Power Domination in $\mathcal{O}^*(1.7548^n)$ using Reference Search Trees

Daniel Raible & Henning Fernau

Univ.Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany
{raible,fernau}@informatik.uni-trier.de

Abstract. The Power Dominating Set problem is an extension of the well-known domination problem on graphs in a way that we enrich it by a second propagation rule: Given a graph $G(V, E)$ a set $P \subseteq V$ is a power dominating set if every vertex is observed after we have applied the next two rules exhaustively. First, a vertex is observed if $v \in P$ or it has a neighbor in P. Secondly, if an observed vertex has exactly one unobserved neighbor u, then also u will be observed as well. We show that Power Dominating Set remains $\mathcal{NP}$-hard on cubic graphs. We designed an algorithm solving this problem in time $\mathcal{O}^*(1.7548^n)$ on general graphs. To achieve this we have used a new notion of search trees called reference search trees providing non-local pointers.

1 Introduction

We study an extension of Dominating Set. The extension originates not from an additional required property for the solution set (e.g., Connected Dominating Set) but by adding a second rule. To be precise we look for a vertex set, called power dominating set, such that every vertex is observed according to the next two rules:

Observation Rule 1 (OR1): A vertex in the power domination set observes itself and all of its neighbors.
Observation Rule 2 (OR2): If an observed vertex v of degree $d \geq 2$ is adjacent to $d-1$ observed vertices, then the remaining unobserved neighbor becomes observed as well.

By skipping the second rule we would exactly arrive at Dominating Set. The second rule is responsible for the non-local character of the problem as it implements a kind of propagation. Due to this propagation mechanism a vertex can observe another vertex at arbitrary distance. Also the sequence of OR2 applications can be arbitrary but leading to the same set of observed vertices. Indeed, many arguments relying on the locality of Dominating Set fail. There is no transformation to Set Cover and thus the algorithm of [3] cannot simply be modified. The problem occurs in the context of monitoring electric power networks. One wishes to place a minimum number of measurement devices at certain points in the network to measure the state variables (for example the

voltage magnitude and the current phase). In that sense OR2 stands for Kirchhoff's law. In this way, we arrive at the definition of the central problem:

Power Dominating Set (PDS)
Given: An undirected graph $G = (V, E)$, and the parameter k.
We ask: Is there a set $P \subseteq V$ with $|P| \leq k$ which observes all vertices in V with respect to the two observation rules OR1 and OR2?

Discussion of Related Results The study of PDS was initiated by Haynes et *al.* [7] where they showed $\mathcal{NP}$-hardness and gave a first polynomial time algorithm for trees. Guo et *al.* [6] and Kneis et *al.* [8] studied this problem independently with respect to parameterized complexity. They proved $W[2]$-hardness if the parameter is the size of the solution by reducing Dominating Set to PDS. As a by-product it turns out that PDS is still $\mathcal{NP}$-hard on graphs with maximum degree four and that there is a lower bound for any approximation ratio of $\Omega(\log n)$ modulo standard complexity assumptions. Additionally, they showed fixed-parameter tractability of PDS with respect to tree-width, where [6] also give a concrete algorithm. [6] achieve this by transforming PDS into a orientation problem on undirected graphs. The problem was also studied in the context of special graph classes like interval graphs [9] and block graphs [12] where linear time algorithms where obtained. Aazami and Stilp [1] improved the approximation lower bound to $\Omega(2^{\log^{1-\epsilon} n})$ and gave an $\mathcal{O}(\sqrt{n})$-approximation for planar graphs. On the other hand also domination problems have been studied in exact algorithmics. Fomin, Gradoni and Kratsch [3] gave a $\mathcal{O}^*(1.5137^n)$-algorithm for Dominating Set where they use the power of the measure and conquer approach. The currently fastest $\mathcal{O}^*(1.5134^n)$-algorithm by Rooj and Bodlaender [11] achieves this slight improvement by a new reduction rule. Fomin, Gradoni and Kratsch [4] showed that the variant Connected Dominating Set can be solved in time $\mathcal{O}^*(1.9407^n)$.

New Results First, we show that PDS remains $\mathcal{NP}$-hard for cubic graphs. As PDS is polynomial time solvable for max-degree-two graphs and $\mathcal{NP}$-hardness was shown for max-degree-four graphs [6, 8], this result closes the gap inbetween. Furthermore, this justifies to follow a branching strategy even in the case of cubic graphs. Note that it is not always true that generally $\mathcal{NP}$-hard graph problems remain $\mathcal{NP}$-hard for cubic graphs. Feedback Vertex Set is a problem where as with PDS cycles play a role (see [6]). But in contrast to general graphs, it is solvable in polynomial time on cubic graphs [10]. Secondly, we present an algorithm solving PDS in time $\mathcal{O}^*(1.7548^n)$, which breaks the trivial 2^n-barrier. The run time analysis proceeds in an amortized fashion using the measure and conquer approach (see [3]). Furthermore, we introduce the concept of a reference search tree. In an ordinary search tree we usuallay cut off branches due to local structural conditions. In a reference search tree we also will cut off branches if we can point to another node of the search tree where we can find no worse solutions. This node must not be a neighbor of the current node but can be anywhere in the search tree, as long as the overall search structure remains acyclic.

Terminology and Notation The *(open) neighborhood* of $v \in V$ is $N(v) = \{w \mid \{w, v\} \in E\}$ and the *closed neighborhood* $N[v] := N(v) \cup \{v\}$. A possible solution set will be denoted P. We call a vertex $v \in V$ *directly observed* by $u \in N(v)$ if u is in the solution, i.e., $u \in P$. The vertex $v \in V$ will be called *indirectly observed* by $u \in V$ if v is observed due to the application of OR2 onto u. An *(a,b)-branch* is a binary branch which reduces the problem measure by an amount of a in one part of the branch and by b in the other.

