitting set approach, each time we solve the current HITTING SET instance, the solution size is a lower bound for a minimum power dominating set. This yields lower bounds for our approach as well as for MFN. Moreover, Gurobi also provides lower bounds for MILP. Additionally, Gurobi provides upper bounds. To also get upper bounds for the implicit hitting set approaches, we use the following greedy heuristic. Whenever we have computed a hitting set  $H$  of the current fort neighborhoods, we greedily add vertices to  $H$ , preferably selecting vertices with many unobserved neighbors, until we have a power dominating set. Afterwards, we make sure that the resulting solution is minimal with respect to inclusion.

With this, we can observe how quickly the different algorithms converge towards the optimal solution. Figure 4 illustrates the behavior of the bounds with respect to the time for two of the four powersimdata instances. All three algorithms use our reduction rules (recall that neither MILP nor MFN were able to solve these instances without them). We clearly see that, with our approach, the gap between upper and lower bounds shrinks quickly, in particular compared to MFN. This validates our assumption that adding many – potentially larger – forts instead of a single minimum size one is highly beneficial. Recall that MFN can increase its lower bound only by at most 1 after finding a new hitting set while we can increase the lower by up to one for each undecided unhit vertex.

Interestingly, for MILP+R the gap between upper and lower bound closes much quicker than for MFN+R. In particular, for the largest USA instance, there is almost no gap left after little more than 100 s. Gurobi also found an optimal solution, but failed to prove the lower bound on its size within the timeout of 10 h. Thus, in cases where a good approximation is acceptable, MILP+R is not much worse than our approach.

**Reduction Rules.** To evaluate the effect of the reduction rules on the performance of our algorithm, we let it run on the pandapower instances with different subsets of reduction rules. Recall that we have several local reduction rules as well as the two non-local rules Dom and NecN. In addition to using all or no reduction rules, we consider the following subsets. Only local rules, only non-local rules, all local rules together with Dom, and all local rules together with NecN.

Figure 5a shows the median running time for each instance in the different settings. In most instances, the reductions could decrease the running time by an order of magnitude or more. Moreover, we can see that in most cases all reduction rules are relevant, i.e., we achieve the lowest run time when using all reduction rules and applying no reduction rules is usually slower than applying any of the rules.



■ **Figure 4** Upper and lower bounds on the optimum value on the *Texas* and *USA* powersimdata instances with preprocessing by our reduction rules. We give the bounds reported by our solver and by MFN, both with added greedy upper bounds, as well as Gurobi for MILP. Lines and shaded areas each start at the time of the first respective bound. Note that the x axis uses a logarithmic scale.

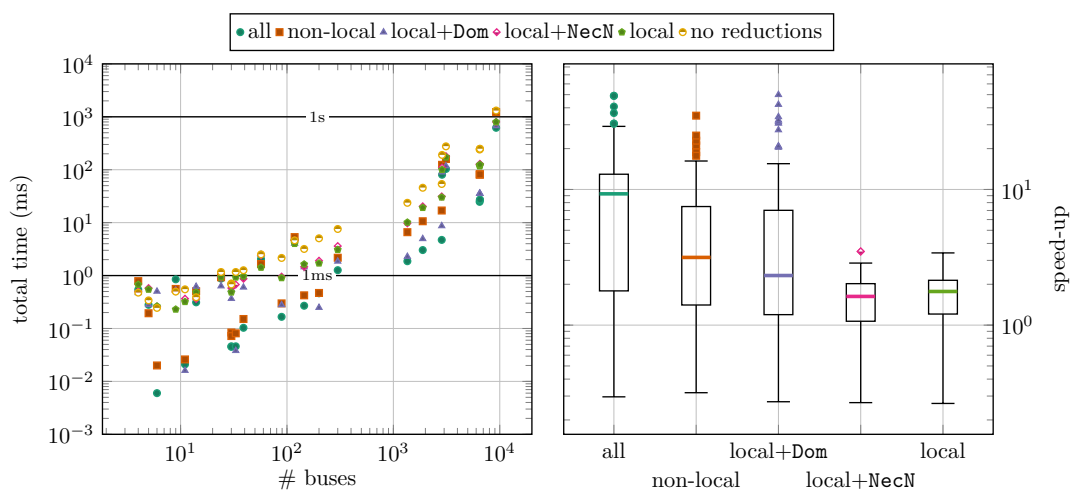
Figure 5b shows the speedup aggregated over all instances of using reduction rules compared to using no reduction rules for our solver. We can see that the median speedup is roughly one order of magnitude when applying all reduction rules. The most interesting observation here is that *local+NecN* does not give any improvement compared to just *local*. In fact, it is slightly slower. However, when combined with *Dom*, *NecN* gives a significant improvement.

