Algorithmica
https://doi.org/10.1007/s00453-024-01283-8

An Efficient Algorithm for Power Dominating Set

Thomas Bläsius[1] · **Max Göttlicher**[1]

Received: 6 November 2023 / Accepted: 18 November 2024

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.

Keywords Power dominating set · Implicit hitting set · Parameterized complexity · Reduction rules

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 proposed by Mili et al. [19] 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

✉ Max Göttlicher
max.goettlicher@kit.edu

Thomas Bläsius
thomas.blaesius@kit.edu

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Published online: 23 December 2024

[8]: 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 [8, 14, 17], 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 [8, 9, 12, 14, 17, 18, 24]. In terms of parameterized complexity, PDS is known to be $W[2]$-hard when parameterized by solution size [12].

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 [12, 17]. 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 [3, 8] 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 [7] and most recently Jovanovic and Voss [16]. A different approach is to reduce PDS to the hitting set problem [5, 22]. 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. The implicit hitting set approach has been used for other problems, e.g., for MaxSAT [21] and TQBF [15]. For PDS, it has been introduced by Bozeman et al. [5]. The strategy of finding forts has been later improved by Smith and Hicks [22], 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 [12], 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 running time of 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 Sect. 3, we show that PDS is $W[P]$-complete. Our reduction rules and the heuristic for extending the hitting set instance are presented in Sect. 4. Section 5 contains our experimental evaluation of the new method and a comparison with the previous methods using a set of benchmark instances.

2 Preliminaries

2.1 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.

2.2 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[1]. 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, we refer to the special case where all vertices are propagating as 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 is a solution $S \subset V$ such that S includes all selected and excludes all excluded vertices, i.e., $X \subseteq S$ and $Y \cap S = \emptyset$. The problem PDS-Extension is defined analogously.

[1] The propagation rule is motivated by Kirchoff's law and Ohm's law in electric transmission networks. Propagating vertices are also called *zero-injection vertices*. In electric networks, they refer to buses in substations without power injection, i.e. without attached loads or generators.

2.3 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.

2.4 Parameterized Complexity

We only give a very brief introduction; for more details, see one of the text books on parameterized complexity [11]. For a parameterized problem, we are given a parameter k in addition to the instance. The running time is then not only analyzed in terms of the input size but also in terms of k. We consider all problem variants introduced above with their canonical parameterization, i.e., parameterized by their solution size. A parameterized problem is *fixed parameter tractable (FPT)* if it can be solved in $f(k) \cdot n^{O(1)}$ where f is a computable function.

The W-Hierarchy To show that a problem is probably not in FPT, one can show hardness in terms of the W-hierarchy. The *W-hierarchy* consists of complexity classes FPT $\subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$, where each inclusion is assumed to be strict. Many graph problems are known to be complete for $W[1]$ and $W[2]$, e.g., INDEPENDENT SET and the above mentioned DOMINATING SET are $W[1]$- and $W[2]$-complete, respectively. We will see that the other variants of PDS defined above are $W[P]$-complete.

Proving $W[P]$-Hardness To show that a problem is $W[P]$-hard, we need to reduce to it from another $W[P]$-hard problem using a *parameterized reduction*, i.e., a reduction that runs in FPT-time such that the change in the parameter is independent of the input size. The $W[P]$-hard problem we reduce from is WEIGHTED MONOTONE CIRCUIT SATISFIABILITY (WMCS), which is defined as follows. A *Boolean circuit* is a directed acyclic graph with a unique sink (the output node), where the sources are inputs and the inner nodes are logic gates (`and`, `or`, `not`). This defines a boolean function mapping the values on the input nodes to a Boolean output in the canonical way. The problem WEIGHTED CIRCUIT SATISFIABILITY (WCS) with parameter k asks whether there is a satisfying assignment that sets k inputs to TRUE and all other to FALSE. WEIGHTED MONOTONE CIRCUIT SATISFIABILITY (WMCS) is the same problem with restriction that there are no `not`-gates. It is known that WMCS is $W[P]$-complete [11].

Inclusion in $W[P]$ Conversely, to show that a problem is in $W[P]$, we use the following theorem.

Theorem 1 ([10, Thm. 3.7], [17]) *Let L be a parameterized problem in NP. Then $L \in W[P]$ if and only if there is a non-deterministic Turing machine deciding L in at most $f(k)\log|x|$ non-deterministic and $(|x|+k)^{O(1)}$ deterministic steps, where x is the input, k the parameter, and f a computable function.*

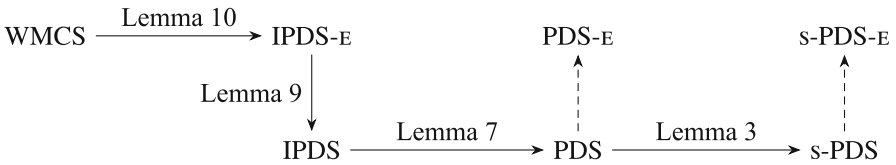


Fig. 1 Reduction steps to show that PDS and its variants are $W[P]$ hard. The solid arrows indicate our parameterized reductions described in this section. The hardness of the extension problems follows from the hardness of their basic problems, indicated by the dashed arrows

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; for a brief introduction, see Sect. 2.4. We start by introducing a variant of the PDS problem that we use as an intermediate problem in our chain of reductions.

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 Sect. 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.

The extension variant IPDS-EXTENSION is defined analogously to PDS-EXTENSION. We first note that proving containment of IPDS-EXTENSION in $W[P]$ is straight forward by giving an appropriate non-deterministic Turing machine. The analogous statement for PDS has been observed before by Kneis et al. [17].

Lemma 2 IMPLICATING POWER DOMINATING SET EXTENSION *is in* $W[P]$.

Proof A nondeterministic Turing machine can guess a solution S of size k in $k \log|V|$ nondeterministic steps. It can then check whether all vertices in G are observed by S in polynomial time. Thus, by Theorem 1, IPDS-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.

3.1 Power Dominating Set to Simple Power Dominating Set

Our chain of reductions to prove $W[P]$-hardness is illustrated in Fig. 1. We start with the reduction from PDS to SIMPLE PDS, which is similar to the proof of $W[2]$-hardness by PDS [12, 17]. The core idea is to simulate a non-propagating vertex with a propagating vertex with an additional leaf attached.

Lemma 3 *There is a parameterized reduction from* POWER DOMINATING SET *to* SIMPLE POWER DOMINATING SET.

Proof Our proof is similar to the proof of W[2]-hardness of POWER DOMINATING SET [12, 17]. Let $G = (V, E)$ be a PDS-instance with non-propagating vertices $Z \subseteq V$ (i.e., $V \setminus Z$ are propagating). We build a SIMPLE- PDS-instance $G' = (V \cup V', E \cup E')$ (i.e., $V \cup V'$ are all propagating) from $G = (V, E)$ by attaching a new leaf to each non-propagating vertex $v \in Z$.

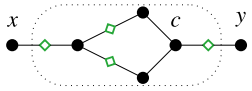
On an intuitive level, attaching a leaf to a vertex has two effects. First, it is never optimal to choose a leaf to be part of the solution. Second, a vertex with an attached leaf can never propagate to any vertex except the leaf. Thus, attaching a leaf to every non-propagating vertex has the desired effect. Note that this is a parameterized reduction as the parameter is not changed. To make this argument more formal, we show that G has a solution of size k if and only if G' has a solution of size k.

Let S be a power dominating set of G, i.e., applying the domination rule and the propagation rule (restricted to propagating vertices) exhaustively observes every vertex in V. Now interpret S as a solution for G'. Note that applying the domination rule has the same effect as before. Additionally, as the application of the propagation rule to G was restricted to propagating vertices, whose neighborhood did not change in G', it can be applied in the same way to G. Thus, all vertices in V will be observed in G', too. Applying additional propagation rules also observes the vertices in V'.

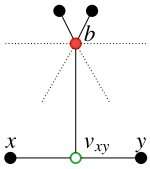
Now let S be a power dominating set of G'. Note that it is never optimal to choose one of the new degree-1 vertices as part of the solution, i.e., we can assume without loss of generality that $S \cap V' = \emptyset$. As S is a power dominating set of G', applying the domination rule and the propagation rule exhaustively observes all vertices. Interpreting S as a solution for G, we can apply the domination rule as for G'. Additionally the propagation rule can be applied in the same way for propagating vertices, i.e., vertices not in Z. If, for G', the propagation rule is applied to a vertex in $u \in Z$, then the newly observed vertex must be the leaf attached to u in the construction of G'. Thus, all applications of the propagation rule in G' also happen in G except for those observing vertices in V'. Hence, S is also a solution in G with non-observed vertices Z. □

3.2 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 shown in Fig. 2a. Using another gadget (shown in Fig. 2b), we eliminate booster edges in a similar way, yielding the reduction.