2 $\mathcal{NP}$-hardness of Planar Cubic Power Dominating Set

We will reduce Vertex Cover to planar cubic PDS. Due to [5] Vertex Cover remains $\mathcal{NP}$-complete on planar cubic graphs. For any planar cubic graph $G(V, E)$ and any $v \in V$ we can denominate the neighbors of v as follows: $N(v) = \{n_{v_1}, n_{v_2}, n_{v_3}\}$. The reduction works as follows: Given a planar cubic graph $G(V, E)$ introduce for every $v \in V$ the gadget T_v depicted in Figure 1, which consists of the vertices in the dotted square. For any $\{u, v\} \in E$ we can find $1 \leq b, c \leq 3$ such that $u = n_{v_b}$ and $v = n_{u_c}$. By introducing the edge $\{c_{v\,b}, c_{u\,c}\}$ we finally get $G'(V', E')$ which is planar and cubic.

Lemma 1. *G has a vertex cover of size $\leq k$ iff G' has a PDS of size $\leq k$.*

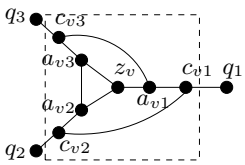

Fig. 1. The gadget T_v. The vertices q_1, q_2, q_3 correspond to vertices of the form $c_{z\,i}$ of some other gadget T_z such that $z \in V$.

According to Lemma 1 planar cubic PDS remains $\mathcal{NP}$-hard.

3 An Exact Algorithm for Power Dominating Set

3.1 Reference Search Trees

We will introduce a new kind of search scheme for combinatorial optimization problems. These problems can usually be modeled as follows. We are given a triple $(\mathcal{U}, \mathcal{S}, c)$ such that $\mathcal{U} = \{u_1, \ldots, u_n\}$ is called the *universe*, $\mathcal{S} \subseteq \mathcal{P}(\mathcal{U})$ is the *solution space* and $c : \mathcal{P}(\mathcal{U}) \to \mathbb{N}$ is the *value function*. Generally we are looking for a $S \in \mathcal{S}$ such that $c(S)$ is minimum or maximum. We then speak of a combinatorial minimization (maximization, resp.) problem. The general search space is $\mathcal{P}(\mathcal{U})$.
The *set vector (sv_Q)* of a set $Q \in \mathcal{P}(\mathcal{U})$ is a 0/1-vector indexed by the elements of $\mathcal{U}$ such that: $sv_Q[i] = 1 \iff u_i \in Q$. We write $sv_Q \in \mathcal{S}$ when we mean

$Q \in \mathcal{S}$. A *solvec* is a $0/1/\star$-vector. We define the following partial order $\preceq$ on solvecs s_1, s_2 of length n:

$$s_1 \preceq s_2 \iff \forall 1 \leq i \leq n : (s_1[i] = \star \Rightarrow s_2[i] = \star) \\ \wedge (s_1[i] = d \ (d \in \{0,1\}) \Rightarrow s_2[i] \in \{d, \star\}).$$

A *branching* is a directed tree $D(V,T)$ with root $r \in V$ such that all arcs are directed from the father-vertex to the child-vertex. For a vertex $u \in V$ the term ST_u refers to the sub-tree rooted at u.

Definition 1. *A* reference search tree (rst) *for a combinatorial minimization (maximization, resp.) problem $(\mathcal{U}, \mathcal{S}, c)$ is a directed graph $D(V, T \cup R)$ together with a injective function label $: V \to \{(z_1, \ldots, z_n) \mid z_i \in \{0, 1, \star\}\}$ with the following properties:*

1. *$D(V,T)$ is a branching.*
2. *$D(V, T \cup R)$ is acyclic.*
3. *Let $u, v \in V(D)$ then u is a descendant of v in $D(V,T)$ iff $label(u) \preceq label(v)$.*
4. *For any set vector sv_Q of a set $Q \in \mathcal{P}(\mathcal{U})$ with $Q \in \mathcal{S}$ and a vertex $v \in V(D)$ such that $sv_Q \preceq label(v)$ we have either one of the following properties:*
   (a) *There exists a leaf $z \in V(ST_v)$ such that $c(label(z)) \leq c(sv_Q)$ ($c(label(z)) \geq c(sv_Q)$, resp.) and $label(z) \in \mathcal{S}$.*
   (b) *There exists a vertex $x \in V(ST_v)$ such that there is exactly one arc $(x, y) \in R$ and we have that there is a $0/1$-vector h with $h \preceq label(y)$, $c(h) \leq c(sv_Q)$ ($c(h) \geq c(sv_Q)$, resp.) and $h \in \mathcal{S}$.*

How can a rst be exploited algorithmically? It is important to see that in a rst all the information for finding an optimal solution is included. Ordinary search trees can be defined by omitting item (b) of Definition 1. In a search tree we skip a solution s with $s \preceq u$ for a sub-tree ST_u if we can find a solution in ST_u which is no worse. In a rst we also have the possibility to make a reference to another subtree ST_f where such a solution could be found. In ST_f it might also be the case that we have to follow a reference once more. So, the only obstacle seems to be that, if we follow reference after reference, we end up in a cycle. But this is prevented by item 2. of Definition 1. An algorithm building up an rst can eventually benefit by cutting of branches and introducing references instead.

3.2 Annotated Power Dominating Set

In what follows we assume that the vertices of the given graph $G(V,E)$ are annotated. To be precise we have a function s which assigns a label to every vertex: $s : V(G) \to \{active, inactive, blank\}$. An *active* (*inactive*, resp.) vertex has already been determined to be (not to be, resp.) part of P. For a *blank* vertex this decision has been not made yet. We will abbreviate the three attributes by (a), (i) and (b). We also define $A := \{v \in V(G) \mid s(v) = (a)\}$, $I := \{v \in V(G) \mid s(v) = (i)\}$ and $B := \{v \in V(G) \mid s(v) = (b)\}$. For any given set $A \subseteq V(G)$ we can determine which vertices are already observed by applying exhaustively OR1 and