## 6 Conclusion

We showed that PDS is  $W[P]$ -complete. This closes the gap in the study of its parameterized complexity. Our reduction uses an auxiliary problem, IPDS, to simulate arbitrary monotone circuits.

Our second contribution in this paper is a set of new reduction rules for PDS. The rules yield partially solved instances of PDS-EXTENSION where some vertices are pre-selected for the power dominating set while other are forbidden from being included. Each rule shrinks the instance by removing vertices or edges, or pre-selects or excludes vertices from being selected. Our reduction rules can be used as a pre-processing step to significantly enhance the performance of existing solvers. Our third and last contribution is a new algorithm for solving PDS based on the implicit hitting set approach. The core of our algorithm is a new heuristic to find missing sets for the implicit hitting set instances. We evaluate the effectiveness of our reduction rules and the performance of our algorithm in experiments on a set of practical power grid instances from the literature. For comparison, we run the same experiments with two different approaches from the literature. The comparison shows clearly that our new heuristic for finding missing fort neighborhoods outperforms the previous approach. Our algorithm outperforms the reference solvers by more than one order of magnitude. Even when combining the other approaches with our reduction rules, our algorithm beats them on most instances. Furthermore, we can solve large instances of continental scale that could not be solved before. We found that our algorithm finds lower bounds on the power dominating number more quickly than Gurobi.

A major advantage of our fort heuristic is that it translates easily to other variants of PDS, as long as it is easy to verify which vertices are observed by a partial solution. Examples of such variant are the  $k$ -POWER DOMINATING SET where propagation is possible



(a) Median running time on each instance with the different subsets of reduction rules.

(b) Aggregated speed-up of each set of reduction rules compared to our algorithm without reduction rules.

■ **Figure 5** Running times and speed-up of our algorithm with different subsets of the reduction rules on the pandapower instances.

if a vertex has less than  $k$  unobserved neighbors or  $l$ -ROUND POWER DOMINATING SET where the number of propagation steps is limited. Other variants, such as CONNECTED POWER DOMINATING SET are less straightforward. It might be interesting to see if connectivity can be efficiently enforced in the implicit hitting set model.

Even though our algorithm shows a significant improvement over the state-of-the-art, there is still some potential for further engineering. Currently, our implementation of the reduction rules is optimized for a single execution as a pre-processing step. Further optimization might make them more efficient, especially when only few vertices have changed between rule applications. This might be useful in more accurate heuristics solutions on large instances or for use in a branching algorithm. Further fast high quality heuristics can provide good upper bounds on the solution size. Such a heuristic, combined with the lower bound provided by our algorithm, might prove optimality earlier, further reducing the run time. Also, other hitting set solvers beside Gurobi exist and our algorithm might benefit from using those instead.

## References

- 1 Ashkan Aazami. Domination in graphs with bounded propagation: algorithms, formulations and hardness results. *Journal of Combinatorial Optimization*, 19:429–456, 2008. doi:10.1007/s10878-008-9176-7.
- 2 T. L. Baldwin, L. Mili, M. B. Boisen, and R. Adapa. Power system observability with minimal phasor measurement placement. *IEEE Transactions on Power Systems*, 8:707–715, 1993. doi:10.1109/59.260810.
- 3 Daniel Binkele-Raible and Henning Fernau. An exact exponential time algorithm for power dominating set. *Algorithmica*, 63(1):323–346, 2012. doi:10.1007/s00453-011-9533-2.
- 4 Thomas Bläsius and Max Göttlicher. An efficient algorithm for power dominating set, 2023. arXiv:2306.09870.