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



(b) Gadget for a edge between x and y. The leaves enforce the selection of b (red). Thus v_{xy} is observed (green) by b and can in turn observe either x or y by the propagation rule.

Fig. 2 Gadgets for implication arcs and booster edges (Color figure online)

Lemma 4 *Every instance of* Implicating Power Dominating Set *can be reduced to an equivalent instance with no implication arcs without changing the parameter.*

Proof Let G be an instance of IPDS with implication arcs A and let $a = (x, y) \in A$ be an implication arc. We show that replacing a with the gadget I_a in Fig. 2a yields an equivalent instance with one less implication arc. Applying this replacement to all implication arcs yields the claim.

To explain this on an intuitive level, consider the implication gadget I_a and first assume that x is observed. It is not hard to verify that, by applying the booster and the propagation rules, all vertices in I_a and y are observed. Conversely, if only y is observed, then c cannot propagate due to its two unobserved neighbors. Thus, I_a mimics the behavior of an implication arc.

To formally prove the claim, let G' be the instance obtained by replacing one implication arc a with the implication gadget I_a. We show G has a solution of size k if and only if G' has a solution of size k. For the first direction, let S be a solution of G. Consider a corresponding sequence $r_1, \ldots, r_\ell$ of observation rules. Let V_i be the set of observed vertices after the application of the first i rules $r_1, \ldots, r_i$. Observe that the vertices in S are only observed after the domination rule is applied and thus we define $V_0 = \emptyset$. Now interpret S as a candidate solution for G'. Based on $r_1, \ldots, r_\ell$, we give a sequence of observation rules on G' that fully observes the graph. Our argument is of inductive nature, i.e., for fixed i, we assume that we have a rule sequence for G' that observes at least the vertices in V_{i-1} and we show that it can be extended to observe the vertices in V_i using a set of rules based on r_i. In the following we discriminate between the possible types of r_i. If r_i is the domination or the booster rule, it can be applied as in G. If r_i is the propagation rule and the propagating vertex is not x or y, it can be applied as in G. Otherwise, note that the only new neighbors to x and y are connected via booster edges. Thus, if x or y are observed, we can apply the booster rule to observe their new neighbors. Afterwards, the propagation rule can be applied as usual, as the relevant vertex has again only one unobserved neighbor. If r_i is the implication rule applied to an implication arc other than $a = (x, y)$, we can apply it as before. Otherwise, the above observation shows that all vertices in the implication gadget I_a and y will be observed, which observes at least all vertices in V_i. It thus follows that after step ℓ all vertices in $V_\ell = V$ will be observed in G'. Additionally the vertices inside the gadget I_a can also be observed by applying observation rules knowing that $x \in V_\ell$ is observed.

Conversely, assume that S is a solution for G'. Without loss of generality, we assume that S does not contain any vertices from I_a as we can otherwise choose x instead and apply booster and propagation rules to observe the whole gadget. Additionally let $r_1, \ldots, r_\ell$ be a sequence of rule applications that observe the whole graph G' and let V_i again be the set of vertices observed after applying $r_1, \ldots, r_i$ where $V_0 = \emptyset$. Now consider the lowest index ι such that $x \in V_\iota$. Then without loss of generality, assume that the rules following r_ι are booster and propagation rules that observe the whole gadget I_a and y before any other rules are applied. Now consider S to be a solution candidate in G. We can apply the sequence of rules $r_1, \ldots, r_\ell$ as in G' except for the rules observing I_a following r_ι. However, this subsequence of rules can be replaced with a single application of the implication rule on (x, y), which concludes the proof. □

The gadget for simulating booster edges, shown in Fig. 2b, requires adding a globally unique selected vertex b to which all 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 two leaves.

Lemma 5 *If an* IPDS*-instance contains a vertex b with two or more leaves that do not have any booster edges or implication arcs, there is a minimum power dominating set containing b.*

Proof We show that at least one of b or its leaves, u and v must be selected. First, assume that none of these vertices is selected. If b is not propagating, there is no observation rule by which u and v can be observed. Otherwise the only candidate rule is the propagation rule which in turn cannot be applied to b before at least one of the leaves is observed. Hence, every power dominating set must contain at least one of b, u or v.

In this case we can always select b and observe the leaves with the domination rule. □

Lemma 6 *Every* IPDS*-instance with booster edges can be reduced to an equivalent instance without booster edges.*

Proof Let $G = (V, E)$ be an IPDS-instance with booster edges and let $xy \in E$ be a booster edge. Our construction requires a known selected vertex. To ensure such a vertex exists, we insert a new vertex b and attach two leaves to b. By Lemma 5 we can assume without loss of generality that b is selected.

We show that subdividing xy by adding a third vertex v_{xy} between x and y and connecting v_{xy} to b yields an equivalent instance G' with one fewer booster edge. Replacing every booster edge in this way yields an instance without booster edges. This gadget construction is also shown in Fig. 2b. Note that we only need to insert b once when replacing the booster edges iteratively.

Intuitively, the gadget introduces a vertex v_{xy} between x and y that is always observed. Then, if either x or y become observed at some point, v_{xy} has only one unobserved neighbor left which becomes observed by applying the propagation rule.

To formally prove the claim, we show that G has a power dominating set of size k if and only if G' has a power dominating set of size $k + 1$. For the first direction let S be

a minimum power dominating set of G and let $r_1, \ldots, r_\ell$ be a sequence of observation rules the application of which observes all vertices. We show that $S \cup \{b\}$ is a power dominating set of G'. Based on $r_1, \ldots, r_\ell$ we give a sequence of rule applications that observes all vertices in G'.

We use an inductive argument, i.e. for a given fixed i we assume that we have a sequence of observation rules $r'_1, \ldots, r'_j$ for G' that observes at least all vertices observed by $r_1, \ldots, r_{i-1}$ in G. Based on r_i, we extend that sequence with observation rules $r'_{j+1}, \ldots, r'_{j'}$ to observe all missing vertices that become observed by applying r_i.

The first rule r'_1 we apply in G' is the domination rule on b.

In each step we handle rules that are affected by the introduction of the gadget.

The booster gadget does not affect the implication rule which can thus be applied as before. The other rules require adaptation if they are applied to x or y.

If r_i if the domination rule applied to x, we use the domination rule in G' and add an application of the propagation rule on v to ensure y becomes observed. The same applies to the domination rule on y. All other edges remain the same in the construction and thus the domination rule can be applied as before.

If r_i is the propagation rule and x is the propagating vertex, propagating to y, we could instead use the booster rule, so we proceed like described there. The same applies if y is the propagating vertex, observing x. All other application of the propagation rule can be applied unchanged in G' due to our induction hypothesis.

If r_i is booster rule applied to xy we replace it by an application of the propagation rule applied to v_{xy}. The inserted vertex v_{xy} is observed in the application of the domination rule in r'_1. Applying the booster rule requires an observed endpoint, let this be x. By assumption, x is also observed in G' where v has thus at most one unobserved neighbor, y. The propagation rule can thus be applied in G' and y becomes observed. Applications of the booster rule to other edges are applied in G' in the same way as in G.

The argument for the other direction follows the same structure with the roles of G and G' swapped. Let S be a minimum power dominating set of G' and let $r'_1, \ldots, r'_\ell$ be a corresponding sequence of observation rules observing all vertices. Given $r'_1, \ldots, r'_\ell$ we construct a sequence of rule applications that observes all vertices in G.

We ignore all rules applied to b and its attached leaves. With the exception of the propagation rule, all observation rules can be applied in G in the same way as in G'. If r'_i is the propagation rule we distinguish three cases by the propagating vertex.

1. The propagating vertex is v_{xy}. We know that x or y is observed; without loss of generality assume this is x. We apply the booster rule on xy and y becomes observed in G.
2. The propagating vertex is x. We first apply the booster rule on xy to make sure y is observed in G. Now x has the same unobserved neighbor in G and G' and thus the rule can be applied like before. The same applies if the propagating vertex is y.
3. Otherwise, the propagation rule can be applied as before.

□

Lemma 7 *There is a parameterized reduction from* IMPLICATING POWER DOMINATING SET *to* POWER DOMINATING SET.