OR2. Due to this we introduce $s' : V(G) \to \{(o)bserved, (u)nobserved\}$ and set $O := \{v \in V(G) \mid s'(v) = (o)\}$ and $U := V(G) \setminus O$. The *state* of a vertex v is the tuple $(s(v), s'(v))$. During the course of the algorithm the states of the vertices (i.e., the labels s, s') will be modified in a way that they represent choices already made. We set $N^\star(v) := \{\{w, v\} \in E \mid s'(w) = (u)\}$ and $d^\star(v) := |N^\star(v)|$. $N^\star(v)$ represents the unobserved neighbors of v. Let $\Delta^\star(G) := \max_{v \in V \wedge s(v)=(b)} d^\star(v)$ and $M(G) = \{v \in B \mid d^\star(v) = \Delta^\star(G)\}$. We define $N^{(i)}(v) = N^\star(v) \cap I$ and $d^{(i)}(v) = |N^{(i)}(v)|$. We will write $d^\star_G(v)$, $N^\star_G(v)$, $d^{(i)}_G$, $N^{(i)}_G(v)$, $s_G(v)$ and $s'_G(v)$ when we are referring to a particular annotated graph G by which the functions are induced. We omit the subscript when it is clear from the context. A vertex $v \in V(G)$ such that $s'(v) = (o)$ and $d^\star(v) = 2$ will be called a *trigger*. A *triggered path* between $v_1, v_k \in V(G)$ with $s(v_1) = (b)$, $s(v_k) = (b)$ is a path $v_1, \ldots, v_k$ such that $s(v_i) = (b)$ and $d^\star(v_i) \leq 2$, or v_i is a trigger for $1 < i < k$. A *triggered cycle* is a triggered path with $v_1 = v_k$. Observe that for all $u \in O$ we have $d^\star(u) \neq 1$ due to OR2.

Algorithm In this section we present reduction rules and the algorithm. Their correctness and run time will be analyzed in the next section. We state the following reduction rules:

Isolated: Let $v \in O \cap B$ such that $d^\star(v) = 0$ then set $s(v) \leftarrow (i)$.

TrigR: Let $v \in V$ be a trigger and $s(v) = (b)$. Then set $s(v) \leftarrow (i)$.

Blank2: Let $v \in V(G)$ with $d^\star(v) \leq 2$, $v \in B \cap U$, $y \in N^\star(v)$ and $s(y) = (i)$. Then set $s(v) \leftarrow (i)$.

Obs3: Let $v \in V(G)$ such that $v \in B \cap U$, $d^\star(v) \leq 1$ and $y \in N(v)$ with $y \in I \cap O$ and $d^\star(y) \geq 3$. Then set $s(v) \leftarrow (i)$.

Trig2: Let $v \in V(G)$ such that $v \in B \cap U$ and $d^\star(v) \leq 1$. If there is a trigger u with $N^\star(u) = \{v, y\}$ and $y \in I \cap U$ then set $s(v) = (i)$.

Observe that for degree-2 vertices there is no valid contraction rule, see Figure 2(a). If we deleted u and connected x and y observation would propagate to z due to OR2. We are now ready to state Alg. 1:

Correctness We will prove correctness of Alg. 1 and the reduction rules using the concept of a reference search tree (see Definition 1). We have to define $\mathcal{U} := V(G)$, $\mathcal{S} := \{S \subseteq V(G) \mid S \text{ is a PDS for } G\}$ and $c(Y) = |Y|$ for every $Y \in \mathcal{P}(\mathcal{U})$. The function $label : V(D) \to \{(e_1, \ldots, e_n) \mid e_i \in \{(a), (i), (b)\}\}$ then expresses which vertices are no more blank, i.e., are active or inactive. Here (a) refers to 1, (i) to 0 and (b) to the $\star$-symbol defined in the function *label* of Definition 1. According to this we set $\bar{(a)} = (i)$ and $\bar{(i)} = (a)$.
The nodes of the rst $V(D)$ represent choices made concerning the blank vertices of $V(G)$. These choices can be due to branching or to applying reduction rules. Hence there is a 1-to-1 correspondence between $V(D)$ and the application of reduction rules and branchings. According to this we will speak of *full nodes*

Algorithm 1 An exact algorithm for Power Dominating Set

Apply OR2 exhaustively.
Apply **Isolated**, **TrigR**, **Blank2**, **Obs3** and **Trig2** exhaustively.
Select form $M(G)$ a vertex v according to the priorities:
  a) $s(v) = (u)$. {*We prefer unobserved vertices*}
  b) $d^{(i)}(v) < d^{\star}(v)$ {*We prefer vertices such that not all neighbors are inactive*}
if $d^{\star}(v) \geq 4$ **then**
  Branch on v by setting 1) $s(v) \leftarrow (i)$ and 2) $s(v) \leftarrow (a)$ in either of the branches.
else if $d^{\star}(v) = 3$ **then**
  Branch on v: 1) $s(v) \leftarrow (i)$ and 2) $s(v) \leftarrow (a)$ and for all $u \in N^{\star}(v)$ with $s(u) = (b)$ set $s(u) \leftarrow (i)$.
else if $d^{\star}(v) = 2$ **then**
  Branch on v by setting 1) $s(v) \leftarrow (i)$ and 2) $s(v) \leftarrow (a)$ in either of the branches.

and *flat nodes*, i.e. full nodes have two children in $D(V,T)$, flat nodes only one. In particular, nodes where reference pointers start are flat.

If we encounter a vertex $v \in V(G)$ with $s(v) = (i)$ in the current node q of the search tree we can find a second node $d_v \in V(D)$ which represents the choice made on v. That is we must have that $label(q) \preceq label(d_v)$. We can find d_v by simply going up the search tree starting from q. We sometimes indicate this relation by writing d_v^q, whereas we omit the superscript where it is clear from the context.

Prerequisites for Correctness Proofs The correctness proofs proceed to some extent in a graphical way. For this we draw the branching (the search tree without references) $D(V,T)$ in the plane with x- and y-coordinates. If u is a point in the plane then $pos_x(u)$ denotes its x- and $pos_y(u)$ its y-coordinate. It is possible to draw $D(V,T)$ satisfying three properties.