- 5 Chassidy Bozeman, Boris Brimkov, Craig Erickson, Daniela Ferrero, Mary Flagg, and Leslie Hogben. Restricted power domination and zero forcing problems. *Journal of Combinatorial Optimization*, 37:935–956, 2018. doi:10.1007/s10878-018-0330-6.
- 6 Boris Brimkov, Derek Mikesell, and Logan Smith. Connected power domination in graphs. *Journal of Combinatorial Optimization*, 38(1):292–315, 2019. doi:10.1007/s10878-019-00380-7.
- 7 Dennis J. Brueni. Minimal pmu placement for graph observability: a decomposition approach. Master’s thesis, Virginia Polytechnic Institute and State University, 1993. doi:10919/45368.
- 8 Dennis J. Brueni and Lenwood S. Heath. The pmu placement problem. *SIAM J. Discret. Math.*, 19:744–761, 2005. doi:10.1137/S0895480103432556.
- 9 Jiong Guo, Rolf Niedermeier, and Daniel Raible. Improved algorithms and complexity results for power domination in graphs. *Algorithmica*, 52(2):177–202, 2008. doi:10.1007/s00453-007-9147-x.
- 10 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL: <https://www.gurobi.com>.
- 11 Teresa W Haynes, Sandra M Hedetniemi, Stephen T Hedetniemi, and Michael A Henning. Domination in graphs applied to electric power networks. *SIAM journal on discrete mathematics*, 15(4):519–529, 2002. doi:10.1137/S0895480100375831.
- 12 Mikoláš Janota and Joao Marques-Silva. Solving qbf by clause selection. In *International Joint Conference on Artificial Intelligence*, 2015.
- 13 Raka Jovanovic and Stefan Voss. The fixed set search applied to the power dominating set problem. *Expert Systems*, 37(6):e12559, 2020. doi:10.1111/exsy.12559.
- 14 Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Parameterized power domination complexity. *Information Processing Letters*, 98(4):145–149, 2006. doi:10.1016/j.ipl.2006.01.007.
- 15 Chung-Shou Liao and Der-Tsai Lee. Power domination in circular-arc graphs. *Algorithmica*, 65(2):443–466, 2013. doi:10.1007/s00453-011-9599-x.
- 16 L. Mili, Thomas L. Baldwin, and R. Adapa. Phasor measurement placement for voltage stability analysis of power systems. *29th IEEE Conference on Decision and Control*, pages 3033–3038 vol.6, 1990. doi:10.1109/CDC.1990.203341.
- 17 Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: A SAT-IP Hybrid MaxSAT Solver. In *International Conference on Theory and Applications of Satisfiability Testing*, 2016. doi:10.1007/978-3-319-40970-2\_34.
- 18 Logan A. Smith and Illya V. Hicks. Optimal sensor placement in power grids: Power domination, set covering, and the neighborhoods of zero forcing forts, 2020. arXiv:2006.03460.
- 19 L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower – An open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, November 2018. doi:10.1109/TPWRS.2018.2829021.
- 20 Guangjun Xu, Liying Kang, Erfang Shan, and Min Zhao. Power domination in block graphs. *Theoretical computer science*, 359(1-3):299–305, 2006. doi:10.1016/j.tcs.2006.04.011.
- 21 Yixing Xu, Nathan P Myhrvold, Dhileep Sivam, Kaspar Mueller, Daniel Julius Olsen, Bainan Xia, Daniel Livengood, Victoria Hunt, Benjamin Rouill’e d’Orfeuill, Daniel B. C. Muldrew, Merrielle Ondreicka, and Megan Bettilyon. U.s. test system with high spatial and temporal resolution for renewable integration studies. *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, 2020. doi:10.1109/PESGM41954.2020.9281850.