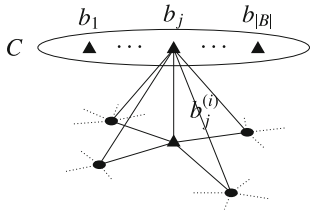


Fig. 3 The vertices b_j in the top-level clique C connect to the neighborhood of their corresponding vertices $b^{(i)}{}_j$ in each copy $G^{(i)}$ of G

Proof An IPDS-instance G may have implication arcs and booster edges, which are not allowed in PDS instances. As shown in Lemmas 4 and 6, we can construct an equivalent instance instance G' which does not contain implication arcs or booster edges. Lacking those special edges, G' is also a PDS instance. Thus, this construction is a parameterized reduction from IPDS to PDS, increasing the parameter, i.e. the solution size, by one in the process. □

3.3 Extension to Non-Extension (For IPDS)

A reduction from IPDS-Extension to IPDS requires a mechanism to enforce the selection of certain vertices and to make sure other vertices are not selected in a minimum power dominating set. We already saw a way to ensure that a vertex gets selected in Lemma 5. The core difficulty thus comes from enforcing the excluded vertices to not be selected. The basic idea how we achieve this is by using many copies of the graph and an additional clique of non-propagating vertices. The vertices in the clique represent the vertices that are allowed to be selected and provide the only connection between the copies. With this construction, selecting vertices outside the clique is never optimal.

Lemma 8 *Every instance of* IPDS-Extension *can be reduced to an equivalent instance without excluded vertices without changing the parameter.*

Proof Let $G = (V, E)$ be an IPDS-Extension-instance and let Y be a set of vertices excluded from a solution to G. We construct an IPDS-Extension instance G' where no vertices are explicitly excluded.

Let $B = \{b_1, \ldots, b_{|B|}\} = V \setminus Y$ be the set of vertices allowed in a solution. We initialize G' with $|V| + 1$ copies of G which we denote by $G^{(i)}$ and a clique C consisting of the vertices in B. Each copy $G^{(i)}$ consists of a copy of the vertices $V^{(i)} = \{v^{(i)} \mid v \in V\}$. These copied vertices are connected by edges if their original counterparts are connected, i.e. $E^{(i)} = \{v^{(i)}w^{(i)} | vw \in E\}$. Each vertex b_j in the clique has edges to all vertices in the neighborhood of its counterparts in the copies, i.e. there is an edge from b_j to every vertex in $N[x^{(i)}]$ in every copy i. Each vertex $v^{(i)}$ in the copy $G^{(i)}$ is propagating if and only if v is propagating in G. All vertices in C are non-propagating. See Fig. 3 for an example.

This construction ensures that vertices selected in one $G^{(i)}$ never observe vertices in other copies or cause them to become observed by the propagation rule. If a minimum power dominating set contains a vertex in one $G^{(i)}$, it must therefore also contain

vertices from each other $G^{(i')}$. Hence, a minimum power dominating set of size less than or equal to $|V|$ cannot contain any vertices outside C.

It remains to show that G has an implicating power dominating set of size k if and only if G' has a power dominating set of size k. Let $S \subseteq V$ be a minimum power dominating set of G and let $r_1, \ldots, r_\ell$ be a sequence of observation rules that observes all vertices in G. It is easy to verify that S is also a power dominating set of G' by modifying the sequence r_j. If r_j is an application of the domination rule, we apply it to the same vertex in G' where it is part of the clique. By definition, S contains only vertices in B. Otherwise, r_i is some other observation rule and we apply it in every copy $G^{(i)}$ in the same way as in G. The first application of a domination rule observes all vertices in C. Thus, after the transformed sequence of observation rules is applied, all vertices in G' are observed.

Now let S be a minimum power dominating set of G'. It follows from the previous direction that, if a minimum power dominating set of G' has more than $|V|$ vertices, G does not have a power dominating set. We show by contradiction that S can only contain vertices in the top-level clique C. Assume $S \setminus B \neq \emptyset$. Because $|S| \leq |V(G)|$, there is at least one $G^{(\hat{i})}$ without any vertices in S. By out assumption that S is a power dominating set, all vertices in $G^{(\hat{i})}$ are observed by S. All vertices in C are non-propagating and have no booster edges or implication arcs. This means that $S \cap B$ is a power dominating set of the induced sub-graph in G' by B and $V(G^{(\hat{i})})$. But then, $S \cap B$ is a power dominating set for G' as all $G^{(i)}$ are identical. This contradicts our assumption that S is a minimum power dominating set and hence no minimum power dominating set can have vertices not in B. □

Lemma 9 *There is a parameterized reduction from* Implicating Power Dominating Set Extension *to* Implicating Power Dominating Set.

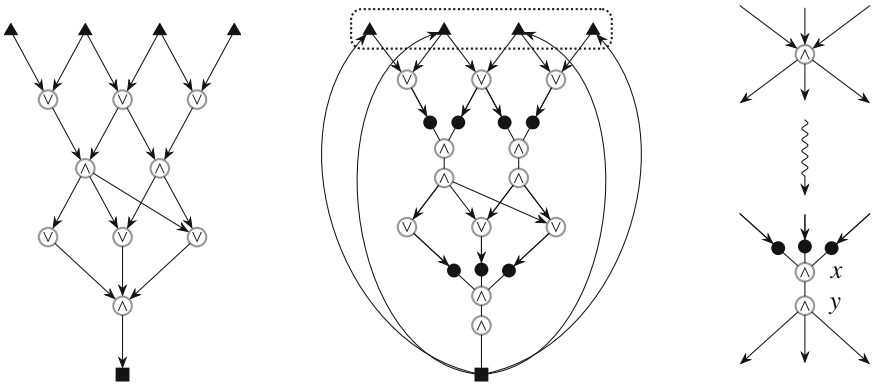
Proof IPDS-Extension differs from IPDS by requiring the solution to contain certain vertices while excluding others. We can use Lemma 5 to force the inclusion of vertices in the solution by adding two leaves. Lemma 8 provides a method to enforce the exclusion of vertices. Together the yield a parameterized reduction from IPDS-Extension to IPDS. □

3.4 WMCS to IPDS-Extension

So far we only considered different extensions of the Power Dominating Set problem. We will now see that those extensions allow a straightforward reduction from Weighted Monotone Circuit Satisfiability, a $W[P]$ complete problem. The core idea is to replace the arcs in the directed acyclic graph describing the circuit with implication arcs and to model `and`-gates using a gadget as show in Fig. 4c. This last step in our reduction chain finally yields the $W[P]$-hardness of PDS.

Lemma 10 *There is a parameterized reduction from* Weighted Monotone Circuit Satisfiability *to* Implicating Power Dominating Set Extension.

Proof Let $C = (V, E)$ be a monotone circuit interpreted as acyclic graph with input nodes $V_{\texttt{in}}$, `and`-gates $V_{\texttt{and}}$, `or`-gates $V_{\texttt{or}}$ and the output node `out`. Figure 4a show



(a) Input monotone circuit consisting of `and`-gates and `or`-gates

(b) Corresponding IPDS instance. Only the triangular vertices in the top clique are selectable.

(c) `and`-gadget

Fig. 4 Example of a transformation from a circuit to an IPDS instance. We transform the circuit in **a** to the semi-directed graph in **b** by replacing the `and`-gates with the gadget in **c** and inserting implication edges from the output to each input

an example of such a circuit. Its corresponding IPDS-EXTENSION-instance is depicted in Fig. 4b. To construct such an instance from C we proceed as follows. We interpret all directed edges E in C as implication arcs A and add further implication arcs from the output node `out` to every input. Next 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; see Fig. 4c for an example. For every incoming edge of v from a vertex u, we place a fresh vertex x_u and add an edge $x_u x$ and an implication arc (u, x_u). We refer to x as the *input* of the gate and to y as the *output* of the gate. We call the fresh nodes x_u *proxy inputs*. They are marked in gray in Fig. 4c. The other gates and vertices are their own input and output. We exclude all vertices except the inputs from a solution to the IPDS-EXTENSION instance.

The basic idea of the construction is that we interpret true values in the circuit as a node being observed in G. We can then simulate `and`-gates by exploiting the propagation rule and `or`-gates using the implication rule. Whenever a single input node of an `or`-gate becomes observed, the implication rule ensures that first the gate and then all its children become observed, too. Each proxy input propagates its observation to the gate input x, which in turn can only propagate its observation to the gate output y when all inputs are observed.

It remains to show that C has a satisfying assignment with at most k true variable if and only if G has a power dominating set of size k. □

▷ Claim (A satisfying assignment of C implies a power dominating set in G) Let $\mathcal{X}$ be a satisfying assignment with true variables $v_1, \ldots, v_k$ and let $x_1, \ldots, x_\ell$ be a topological ordering of the nodes and gates with TRUE output. We note that the sub-graph induced in C by $\{x_1, \ldots, x_\ell\}$ must be connected and contains a path from the input nodes to the output. A candidate solution for the IPDS-EXTENSION-instance G is $S = \{x_1, \ldots, x_\ell\}$. We use a two step argument two show that S does

indeed observe all vertices in G. First, we show inductively, that if the vertices in S are observed, the output node becomes observed. When the output node is observed, all inputs become observed by the implication arcs from the output node. The second step is to show that this suffices to observe all remaining vertices.