- Firstly, if $v \in V(D)$ is a father of $u \in V(D)$ then $pos_y(v) > pos_y(u)$.
- Secondly, let $v \in V(D)$ have two children $u_v, u_{\bar{v}}$, i.e., it is a full node. $u_{\bar{v}}$ corresponds to the branch where we set $s(y) = (i)$, in u_v we decided $s(y) = (a)$ for some $y \in V(G)$. We want D to be drawn such that for all $z \in ST_{u_{\bar{v}}}$ we have $pos_x(v) > pos_x(z)$ and for all $z' \in ST_{u_v}$ we have $pos_x(z') > pos_x(v)$. Hence we may speak of $u_{\bar{v}}$ as the left and u_v as the right child of v. According to this we will refer to them as $l(v)$ and $r(v)$, respectively.
- Thirdly, let $v \in V(D)$ be a flat node with its only child v_c. Then we require that $pos_x(v) = pos_x(v_c)$.

The subsequent correctness proofs proceed as follows: Every time we skip a possible solution we show that we can insert a reference to some node $u \in V(D)$ of the search tree such that we can find a solution z with $label(z) \preceq label(u)$ which is no worse. Additionally, we show that these references always point from the left to the right (with respect to the x-coordinate) in the drawing of $D(V,T)$. This way we assure acyclicity of the final rst $D(V, T \cup R)$ which is implicitly built up by the algorithm.

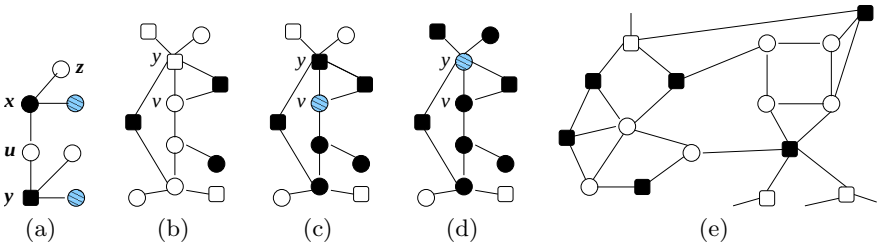

Fig. 2. Filled vertices are observed, white vertices are unobserved. Round vertices are blank, square vertices are inactive. Shaded vertices are active.

Lemma 2. *Let us fix an annotated PDS instance $G(V, E)$ that corresponds to some node $q \in V(D)$ in the search tree. Let $u \in V(G)$ with $u \in I \cap O$ and $d^\star_G(u) \geq 3$. Then $d^q_u \in D(V, T)$ is a full node.*

Proof. Suppose the contrary. Due to $d^\star_G(u) \geq 3$ none of the reduction rules in Alg.1 have set $s(u) \leftarrow (i)$ and thus there is no reference starting in u. The only remaining possibility is the second part of the branch in step 7 of Alg. 1. Now suppose by setting $s(v) \leftarrow (a)$ for some $v \in V(G)$ the algorithm has set also $s(u) \leftarrow (i)$ and $s'(u) = (o)$ implicitly. We now examine the situation right before this happened. This situation is reflected by some annotated graph $G'(V, E)$. We must have $d^\star_{G'}(v) = 3$ and $s'_{G'}(u) = (u)$. Suppose at this point $s'_{G'}(v) = (u)$. From this it follows that $d^\star_{G'}(u) \geq 4$ due to our premise. But this contradicts the choice of v as branch vertex. Therefore we must have $s'_{G'}(v) = (o)$. But once more this contradicts the choice of v since we have $s'_{G'}(u) = (u)$ at that point (as step 7 applied u should have been observed by v directly). □

Lemma 3. *Applying* **Blank2**, **Obs3**, **Trig2**, **TrigR**, **Isolated** *and step 7 of Alg. 1 is correct.*

Proof. We will prove the following: 1) for every vertex $v \in V(G)$ with $s(v) = (i)$ either d_v is a full node or there is a vertex h such that d_h is a full node and we inserted a reference $(v, r(d_h))$ such that if there is a solution with $s(v) = (a)$ at that point in the search tree we can find a no worse one z with $label(z) \preceq label(r(d_h))$. 2) Every reference is pointing from the left to the right in the drawing of $D(V, T)$. As step 7 makes use of this fact, it will be proven in parallel. The proof is by induction on the height s of the search tree. In case $s = 0$ nothing is to show. If $s > 0$ we will distinguish between the different operations:

Blank2 Let q be the current search tree node and, w.l.o.g.,
$label(q) = (e_1, \ldots, e_{l-1}, (b), \ldots, (b))$ such that e_l corresponds to v and e_1 to y (with v and y we are referring to the definition of **Blank2**). Suppose **Blank2** applies to v and d^q_y is a full node (see Figure 2(b)). Suppose there is a solution
$k := (e_1, \ldots, e_{l-1}, (a), e_{l+1}, \ldots, e_n) \preceq label(q)$ as indicated in Figure 2(c). Then also $k' := (\bar{e}_1, \ldots, e_{l-1}, (i), e_{l+1}, \ldots, e_n)$ is a solution (due to $d^\star(v) \leq 2$ and OR2, see Figure 2(d)). Hence we insert a reference $(q, r(d^q_y))$ as $k' \preceq label(r(d^q_y))$

which is pointing from the left to the right. This reference means that we can find a no worse solution compared to k. We find this solution in the sub search tree $ST_{r(d_y^q)}$ as $label(k') \preceq label(r(d_y^q))$ or else we have to follow another reference to the right. Therefore we can skip the possibility of setting $s(v) \leftarrow (a)$.
If d_y^q is a flat node than due to the induction hypothesis there must be a reference (d_y^q, h') such that $h' \in V(D)$ is the right child of some $h \in V(D)$ which is a full node and (d_y^q, h') points from the left to the right. We can rule out the solution k again as k' is no worse. Due to (d_y^q, h') also k' is skipped as there must be a alternative solution z with $label(z) \preceq h'$ such that z is no worse than k'. Thus we can insert the reference (q, h') pointing also from the left to the right.

Trig2 The proof is completely analogous to the first item.

Obs3 From Lemma 2 we have that y is a full node. The correctness follows now analogously to the first part of the first item.