To show that S does observe the output node, we use an inductive proof. It is based on the assumption that for any fixed i, we can find a sequence of observation rules that observes at least the vertices in G corresponding to the TRUE gates $x_1, \ldots, x_{i-1}$. To extend the sequence to x_i, we differentiate by the type of x_i:

1. If x_i is an input, it must be in $\mathcal{X}$ and thus is selected and thus observed by the domination rule.
2. If x_i is an `or`-gate, there is some TRUE gate x_j with $j < i$. By our induction hypothesis, we know that the corresponding vertex in G is observed and we can use the implication rule on the implication arc to x_i.
3. If x_i is an `and`-gate, all gates with edges into x_j are TRUE and their corresponding vertices are thus observed. We first apply the implication rule on the implication arcs to the proxy inputs, then use the propagation rule to observe the gate input. The gate input thus has only one unobserved neighbor left, the gate output, which becomes observed by the propagation rule.
4. If, finally, x_i is the output node, we apply the implication rule on its parent node, observing x_i.

After the output node is observed, we can use the implication rule on the implication arcs to the inputs, observing all of them. Note that in a monotone circuit with all inputs set to TRUE, all gates and the output are TRUE, too. Then, by the same inductive argument as above, all vertices in G become observed.

▷ Claim (A power dominating set of G implies a satisfying assignment of C) Let S be a minimum power dominating set of G. By construction, all vertices in S are input nodes and thus we obtain a candidate solution $\mathcal{X}$ where all inputs in S are set to TRUE and all others to FALSE. It remains to show that $\mathcal{X}$ is indeed a solution of C. Let $r_1, \ldots, r_{\hat{\ell}}$ be a sequence of observation rules for S that observes all vertices in G and let V_i be the set of vertices observed after the application of rule i. We only consider the subsequence $r_1, \ldots, r_\ell$ where $r_{\ell+1}$ is the first application of the implication rule on an implication arc from `out` to an input vertex.

We claim that every gate in C with an observed output node in V_ℓ outputs TRUE when evaluating the circuit. Assume the claim is false and let $\hat{\iota} \leq \ell$ be the smallest index such that $V_{\hat{\iota}}$ contains a gate output vertex v whose gate x_v in C outputs FALSE. We consider the different gate types of x_v and show that this leads to a contradiction.

1. If x_v is an input, we know that $x_v \in \mathcal{X}$. Otherwise, the only way to observe x_v is by an implication arc from `out`, which we explicitly excluded from $r_1, \ldots, r_\ell$.
2. If x_v is an `or`-gate or `out`, v has at least one observed predecessor and became observed by the implication rule. Because $\hat{\iota}$ is minimal, x_v also has a TRUE input and therefore outputs TRUE.
3. If x_v is an `and`-gate, the gate input and the proxy inputs must all be observed. Each proxy input v_u has an incoming implication arc from the output vertex u of

some other logic gate. By definition of V_i, we know that u is observed and thus x_u outputs TRUE. So all inputs of x_v are true and it outputs TRUE.

The output `out` is thus TRUE, concluding the proof.

This proof concludes the chain of reductions as presented in Fig. 1. We saw that we can reduce from the $W[P]$-complete WEIGHTED MONOTONE CIRCUIT SATISFIABILITY to IMPLICATING POWER DOMINATING SET EXTENSION in Lemma 10. IPDS-EXTENSION, in turn, can be represented in terms of IMPLICATING POWER DOMINATING SET, as seen in Lemma 9. Lemma 7 showed that booster edges and implication arcs can be replaced by gadgets with only normal vertices. Propagating vertices can also be eliminated as we showed in Lemma 3. All these transformations are parametric reductions and thus SIMPLE- PDS is at least as hard as WMCS, i.e. both are $W[P]$-hard.

Corollary 11 POWER DOMINATING SET *is W[P] complete.*

Proof By Theorem 1, IPDS-EXTENSION is in $W[P]$. Being a special case of IPDS-EXTENSION, PDS it is also contained in $W[P]$. The $W[P]$-hardness follows from the reduction chain in Lemmas 10, 9, 7, 3. □

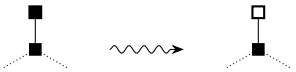
4 Solving POWER DOMINATING SET

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 Sect. 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 Sect. 4.2 we split the instance into several components that can be solved independently. Finally, each of these subinstances is solved exactly using the implicit hitting set approach [5] with our new strategy for finding new sets that need to be hit; see Sect. 4.3. Our experiments show that this strategy yields significantly better results than the previous one; see Sect. 5.2.

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 Sect. 5, we also use an MILP for this step. This MILP formulation is based on the formulation for PDS by Jovanovic and Voss [16] and we discuss our adjustments in Appendix A.1. Moreover, instead of solving the subinstances optimally, one can instead use a heuristic solver. In the latter case, 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.

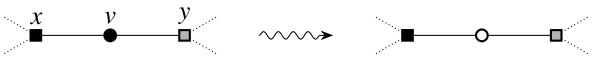
4.1 Reduction Rules

Many of our reduction rules are local in the sense that they transform one substructure into a different substructure. Of those local reduction rules, most are illustrated in Fig. 5.

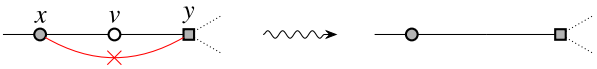


(a) `Deg1a`: If an undecided leaf is attached to a non-excluded vertex. The reduction rule excludes the leaf.

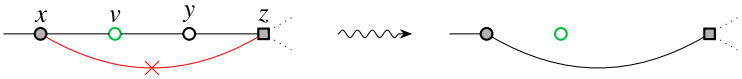
(b) `Deg1b`: Two cases of excluded leaves attached to a vertex. On the left, the parent vertex is propagating and becomes non-propagating. On the right, the parent vertex is non-propagating and becomes pre-selected. The attached leaf is removed in both cases.



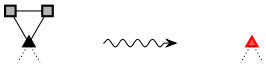
(c) `Deg2a`: A vertex v of degree two can safely be excluded if it has an undecided neighbor.



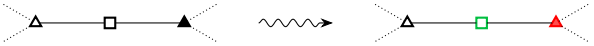
(d) `Deg2b`: The neighbors x and v of degree two can be merged if x is not adjacent to v's other neighbor y.



(e) `Deg2c`: If v is observed and has two non-adjacent unobserved neighbors of degree two, x and y, with y being excluded, then we remove y and the edge xv and connect x to the other neighbor z of y.



(f) `Tri`: We pre-select the undecided and remove the two vertices of degree two in the triangle.



(g) `OnlyN`: If an excluded vertex is surrounded by non-propagating vertices, only one of which is not excluded, we pre-select the non-excluded vertex.

Fig. 5 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 vertices ▲ are pre-selected. Vertices filled gray ◘ may be undecided, excluded or pre-selected. Green vertices □ are observed but not pre-selected. The non-existence of an edge is indicated in red ⊣⊢ (Color figure online)

Our first set of reduction rules stems from the observation that leaves, i.e. vertices of degree one, are usually not part of an optimum solution and can thus safely be excluded. To keep the rules safe, we only delete excluded leaves and only exclude leaves that can be observed from some other vertex. See Fig. 5a, b for a visual example.

Reduction `Deg1a`. Let v be an undecided leaf attached to a non-excluded vertex w. Then mark v as excluded.

Reduction `Deg1b`. Let v be an excluded leaf attached to w. If w is propagating, delete v and set w to non-propagating. If w is not excluded and not propagating, delete v and select w, i.e., set $X \leftarrow X \cup \{w\}$.

A case very similar to `Deg1b` is illustrated in Fig. 5f. Two adjacent vertices x and y of degree two share an undecided neighbor z Similar to Lemma 5, we can always select z.

Reduction `Tri`. Let x and y be two adjacent vertices of degree two with a common neighbor z. If z is undecided, select z and delete x and y.

The possibility of applying the propagation rule leads to a number of further reduction rules related two vertices of degree two, illustrated in Fig. 5c–e. Note, for example, that vertices of degree two with an undecided neighbor never need to be selected. One can always select the neighbor instead.

Reduction `Deg2a`. Let $v \in Z \setminus (X \cup Y)$ be an undecided propagating vertex with $d(v) = 2$ and let x and y be its neighbors. If x and y are not adjacent and if at least one of x and y is not excluded, exclude v.

If two adjacent vertices have degree two, the propagation rule ensures that if one becomes observed, the other becomes observed, too. The two vertices can thus be merged.

Reduction `Deg2b`. Let $v \in Y \cap Z$ be an excluded propagating vertex with $d(v) = 2$ and let x and y be its neighbors. If x and y are not adjacent and if at least one of x and y is propagating and has degree 2, delete v and add an edge between x and y instead.

Something similar happens when two unobserved vertices of degree two share an observed neighbor with no other unobserved neighbors. Recall the gadget used in the removal of the booster edges in Lemma 6, which works in the same way. If one of the two unobserved vertices becomes observed, the other becomes observed by the propagation rule. If both vertices additionally have degree two, they thus behave like a single vertex to their other two neighbors.

Reduction `Deg2c`. Let v be an observed excluded propagating vertex with precisely two unobserved neighbors, both of degree two, x and y. If x and y are not connected, y must have another neighbor z. In this case remove y and the edge xv and add a new edge xz.

There are several trivial cases where a vertex must be selected to form a feasible solution. If an undecided unobserved vertex has no neighbors, clearly that vertex must be selected. See Fig. 5g for an example. Similarly, if an unobserved excluded vertex has only non-propagating only one of which is not excluded, that neighbor must be selected.

Reduction `OnlyN`. Let v be an excluded propagating vertex of degree two with two non-propagating neighbors x and y. If x is excluded and y is undecided, select y.

Reduction `Isol`. (Isolated) Let v be an undecided isolated vertex. Then pre-select v.

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

We do not need the propagation rule to observe the vertices observed by the currently selected vertices. Instead we can connect all those vertices directly to selected vertices and observe them by the domination rule instead and remove the now redundant edges. This rule is most useful when combined with the other rules. In particular, all vertices with only observed neighbors become leaves and are then removed by reductions `Deg1a` and `Deg1b`.

Reduction ObsE. (Observed Edge) Let $vw \in E$ be an edge with two observed but not pre-selected endpoints and let $x \in X$. Then delete vw and instead insert edges vx and wx.

The previous rules identified vertices to be forbidden from a solution by looking at small localized structures. We can identify those vertices and many more by looking at the vertices that would become observed by selecting a vertex. 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 $\{v\}$. We use the observation neighborhood to formulate the following reduction rule.

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.

Reductions Deg1b, Tri and OnlyN can be seen as special cases of a more general rule. All three identify vertices that must be active in a minimum solution. There are, however some vertices that must be active in every solution. We can identify all those vertices by looking at the observation neighborhood of all other undecided vertices. This tells us if there is a solution in which the vertex is not selected.

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 Binkele-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"] that is applicable is superseded by a combination of 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, 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

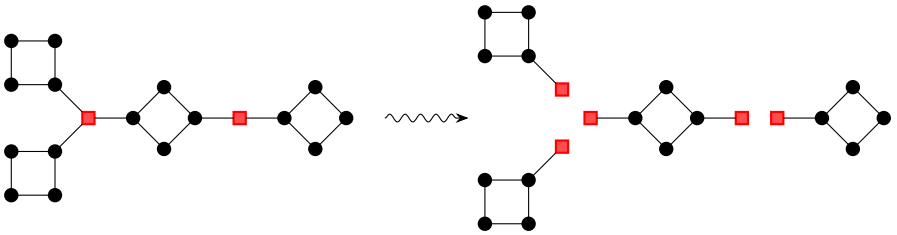


Fig. 6 Example of independent sub-problems arising from the kernel on the left. Each connected component on the right can be solved independent from the others (Color figure online)

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 Sub-instances

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

Proof Let G_i be the induced sub-graph on $N[S_i]$ and let S_i be a minimum solution of G_i. Let $S = S_1 \cup \cdots \cup S_\ell$ be a candidate solution. We show that each S_i observes at least the vertices C_i in G. Now consider a sequence of observation rules observing all vertices in G_i. Clearly, the domination rule can be applied in G in the same way as in G_i. No unselected vertex in G_i has neighbors outside G_i, so the propagation rule can also be applied in the same way. As this applies to all G_i, all vertices in G are observed and thus S is a solution.

It remains to show that S is a minimum solution. Assume S is not a minimum power dominating set and let $S*$ be a minimum solution. Then there is a sub-graph

G_i with $|C_i \cap S_I| > |C_i \cap S*|$ and $C_i \cap S*$ must be a power dominating set of G_i. This contradicts the assumption that S_i is minimal. □

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. See Fig. 6 for an example of an instance split into subinstances. The idea of dividing the problem into smaller chunks along cut-vertices has also been applied to the related zero-forcing problem [20].

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 [5, 22] 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. For example, if a candidate solution S is not a power dominating set, some vertices remain unobserved after applying the observation rules. Those unobserved vertices always constitute a fort, albeit possibly a very large one; see Lemma 13 and [6]. Moreover, each fort can only force at most one additional vertex in the next hitting set. Thus, when adding only few forts in every step, we have to potentially solve more HITTING SET instances. Our approach addresses both points by finding multiple forts at once based on the remaining unobserved vertices. Our method of finding forts is based on the following lemma, which was shown in [6].

Lemma 13 *Let $G = (V, E)$ be a graph and let $S \subseteq V$ be a set of selected vertices. Further, let R be 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.*

The proof is straightforward, so we restate it here.

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 constradicts the exhaustive application of the propagation rule and thus the set of unobserved vertices is a fort. □

By Lemma 13, 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 $U = \{u_1, \ldots, u_\ell\}$ and define a sequence $U_0, \ldots, U_\ell \subseteq U$. 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 ℓ 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 be the different algorithms converge to the optimal solution. Thirdly, we evaluate the impact of the different reduction rules.

5.1 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 along with all data sets and evaluation scripts is available on GitLab.[2] For the comparison with the previous state-of-the-art, we use the MILP formulation approach by Jovanovic and Voss [16]. In the following, we refer to this algorithm with `MILP`. The second solver by Smith and Hicks [22] 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 [22]. 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 [13].

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 [23]. We further use the Eastern, Western, Texas and US instances from the powersimdata set[3] [25] 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.

5.2 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.

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. In fact, we found that our reduction rules were able to solve 9 out of 26 instances on their own. In those cases, no solver without reduction rules could compete. Out of the remaining instances, our our solver without reduction rules was the fastest on 3 instances while our solver with reduction rules was the fastest on all others.

For the larger powersimdata instances, neither `MILP` nor `MFN` were able to compute an optimal solution without using our reduction rules within the time limit. Thus, for these instances, we only compare our solver with `MFN+R` and `MILP+R`. Table 3 shows the results. Observe that for `Eastern`, our algorithm finished after 16 min while `MFN`

[2] https://gitlab.com/Aldorn/pds-code

[3] https://github.com/Breakthrough-Energy/PowerSimData

Table 1 Run times of different combinations of PDS solvers and reduction rules on the pandapower data set

instance case*	γ_P #	MILP[a] s	MILP+R[a] s	MFN[a] s	MFN+R[a] s	Ours s	Ours+R s	Speedup[b]
4gs	2	862µ	1.09m	1.60m	1.44m	**496µ**	518µ	1.7
5	2	1.07m	676µ	1.08m	1.34m	290µ	**285µ**	3.7
6ww[c]	1	1.07m	**6µ**	450µ	20µ	188µ	**6µ**	75.0
9	2	1.89m	900µ	3.77m	2.23m	708µ	**484µ**	3.9
11_iwamoto[c]	2	3.51m	**18µ**	3.54m	**18µ**	546µ	**18µ**	194.8
14	3	1.63m	1.28m	2.67m	2.24m	530µ	**460µ**	3.6
24_ieee_rts	6	4.27m	4.97m	5.40m	3.76m	1.24m	**726µ**	5.9
30[c]	6	2.83m	58µ	4.69m	143µ	704µ	**56µ**	50.5
ieee30[c]	6	3.41m	132µ	4.35m	**43µ**	602µ	44µ	77.5
33bw[c]	11	1.82m	**45µ**	5.75m	54µ	712µ	144µ	12.6
39[c]	9	6.16m	267µ	12.73m	**103µ**	1.76m	120µ	51.4
57	12	8.91m	9.93m	23.58m	15.34m	**1.57m**	1.79m	5.0
89pegase[c]	13	21.57m	171µ	11.90m	423µ	2.46m	**169µ**	70.4
118	29	13.72m	12.59m	64.71m	51.13m	**5.03m**	5.90m	2.3
145[c]	18	121.84m	**267µ**	28.94m	271µ	4.19m	269µ	107.6
illinois200[c]	39	20.38m	**273µ**	166.20m	278µ	25.27m	283µ	72.0
300	72	27.94m	2.80m	204.81m	4.19m	11.74m	**1.40m**	20.0
1354pegase	311	101.02m	2.50m	2.06	2.85m	40.89m	**1.95m**	51.9
1888rte	375	552.09m	6.54m	6.34	5.20m	125.95m	**3.07m**	179.6
2848rte	585	602.79m	7.41m	15.37	6.43m	154.96m	**5.04m**	119.5
2869pegase	612	1.22	220.63m	16.43	1.59	182.89m	**73.88m**	16.5
3120sp	768	1.34	273.92m	37.89	1.68	252.51m	**97.74m**	13.7
6470rte	1303	2.99	44.27m	89.97	71.98m	720.30m	**26.58m**	112.5
6495rte	1314	3.56	46.35m	87.05	92.95m	825.62m	**27.17m**	130.9
6515rte	1315	3.93	45.93m	92.09	97.57m	800.46m	**27.75m**	141.7
9241pegase	2010	5.70	1.32	217.46	13.67	820.84m	**520.57m**	11.0