Step 7 We only have to consider the second part of the branch: $s(v) \leftarrow (a)$ and for all $u \in N^\star(v)$ with $s(u) = (b)$ set $s(u) \leftarrow (i)$. Let $N^\star(v) = \{a, b, c\}$. We make a case distinction concerning $d^{(i)}(v)$. Let $q \in D(V)$ be the current search tree node before branching and let
$label(q) = (e_1, \ldots, e_l, (b), \ldots, (b))$, where the entries e_1 corresponds to v, e_2 to a and e_3 to b. Therefore we have $e_1 = (b)$. Assume there is a PDS $P \ni v$ with $sv_P \preceq label(q)$.

$d^{(i)}(v) = 0$ If $|P \cap N^\star(v)| \in \{2, 3\}$ then also $P \setminus \{v\}$ is a PDS due to OR2. If w.l.o.g $P \cap N^\star(v) = \{a\}$ then also $P' := P \setminus \{v\} \cup \{b\}$ is a PDS where P' is covered by the first part of the branch.

$d^{(i)}(v) = 1$ The only cases $|P \cap N^\star(v)| = 1$ and $|P \cap N^\star(v)| = 2$ can be handled analogously to the case $d^{(i)} = 0$.

$d^{(i)}(v) = 2$ W.l.o.g. $P \cap N^\star(v) = \{a\}$. Assume there is a PDS corresponding to $k := ((a), (a), (i), e_4, \ldots, e_n) \preceq label(q)$; $k' := ((i), (a), (a), e_4, \ldots, e_n)$ is then a solution, too. Suppose d_b^q is a full node. Then $k' \preceq label(r(d_b^q))$ and hence we insert a reference $(q, r(d_b^q))$. If d_b^q is a flat node there must be a reference (d_b^q, h'). Then insert (q, h').

The inserted references are pointing all from the left to the right in the drawing of $D(V, T)$. This ensures acyclicity of $D(V, T \cup R)$. □

Note that the reduction rules treated in Lemma 3 are not valid on their own. They are only correct because they are referring to solutions which Alg. 1 definitely will consider. In other words, if we are given an annotated graph G, where the annotation is not due to Alg. 1 we cannot apply these reduction rules.

Run Time Analysis We define the following sets: $\hat{I} = \{v \in I \mid s'(v) \neq (o), \exists u \in N^\star(v) : s(u) = (b)\}$, $\hat{O} = (O \cap B)$, $\hat{B} = B \cap U$. Here $\hat{I}$ comprises the inactive vertices, which are not observed such that they have at least one neighbor which is blank. In $\hat{O}$ we find the observed vertices for which we have not yet decided if there are active or not. Also for any $v \in \hat{O}$ we have $d^\star(v) \geq 2$

(**Isolated**). $\hat{B}$ contains the unobserved blank vertices. The measure we use in our run time is the following one:

$$\varphi(G) = |\hat{B}| + \beta \cdot |\hat{O}| + \gamma \cdot |\hat{I}| \text{ with } \beta = 0.51159, \gamma = 0.48842$$

We will now analyze the different branchings in Alg. 1. In general we can find integers ℓ, k with $\ell + k = d^\star(v)$ such that $\ell = |N^\star(v) \cap \hat{I}|$ and $k = |N^\star(v) \cap \hat{B}|$.

$d^\star(v) \geq 4$: The first case is when we have chosen a vertex v with $d := d^\star(v)$. We will explicitly analyze the case when $d = 4$. We show that any case occurring for $d > 4$ is run time upper bounded by some case when $d = 4$. First we will distinguish between the circumstances that $s'(v) = (o)$ and $s'(v) = (u)$.

$s'(v) = (u)$ In the branch where we set $s(v) \leftarrow (a)$ we get a reduction in $\varphi(G)$ of $1 + \ell \cdot \gamma + k \cdot (1 - \beta)$. This is due to v becoming observed and active, the vertices in $N^\star(v) \cap \hat{I}$ becoming inactive and observed and $N^\star(v) \cap \hat{B}$ becoming observed and blank. In the branch setting $s(v) \leftarrow (i)$ we reduce $\varphi(G)$ by at least $(1 - \gamma)$ (we obtain a greater reduction if v drops out of $\hat{I}$). As $(1 - \beta) < \gamma$ the worst case is the branch $(1 + 4(1 - \beta), 1 - \gamma)$ which is upper bounded by $\mathcal{O}^*(1.6532^n)$.

$s'(v) = (o)$ In the branch where we set $s(v) \leftarrow (a)$ we get a reduction in $\varphi(G)$ of $\beta + \ell \cdot \gamma + k \cdot (1 - \beta)$. Here we get only a reduction of β from v as it is already observed. In case $s(v) \leftarrow (i)$ the reduction is again β as v drops out of $\hat{O}$. As $(\beta + 4(1 - \beta), \beta)$ is the worst branch we have a upper bound of $\mathcal{O}^*(1.7548^n)$.

We examine now cases with $d > 4$. Here the worst case is analogously when $k = d$. But it is also no better as the case when $k = 4$ and $d = 4$, which was already considered.

$d^\star(v) = 3$: We first focus on the case where $\ell \leq 2$. As we get a reduction of one for every vertex in $N^\star(v) \cap \hat{B}$ the worst case is when $\ell = 2$. Now if $s'(v) = (u)$ then this results in a $(2 + 2 \cdot \gamma, 1 - \gamma)$ branching. If $s'(v) = (o)$ we have a $(\beta + 2 \cdot \gamma + 1, \beta)$-branching. $\mathcal{O}^*(1.7489^n)$ is an upper bound for both.