The lowest running time is marked in bold

Note that γ_P 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

[a] Numbers here were obtained from our interpretation of the respective approach

[b] Speedup of Ours+R compared to the faster of MILP and MFN

[c] Solved optimally by reduction rules

did not finish after more than 6 h, with a lower bound that was still more than 100 vertices below the optimal solution. Observe that the number of fort neighborhoods $|\mathcal{F}|$ is slightly lower for MFN, which is to be expected as this 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 Run times of different combinations of PDS solvers and reduction rules on the pandapower data set with all vertices considered propagating

instance case*	γ_P #	MILP[a] s	MILP+R[a] s	MFN[a] s	MFN+R[a] s	Ours s	Ours+R s	Speedup[b]
4gs[c]	1	2.40 m	**2** µ	388 µ	3 µ	196 µ	3 µ	129.3
5[c]	1	2.99 m	**7** µ	570 µ	11 µ	241 µ	19 µ	30.0
6ww[c]	1	3.30 m	12 µ	1.53 m	12 µ	251 µ	**7** µ	218.4
9	2	3.47 m	3.24 m	2.73 m	1.55 m	616 µ	**249** µ	11.0
11_iwamoto[c]	2	3.73 m	10 µ	980 µ	**8** µ	295 µ	**8** µ	122.5
14[c]	2	15.35 m	57 µ	3.67 m	27 µ	282 µ	**22** µ	166.9
24_ieee_rts[c]	3	166.49 m	61 µ	5.34 m	**48** µ	590 µ	60 µ	89.0
30[c]	3	28.52 m	44 µ	6.70 m	**33** µ	1.29 m	37 µ	181.0
ieee30[c]	3	23.13 m	48 µ	8.26 m	46 µ	1.11 m	**44** µ	187.7
33bw[c]	2	15.17 m	**10** µ	10.22 m	12 µ	968 µ	13 µ	786.1
39[c]	5	24.43 m	**53** µ	7.30 m	54µ	1.60 m	66 µ	110.7
57	3	154.01 m	55.12 m	18.06 m	11.79 m	4.52 m	**924** µ	19.5
89pegase[c]	5	1.26	433 µ	12.88 m	364 µ	39.39 m	**165** µ	78.0
118	8	498.66 m	105.68 m	78.32 m	49.09 m	11.31 m	**5.90** m	13.3
145[c]	13	25.14	354 µ	23.13 m	**332** µ	4.62 m	336 µ	68.9
illinois200[c]	20	46.67 m	**116µ**	15.32 m	359 µ	459.57 m	**116** µ	132.1
300	30	6.16	372.88 m	312.35 m	97.38 m	272.78 m	**17.12** m	18.2
1354pegase	176	1.52	20.64 m	1.27	15.71 m	513.53 m	**3.80** m	333.7
1888rte	235	5.49	7.99 m	1.79	4.50 m	554.25 m	**2.58** m	693.6
2848rte	352	9.42	4.52 m	2.15	5.21 m	732.33 m	**3.29** m	653.5
2869pegase	305	469.69	2.00	8.40	326.75 m	8.07	**44.05** m	190.7
3120sp	231	659.27	5.04	30.78	3.49	471.23	**120.08** m	256.3
6470rte	745	52.80	169.64 m	16.26	34.71 m	6.74	**14.84** m	1,095.6
6495rte	757	647.15	161.39 m	14.62	37.40 m	5.04	**15.93** m	917.4
6515rte	757	85.10	144.25 m	14.74	36.92 m	5.77	**14.88** m	990.4
9241pegase	811	> 7,200.00	43.49	218.88	4.65	37.73	**481.43** m	454.7

The lowest running time is marked in bold
Observe that some run times are given in milli- or microseconds
[a]Numbers here were obtained from our interpretation of the respective approach
[b]Speedup of Ours+R compared to the faster of MILP and MFN
[c]Solved optimally by reduction rules

In the literature, most other solvers only consider networks consisting solely of propagating vertices. For comparison, we conducted the experiment on the same instances but with all vertices propagating, see Table 2. Observe that γ_P is lower when all vertices are propagating. With only propagating vertices in the input, our reduction rules could solve 13 out of 26 instances on their own. On all remaining instances, our solver combined with the reduction rules was the fastest and achieved a median speedup of 176.2.

Comparing the results in Tables 2 and 1, we observe that the solvers react differently to all vertices being propagating. While MILP and our solver without reductions are somewhat faster with some non-propagating vertices in the input, MFN and our solver with reductions perform better when all vertices are propagating. This leads to an interesting effect: Overall, MILP performs better than MFN when the input contains non-propagating vertices but considerably worse when all vertices are propagating. Nonetheless, our solver remains the fastest in both cases.

5.3 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 7 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 performs not much worse than our approach.

5.4 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 was 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.

Table 3 Comparison between our algorithm, MILP and MFN on the larger powersimdata US instances preprocessed with our reduction rules

Instance	Input		Our Solver			MFN+R			MILP+R	
	n	$\|Z\|$	γ_P	$\|\mathcal{F}\|$	t (s)	γ_P	$\|\mathcal{F}\|$	t (s)	γ_P	t (s)
Texas	2000	376	411	838	0.98	411	659	17.73	411	1.81
Western	10,024	4106	1825	2618	1.55	1825	2010	158.51	1825	2.16
Eastern	70,047	30,332	12,895	27,019	552.46	>12,789	>15,043	>10 h	>12,890	>10 h
USA	82,071	34,814	15,131	30,357	728.62	>14,124	>16,391	>10 h	>15,126	>10 h

The number of vertices is n, $|Z|$ is the number of non-propagating vertices and $|\mathcal{F}|$ is the size of the arising hitting set instance, if applicable. For the solvers, we report the power dominating number γ_P (or the best found lower bound) as well as the number of fort neighborhoods $\mathcal{F}$ and the run time

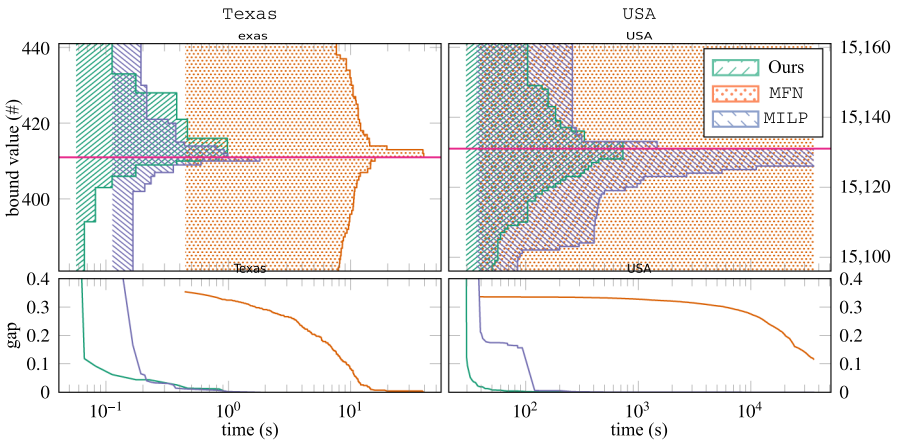
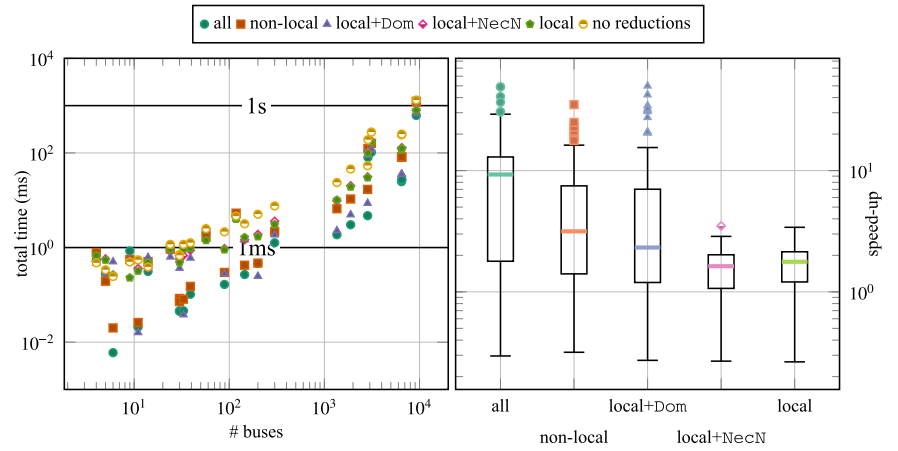


Fig. 7 Upper and lower bounds on the optimum value on the `Texas` and `USA` powersimdata instances with preprocessing by our reduction rules. We give the bounds reported by our solver and by `MFN`, both with added greedy upper bounds, as well as Gurobi for `MILP`. Lines and shaded areas each start at the time of the first respective bound. Note that the x axis uses a logarithmic scale (Color figure online)



(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

Fig. 8 Running times and speed-up of our algorithm with different subsets of the reduction rules on the pandapower instances. **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 (Color figure online)

Figure 8a 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.