Now due to the priorities in step 4 of Alg. 1 we select a vertex v such that $\ell = 3$ with least priority. We first examine the case where $s'(v) = (u)$ and $\ell = 3$. Now suppose for all $u \in N^\star(v)$ we have $N^\star(u) \cap B = \{v\}$ (✪). Then in the branch $s(v) \leftarrow (i)$ we get an additional amount of $3 \cdot \gamma$. This is due to the fact that the vertices in $N^\star(v)$ will drop out of $\hat{I}$. Hence, we have a $(1 + 3 \cdot \gamma, (1 - \gamma) + 3 \cdot \gamma)$ branch.
Conversely, there is a $u \in N^\star(v)$ with $N^\star(u) \cap B = \{v, u_1, \ldots, u_s\}$ and $s \geq 1$ (✲). If $s'(u_1) = (o)$ then due to **TrigR** and the choice of v we have $d^\star(u_1) = 3$. In $s(v) \leftarrow (a)$ u will become a trigger and is reduced away from $\varphi(G)$ due to **TrigR**. This means we have a $(1 + 3 \cdot \gamma + \beta, 1 - \gamma)$ branch.
If $s'(u_1) = (u)$ then we have $d^\star(u_1) = 3$ due to **Blank2** and the choice of v. Also it holds that $d^{(i)}(u_1) = 3$ again by the choice of v. Hence in $s(v) \leftarrow (a)$ the $\star$-degree of u_1 drops by one. Therefore **Blank2** applies on u_1 and it will not appear in $\varphi(G)$ anymore which leads to a $(2 + 3 \cdot \gamma, 1 - \gamma)$ branch. $\mathcal{O}^*(1.6489^n)$

upper bounds both possibilities.
The second possibility for v is $s'(v) = (o)$, yielding a $(\beta + 3 \cdot \gamma, \beta + 3 \cdot \gamma)$-branch for case ✪. In case of ✲, $s'(u_1) = (o)$ is necessary or otherwise, we contradict the choice of v ($d^\star(u_1) \geq 3$), or **Blank2** applies to u_1 ($d^\star(u_1) \leq 2$). Again we have $d^{(i)}(u_1) = 3$. Hence, this gives a $(2\beta + 3\gamma, \beta)$-branch, as u_1 becomes a trigger. An upper bound for both is $\mathcal{O}^*(1.7488^n)$.

$d^\star(v) \leq 2$:
Lemma 4. *In step 10 of Alg.1 we have:*

1. *For all $u \in V$ with $d^\star(u) \geq 3$ it follows that $s(u) = (i)$.*
2. *Let $v \in V(G)$ with $s(v) = (b)$ and $s'(v) = (u)$ chosen for branching then:*
   (a) *For all $u \in N^\star(v) : s(u) = (b)$.*
   (b) *If $d^\star(v) \leq 1$ then for all $u \in N(v) \setminus N^\star(v) : s(u) = (i)$ and $d^\star(u) = 2$.*
3. *$O = O \cap I$. ($O \cap B = \emptyset$, alternatively).*

Proof. 1. Otherwise, we have a contradiction to the choice of v. *2(a)* Otherwise, **Blank2** applies. *2(b)* Note that $s'(u) = (o)$. Suppose $s(u) = (b)$ then either **TrigR** applies ($d^\star(u) = 2$) or we have a contradiction to the choice of v ($d^\star(u) \geq 3$). From $s(u) = (a)$ it follows that $s'(v) = (o)$, a contradiction. If we had $d^\star(u) \geq 3$ and $s(u) = (i)$ then **Obs3** applies. This contradicts $s(v) = (b)$.
3. Let $u \in O \setminus I$ then $d^\star(u) \in \{0, 1, 2\}$ is ruled out by **Isolated**, **OR2** and **TrigR**. If $d^\star(u) \geq 3$ then from item 1. it follows that $u \in I$, a contradiction. □

Let v be the vertex chosen in step 9 of Alg.1. Let $\tilde{G} := G[B]$ and note that $B = \hat{B}$ due to Lemma 4.3. $\tilde{G}$ consists of paths and cycles formed by vertices in $\hat{B}$ due to Lemma 4.2a and the fact that for all $z \in B$ we have $d^\star(z) \leq 2$ (see Figure 2(e)). v belongs to one of those components. Explore G the following way:

1. For all $u \in B$ set $visited(u) \leftarrow f$.
2. If there is $u \in N^\star(v)$ with $visited(u) = f$ then set $visited(v) \leftarrow t$ and $v \leftarrow u$.
3. If there is $t \in N(v)$ with $t \in O$ (due to Lemma 4.2b u is a trigger) such that $d^\star(t) = \{v, u\}$ and $visited(u) = f$ then set $visited(v) \leftarrow t$ and $v \leftarrow u$.
4. If one of the steps 3 or 4 applied goto 2.. Else set $visited(v) \leftarrow t$ and stop.

Let $W := \{u \in B \mid visited(u) = t\}$. W comprises the visited vertices in $\hat{B}$. Either W is path or a cycle containing at least two vertices from $\hat{B}$ (as long $|V(G)| > 1$ and due to Lemma 4.2b). Either v has a blank neighbor or it has a trigger as neighbor. Now observe that any vertex in W is equally likely to be set active: Once there is an active vertex from W the whole vertex-set W will be observed due to OR2. Also any additional trigger $t' \notin W$ which is a neighbor of some $v' \in W$ only depends on v' being observed.
Considering the branch $s(v) \leftarrow (a)$ due to exhaustively applying OR2 for any $v' \in W$ we have $N(v') \subseteq O$ afterwards. Hence W will drop out of $\hat{B}$ but will also not be included in $\hat{O}$ due to **Isolated**. Thus there is a reduction of $|W| \geq 2$.
In case $s(v) \leftarrow (i)$ due to applying **Blank2** and **Trig2** we have that $W \subset I \setminus \hat{I}$ (Lemma 4.2a/2b) and thus a reduction of $|W|$. Summing up we have a $(2, 2)$ branch which we upper bound by $\mathcal{O}^*(1.415^n)$.

Note that the instances occurring at this point of Alg.1 are still $\mathcal{NP}$-hard to solve. There is a simple reduction from CUBIC PDS. For any vertex v, create a cycle C_v of length three. If $\{u, v\} \in E$, connect free vertices $x \in C_v$ and $y \in C_u$ with an inactive trigger. So, we have no alternative to continuing the branching.

Theorem 1. POWER DOMINATING SET *can be solved in time* $\mathcal{O}^*(1.7548^n)$.