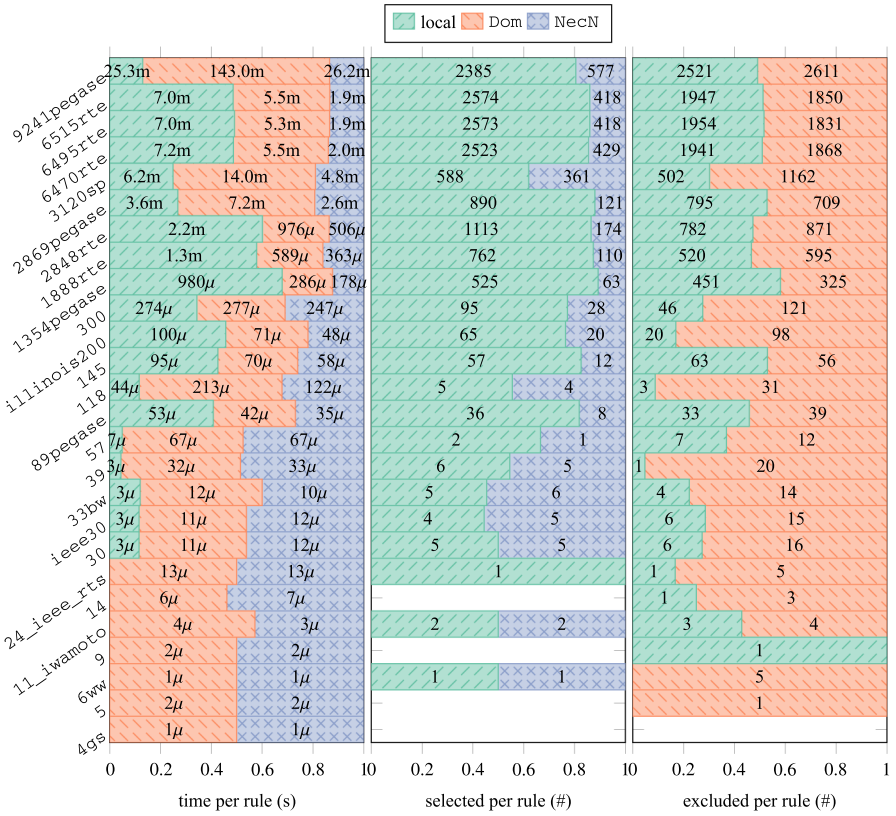


Fig. 9 Time taken by the local and non-local reduction rules and number of selected and excluded vertices by reduction rule. The numbers in the bar represent the absolute value for each rule and the bar width the number relative to the total time or number of vertices processed by all rules. A blank line means no vertices were selected or excluded, respectively (Color figure online)

Figure 8b 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.

Figure 9 shows the running times of the reduction rules and the numbers of selected and excluded vertices per rule for each instance. The time needed for the local rules is combined to compare them with Dom and NecN. The local rules also identify most of the selected vertices, but the effect of Dom is clearly visible when it comes to identifying vertices to exclude. This effect leads to the speed-up we see when using the reduction rules and thus offsets their higher running time.

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 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.

A Non-propagation for MILP Formulations and Fort Neighborhoods

A.1 MILP Formulation

There are several MILP formulations for SIMPLE- PDS, e.g. [16] and [7], but to the best of our knowledge these have not been applied to PDS-EXTENSION. We discuss how these formulations can be modified in a straightforward manner to solve the extension problem and to accommodate non-propagating vertices.

We use the formulation by Jovanociv and Voss [16] as a basis for our MILP. They represent each vertex with two variables x_v and s_v. The binary variable x_v represents whether a vertex is selected. The third variable $p_{v,w}$ represents a vertex v propagating to another vertex w. To prevent cycles, Jovanovic and Voss count the number of propagation steps on the path from a selected vertex in the variable s_v.

Non-propagating vertices can be accounted for by introducing an additional constraint forcing $p_{v,w} = 0$ if v is non-propagating. If a vertex v is excluded, we add a constraint $x_v = 0$ and if v is selected, we add a constraint $x_v = 1$.

While Jovanovic and Voss require s_v to be integral, we relax this constraint and allow continuous values instead. The steps only require a minimum offset but not integrality. There are also several other redundant constraints in the model. We did not remove all of them as we found in preliminary experiments that they improve the solver performance. We did remove one constraint, namely $p_{v,w} + p_{w,v} \leq 1$. This constraint follows from constraint (6) and seems to lead to a significant decrease in performance. For completeness we state the resulting model with all our modifications.

$$\text{Minimize} \sum_{v \in V} x_v \tag{1}$$

$$\text{s.t. } s_v \leq x_t + M(1 - x_t) \qquad \forall v \in V, t \in N[v] \tag{2}$$

$$s_v \leq M(x_v + \sum_{w \in N(v)} (x_w + p_{w,v})) \qquad \forall v \in V \tag{3}$$

$$\sum_{w \in N(v)} p_{w,v} \leq 1 \qquad \forall v \in V \tag{4}$$

$$\sum_{w \in N(v)} p_{v,w} \leq 1 \qquad \forall v \in V \tag{5}$$

$$s_v \geq s_t + 1 - M(1 - p_{w,v}) \qquad \forall v \in V, w \in N(v), t \in N[w] \setminus \{v\} \tag{6}$$

$$1 \leq s_v \leq |V| \qquad \forall v \in V \tag{7}$$

$$x_v = 1 \qquad \forall v \in X \tag{8}$$

$$x_v = 0 \qquad \forall v \in Y \tag{9}$$

$$p_{v,w} = 0 \qquad \forall v \in Z, w \in N(v) \tag{10}$$

$$x_v, p_{w,v} \in \{0, 1\} \tag{11}$$

Lemma 14 *Let G be a* PDS-EXTENSION*-instance. The MIP formulation has a solution with objective value k if and only G has a solution of size k.*

Proof Let $G = (V, E)$ be a PDS-EXTENSION-instance and let its MIP be defined as above. □

▷ Claim 15 (If G has a solution of size k, there is a satisfying assignment (x, s)) Let S be a minimum power dominating set of G and let $r_1, \ldots, r_\ell$ be a sequence of rule applications that observes G. Without loss of generality, we assume that all applications of the domination rule occur before the first application of the propagation rule. Every vertex is observed either by the domination rule or by the observation rule. If v is observed by the domination rule we set $x_v = s_v = 1$, otherwise we set $x_v = 0$ and thus constraint (2) and (7) to (9) are satisfied. Only if a vertex u observes another vertex v by the propagation rule, we set $p_{u,v} = 1$ and $s_v = 1 + \max\{s_w \mid w \in N[u] \setminus \{v\}\}$ which satisfies constraint (3). By default we set $p_{u,v} = 0$, in particular for all non-propagating vertices, satifying constraints (4) to (6). We only assign 1 or 0 to the variables, so constraint (11) is satisfied.

▷ Claim 16 (If the MIP has a solution of weight k, then G has a solution of size k) We construct a candidate solution $S = \{v \mid x_v = 1\}$. Let $V = v_1, \ldots, v_{|V|}$ be sorted by the value of s_v in ascending order. We show inductively that S is a solution of G by iterating the vertices sorted by their value of s_v in ascending order. In each step i we assume that there is a sequence of observation rules that observes all vertices up to v_{i-1} and we extend this sequence to observe v_i.

First, we apply the domination rule to all vertices in S, i.e. with $x_v = 1$. This observes $N[S]$. By constraints (2) and (7) all vertices in $N[S]$ must have $s_v = 1$. Note that by constraint (7) there is no vertex with $s_v < 1$.