We like to comment that Alg. 1 achieves a run time of $\mathcal{O}^*(1.6212^n)$ on cubic graphs. This can be seen by modifying the general analysis. Simply choose $\beta = 0.8126$ and $\gamma = 0.3286$ and skip the part where $d^\star(v) \geq 4$.

4 Conclusion and Further Perspectives

Speed-Up With Exponential Space. We precompute optimal solutions for all vertex induced subgraphs with no more than $\omega = 0.1103n$ vertices. For any subgraph G_S we create $2^{\omega n}$ instances by deciding for all $v \in V(G_S)$ if they are observed yet or not. By solving each of these instances by brute force, we spend $4^{\omega n}$ steps for any induced subgraph with predetermined observation pattern. Thus we need $\mathcal{O}^*(\binom{n}{\omega n}4^{\omega n}) \in \mathcal{O}^*(1.6493^n)$ steps for building up a table of size $\mathcal{O}(1.5275^n)$. Let $R = V(G) \setminus \{v \in O \mid N[v] \subseteq O\}$. Once we arrived at a graph G with $|R| \leq \omega n$ in Alg. 1, we can look up the rest of the solution by inspecting the table entry which is determined by $G[R]$ and $R \cap O$. Thus Alg. 1 will run in $\mathcal{O}^*(1.7548^{(1-\omega)n}) \subseteq \mathcal{O}^*(1.6493^n)$. It is important to notice that we ignored the fact that there might be active and inactive vertices in $G[R]$. The correctness follows from the fact that the state of observation of $G[V \setminus R]$ is independent of how $G[R]$ is observed. Also any solution for $G[R]$ which ignores the labels active and inactive cannot be worse than one that does not. By choosing $\omega = 0.092972$ the same algorithm solves CUBIC PDS in $\mathcal{O}^*(1.55^n)$ steps using $\mathcal{O}(1.4533^n)$ space.

Notice that this type of speed-up relies on the fact that no branching or reduction rule ever changes the (underlying) graph itself, but rather, the existing graph is annotated. This property is also important when designing improved algorithms with the help of reference search trees, since it might be tricky to argue to find a solution not worse than the ones to be expected in a particular branch of a search tree somewhere else in the tree, when the instance is (seemingly) completely changed.

Résumée. In this paper, we designed an exact algorithm for POWER DOMINATING SET consuming $\mathcal{O}^*(1.7548^n)$ time. To achieve this we made intensive use of the concept of a reference search tree. This means that we where able to cut off branches by referring to arbitrary locations in the search tree where one can find equivalent solutions. Maybe the term search-DAG expresses this property also quite well. In the long version of [2] we already applied the concept of an rst successfully. We proved the correctness of a reduction rule whose application was critical for the run time. We expect that we can exploit reference search trees further by designing exact algorithms for non-local problems or improving existent ones (e.g., CONNECTED DOMINATING SET/VERTEX COVER or MAX

Internal Spanning Tree). For this kind of problems we are not allowed to delete vertices due to selecting vertices into the solution or not. We rather have to label them. Many algorithms come to decisions by respecting them. We rather try to make use of them. Let x be a labeled vertex not selected into the solution and y an unlabeled vertex. Suppose by re-labeling x (taking x into the solution) and excluding y from the solution we have a solution which is no worse to the possibility of taking y into the solution. Then we can skip this last possibility by inserting a reference. We imagine that this arguing is also possible when several vertices are involved.

References

1. A. Aazami and M. D. Stilp. Approximation algorithms and hardness for domination with propagation. In *APPROX-RANDOM*, volume 4627 of *LNCS*, pages 1–15. Springer, 2007.
2. H. Fernau and D. Raible. Exact algorithms for maximum acyclic subgraph on a superclass of cubic graphs. In *WALCOM*, volume 4921 of *LNCS*, pages 144–156. Springer, 2008, long version available as Technical Report 08-5, Technical Reports Mathematics / Computer Science, University of Trier, Germany, 2008.
3. F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. In *ICALP*, volume 3580 of *LNCS*, pages 191–203. Springer, 2005.
4. F. V. Fomin, F. Grandoni, and D. Kratsch. Solving connected dominating set faster than 2^n. In *FSTTCS*, volume 4337 of *LNCS*, pages 152–163. Springer, 2006.
5. M. R. Garey, D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, vol. 32, issue 4, pages 237–267, 1976.
6. J. Guo, R. Niedermeier, and D. Raible. Improved algorithms and complexity results for power domination in graphs. In *FCT*, volume 3623 of *LNCS*, pages 172–184. Springer, 2005.
7. T. W. Haynes, S. Mitchell Hedetniemi, S. T. Hedetniemi, and M. A. Henning. Domination in graphs applied to electric power networks. *SIAM J. Discrete Math.*, 15(4):519–529, 2002.
8. J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Parameterized power domination complexity. *Inf. Process. Lett.*, 98(4):145–149, 2006.
9. C.-S. Liao and D.-T. Lee. Power domination problem in graphs. In *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 818–828, 2005.
10. E. Speckennmeyer. On feedback vertex sets and nonseparating independent sets in cubic graphs. *Journal of Graph Theory*, 3:405–412, 1988.
11. J. M. M. van Rooij and H. L. Bodlaender. Design by measure and conquer, a faster exact algorithm for dominating set. In *STACS 2008*, volume 08001 of *Dagstuhl Seminar Proceedings*, pages 657–668. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.
12. G. Xu, L. Kang, E. Shan, and M. Zhao. Power domination in block graphs. *Theor. Comput. Sci.*, 359(1-3):299–305, 2006.

Appendix: Omitted proofs

$\mathcal{NP}$-hardness of the cubic case

Before proving Lemma 1, we exhibit some properties of a PDS P for G'.