Now let u be a vertex with $s_v > 1$. By constraint (2) we know that $N[u] \cap S$ is empty and thus constraint (3) implies that there is at least one other vertex $v \in N(u)$ with $p_{v,u} = 1$. Constraint (4) ensures that there is only one such v. Then constraint (6) states that $s_u > s_w$ for all vertices $w \neq u$ in the closed neighborhood $N[v]\setminus\{u\}$. By our induction hypothesis we thus know that all vertices in $N[v] \setminus \{u\}$ are observed, i.e. v is observed and has only one unobserved neighbor, u. We can thus apply the propagation rule and u becomes observed.

In the above formulation we account for propagation and selected and excluded vertices by introducing additional constraints. Instead of introducing additional constraints, we can modify the existing constraints to account for these additional properties. The resulting model contains fewer variables and constraints from the start.

A.2 Fort Neighborhoods

Both previous approaches using fort neighborhoods search for fort neighborhoods of minimum size [5, 22]. We adapt their ILP formulations for the two sub-problems of finding a hitting set and finding a violated fort to PDS-EXTENSION. Let $G = (V, E)$ be a PDS-EXTENSION-instance with selected vertices X, excluded vertices Y and non-propagating vertices Z. We introduce a variable s_v for each vertex, representing

whether that vertex is in the power dominating set. The following model finds a minimum hitting set of M that contains all vertices in X and no vertices in Y.

$$\text{Minimize} \sum_{v \in V} s_v \tag{12}$$

$$\text{s.t.} \sum_{v \in F} s_v \geq 1 \qquad \forall F \in M \tag{13}$$

$$s_v = 1 \qquad \forall v \in X \tag{14}$$

$$s_v = 0 \qquad \forall v \in Y \tag{15}$$

$$s_v \in \{0, 1\} \tag{16}$$

To find a violated fort neighborhood of minimum size, we adapt the method introduced by Smith and Hicks [22]. Let S be a set of selected vertices and let R be the set of vertices observed by exhaustive application of the observation rules. For each vertex we introduce two variables x_v and y_v which represent whether v is in the fort or in the neighborhood of the fort, respectively.

$$\text{Minimize} \sum_{v \in V} y_v \tag{17}$$

$$\text{s.t.} \sum_{v \in V} x_v \geq 1 \tag{18}$$

$$x_v = 0 \qquad \forall v \in R \tag{19}$$

$$x_v + \sum_{w \in N(v)} \geq x_u \qquad \forall u \in V \, \forall v \in N(u) \tag{20}$$

$$y_v \geq x_w \qquad \forall v \in V \, \forall w \in N[v] \tag{21}$$

$$x_v \in \{0, 1\} \tag{22}$$

Note that this model has a feasible solution if and only if $R \neq V$.

Acknowledgements A version of this work appeared on ESA 2023 Track B [4].

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the German Research Foundation (DFG) as part of the Research Training Group GRK 2153:Energy Status Data—Informatics Methods for its Collection, Analysis and Exploitation.

Declarations

Conflict of interest The authors have no Conflict of interest to declare that are relevant to the content of this article.

References

1. Aazami, A.: Domination in graphs with bounded propagation: algorithms, formulations and hardness results. J. Comb. Optim. **19**, 429–456 (2008). https://doi.org/10.1007/s10878-008-9176-7
2. Baldwin, T.L., Mili, L., Boisen, M.B., Adapa, R.: Power system observability with minimal phasor measurement placement. IEEE Trans. Power Syst. **8**, 707–715 (1993). https://doi.org/10.1109/59.260810
3. Binkele-Raible, D., Fernau, H.: An exact exponential time algorithm for power dominating set. Algorithmica **63**(1), 323–346 (2012). https://doi.org/10.1007/s00453-011-9533-2
4. Bläsius, T., Göttlicher, M.: An efficient algorithm for power dominating set. In: Li-Gørtz, I., Farach-Colton, M., Puglisi, S.J., Herman, G. (eds.) 31st Annual European Symposium on Algorithms, ESA 2023, September 4–6, 2023, Amsterdam, The Netherlands. volume 274 of LIPIcs. Schloss Dagstuhl–Leibniz–Zentrum für, pp. 21:1–21:15. Informatik, New York (2023). https://doi.org/10.4230/LIPICS.ESA.2023.21
5. Bozeman, C., Brimkov, B., Erickson, C., Ferrero, D., Flagg, M., Hogben, L.: Restricted power domination and zero forcing problems. J. Comb. Optim. **37**, 935–956 (2018). https://doi.org/10.1007/s10878-018-0330-6
6. Brimkov, B., Fast, C.C., Hicks, I.V.: Computational approaches for zero forcing and related problems. Eur. J. Oper. Res. **273**(3), 889–903 (2019)
7. Brimkov, B., Mikesell, D., Smith, L.: Connected power domination in graphs. J. Comb. Optim. **38**(1), 292–315 (2019). https://doi.org/10.1007/s10878-019-00380-7
8. Brueni, D.J.: Minimal PMU placement for graph observability: a decomposition approach (1993) http://hdl.handle.net/10919/45368
9. Brueni, D.J., Heath, L.S.: The PMU placement problem. SIAM J. Discret. Math. **19**, 744–761 (2005). https://doi.org/10.1137/S0895480103432556
10. Cai, L.M., Chen, J., Downey, R., Fellows, M.: On the structure of parameterized problems in NP. Inf. Comput. **123**(1), 38–49 (1995). https://doi.org/10.1006/INCO.1995.1156
11. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity, vol. 4. Springer, Berlin (2013). https://doi.org/10.1007/978-1-4471-5559-1
12. Guo, J., Niedermeier, R., Raible, D.: Improved algorithms and complexity results for power domination in graphs. Algorithmica **52**(2), 177–202 (2008). https://doi.org/10.1007/s00453-007-9147-x
13. Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual (2023). https://www.gurobi.com
14. Haynes, T.W., Hedetniemi, S.M., Hedetniemi, S.T., Henning, M.A.: Domination in graphs applied to electric power networks. SIAM J. Discrete Math. **15**(4), 519–529 (2002). https://doi.org/10.1137/S0895480100375831
15. Janota, M., Marques-Silva, J.: Solving QBF by clause selection. In: International Joint Conference on Artificial Intelligence (2015)
16. Jovanovic, R., Voss, S.: The fixed set search applied to the power dominating set problem. Expert Syst. **37**(6), e12559 (2020). https://doi.org/10.1111/exsy.12559
17. Kneis, J., Mölle, D., Richter, S., Rossmanith, P.: Parameterized power domination complexity. Inf. Process. Lett. **98**(4), 145–149 (2006). https://doi.org/10.1016/j.ipl.2006.01.007
18. Liao, C.-S., Lee, D.-T.: Power domination in circular-arc graphs. Algorithmica **65**(2), 443–466 (2013). https://doi.org/10.1007/s00453-011-9599-x
19. Mili, L., Baldwin, T, Adapa, R.: Phasor measurement placement for voltage stability analysis of power systems. In: 29th IEEE Conference on Decision and Control, vol.6, pp. 3033–3038 (1990). https://doi.org/10.1109/CDC.1990.203341
20. Row, D.D.: A technique for computing the zero forcing number of a graph with a cut-vertex. Linear Algebra Appl. **436**(12), 4423–4432 (2012)
21. Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid MaxSAT solver. In: International Conference on Theory and Applications of Satisfiability Testing (2016). https://doi.org/10.1007/978-3-319-40970-2_34

22. Smith, L.A., Hicks, I.V.: Optimal sensor placement in power grids: power domination, set covering, and the neighborhoods of zero forcing forts. arXiv, arXiv:2006.03460 (2020)
23. Thurner, L., Scheidler, A., Schäfer, F., Menke, J., Dollichon, J., Meier, F., Meinecke, S., Braun, M.: pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. IEEE Trans. Power Syst. **33**(6), 6510–6521 (2018). https://doi.org/10.1109/TPWRS.2018.2829021
24. Xu, G., Kang, L., Shan, E., Zhao, M.: Power domination in block graphs. Theor. Comput. Sci. **359**(1–3), 299–305 (2006). https://doi.org/10.1016/j.tcs.2006.04.011
25. Xu, Y., Myhrvold, N., Sivam, D., Mueller, K., Olsen, D., Xia, B., Livengood, D., Hunt, V., d'Orfeuil, B.R., Muldrew, D., Ondreicka, M., Bettilyon, M.: U.S. test system with high spatial and temporal resolution for renewable integration studies. In: IEEE Power and Energy Society General Meeting (PESGM), pp. 1–5 (2020). https://doi.org/10.1109/PESGM41954.2020.9281850