Lemma 5. *If G' has a PDS P then there is also a PDS P' with $|P'| \leq |P|$ such that for all gadgets T_v we have: a) $|V(T_v) \cap P'| \leq 1$, b) if $|V(T_v) \cap P'| = 1$ then $V(T_v) \cap P' = \{z_v\}$ and $\{z_v\}$ is a PDS for $G'(N[T_v])$, c) if $|V(T_v) \cap P'| = 0$ then a_{vi} is indirectly observed by c_{vi}, $1 \leq i \leq 3$, and z_v is indirectly observed by a_{v1}.*

Proof. Let P be an arbitrary PDS for G' such that $r := |P \cap (\cup_{v \in V} z_v)|$ is maximum. Assume that $T_v \cap P \neq \emptyset$ but $z_v \notin P$. Set $P' = (P \setminus V(T_v)) \cup \{z_v\}$. Independently of the vertices in $P' \setminus \{z_v\}$, due to $z_v \in P'$ and by exhaustively applying OR2, the vertices a_{vi}, q_i and c_{vi} $(1 \leq i \leq 3)$ will be observed. As $G' \setminus V(T_v)$ only interacts via the q_i's with T_v and as they are also observed, P' must be a PDS. But we have $|P' \cap (\cup_{v \in V} z_v)| > r$ which is a contradiction. This shows $a)$ and $b)$. To show $c)$ observe the following. If $|G[T_v] \cap P'| = 0$ then T_v must be observed by using OR2 and therefore there must be a c_{vi} which is indirectly observed. If only one of the c_{vi}'s is indirectly observed (necessarily by q_i) then observation cannot propagate further by OR2 as any c_{vi} would have two unobserved neighbors. To be explicit, in case c_{v1} is observed the two vertices are a_{v1}, c_{v2}, in case c_{v2} we have a_{v2}, c_{v1} and in case c_{v3} the vertices a_{v3}, a_{v1} are unobserved. Thus there are unobserved vertices in T_v and P is not a PDS. Now we must rule out that there are exactly two vertices c_{vi}, c_{vj} being each observed by q_i and q_j. There are exactly three possibilities for this which are depicted in Figures 3(a), 3(b) and 3(c) where we also applied OR2 exhaustively. In any case some unobserved vertices remain. Note that if in Figure 3(c) also c_{v3} is observed then an OR2 application is triggered on a_{v_1} which observes z_v. Then due to another OR2 application on c_{v3} observation reaches a_{v3} and hence T_v is observed showing $c)$.

Proof (Lemma 1). Let S be a vertex cover with $|S| \leq k$ for G. Let $V = \{v_1, \ldots, v_n\}$. We want to point out that for every vertex in $u \in V \setminus S$ we have $N(u) \subseteq S$. Otherwise, S is not a vertex cover. Now construct a PDS P the following way. For every $v \in S$ add the vertex z_v of T_v to P. Observe that any gadget T_v with $v \in P$ is now completely observed after applying OR2 exhaustively. We also can apply OR2 to the q_i's such that their neighbors outside T_v will be observed. If $v \notin S$ then due to $N(v) \subseteq S$ and our last observation we have that any a_{vi} is indirectly observed by q_i. Then by OR2 observation propagates to all other vertices in T_v analogously as in Lemma 5c).
Let P be a PDS with $|P| \leq k$ for G'. Due to Lemma 5 we assume that, w.l.o.g., $P = \{z_1, \ldots, z_\ell\}$ with $\ell \leq k$. Now suppose that $VC = \{v_1, \ldots, v_\ell\}$ is not a vertex cover in G. This means there is an edge $\{u, v\} \cap VC = \emptyset$. There must be some $c_{v\,b}$ and $c_{u\,c}$ $(1 \leq b, c \leq 3)$ with $\{c_{v\,b}, c_{u\,c}\} \in E_{G'}$, see Figure 3(d). Since $u, v \notin P$ Lemma 5c) states that $c_{v\,b}$ is indirectly observed by $c_{u\,c}$ and vice versa. This is a contradiction to the definition of OR2.

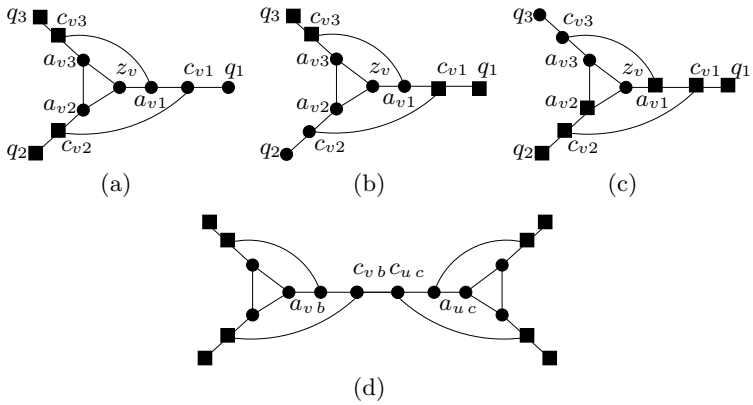

Fig. 3. Observed vertices are depicted as filled squares.

Regarding the correctness of TrigR

Here, the detailed discussion can be found: **TrigR** Let $q \in D(V)$ be the current search tree node before applying **TrigR** and let, w.l.o.g., $label(q) = (e_1, \ldots, e_l, (b), \ldots, (b))$, e_1 corresponds to y and e_{l+1} to v. Let P be a solution which contains v such that $sv_P \preceq label(q)$. Suppose that w.l.o.g. $s(y) = (b)$. Then P could be replaced by $P' = P \setminus \{v\} \cup \{y\}$. As we have $s(y) = (b)$ we will find a solution which is no worse later in the sub search tree ST_q. Now suppose $s(x) = s(y) = (i)$. If there is a solution corresponding to $k := (e_1, \ldots, e_l, (a), e_{l+2}, \ldots, e_n) \preceq label(q)$ then also $k' := (\bar{e}_1, \ldots, e_l, (i), e_{l+1}, \ldots, e_n)$ is one which is no worse.

If d_y^q is a full node we have that $k' \preceq label(r(d_y^q))$ and we insert a reference $(q, r(d_y^q))$. If d_y^q is a flat node there is a reference (d_y^q, h'). Analogously as in the cases before we can insert (q, h').