Journal of Software for Algebra and Geometry

The power domination toolbox

Johnathan Koch and Beth Bjorkman

vol 16 2026

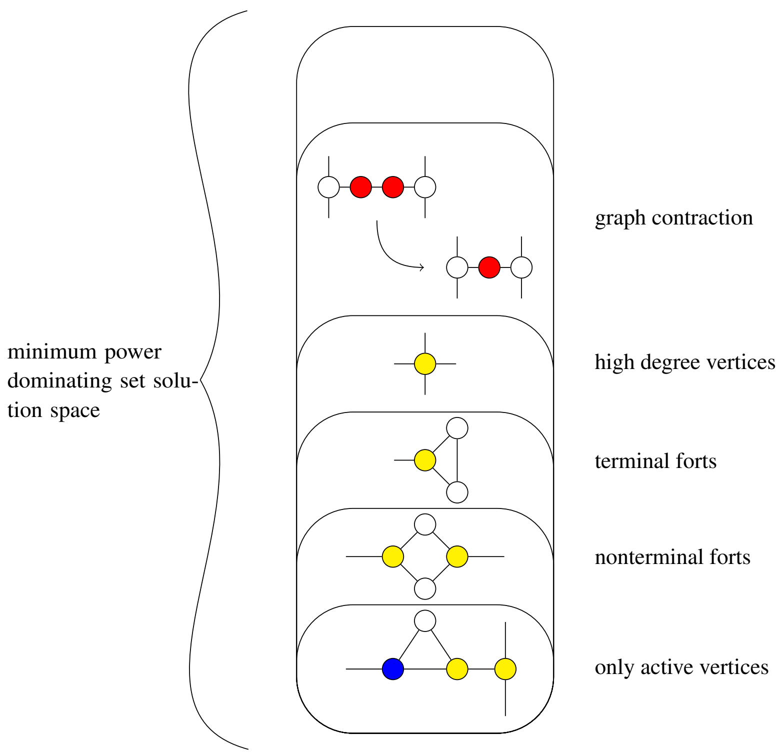
JSAG **16** (2026), 59–77
https://doi.org/10.2140/jsag.2026.16.59

The Journal of Software for
Algebra and Geometry

The power domination toolbox

JOHNATHAN KOCH AND BETH BJORKMAN

ABSTRACT: Phasor measurement units (PMUs) are placed at strategic vertices in an electrical power network to monitor the flow of power. Determining the minimum number and optimal placement of PMUs is modeled by the graph theoretic process called *power domination*. This paper describes the *power domination toolbox* (*PDT*), which efficiently identifies a minimum number of PMU locations that monitor the entire network. The PDT leverages graph theoretic literature to reduce the complexity of determining optimal PMU placements by: reducing the order of the graph (contraction), leveraging zero forcing forts, sorting the remaining solution space, and parallel computing. The PDT is a drop-in replacement of the current state-of-the-art exhaustive search algorithm in Python and maintains compatibility with SageMath. The PDT can identify minimum PMU placements for graphs with hundreds of vertices on personal computers and can analyze larger graphs on high performance computers. The PDT affords users the ability to investigate power domination on graphs previously considered infeasible due to the number of vertices resulting in a prohibitively long run-time.



MSC2020: 05C57, 05C69, 68R10.
Keywords: optimal sensor placement, phasor measurement units, graph methods, power domination.

1. INTRODUCTION. In 2003, a blackout in the Ohio power grid cascaded through a large portion of the Northeastern United States and Canada [13]. Blackouts this extensive can be mitigated by monitoring the power grid with phasor measurement units (PMUs) and acting quickly on the information they provide. PMUs use conservation of energy laws to observe phasor measures at distant locations in addition to the directly connected power lines. One goal for power grid planners is to maximize grid coverage while minimizing the number of installed PMUs to minimize the cost of grid maintenance. A minimum set of locations to install PMUs in order to monitor the entire network is called a minimum power dominating set (PDS).

Currently, some algorithms that are used to find optimal PMU placement locations include: genetic algorithm, particle swarm optimization, tabu search, greedy algorithm, integer linear programming, integer quadratic programming, simulated annealing, hybrid algorithm, exhaustive search, depth-first search, and minimum spanning tree [11]. Recently, Hicks and Smith [14] have implemented integer linear programming methods in Gurobi to achieve results on large graphs. To build the integer linear programming constraints, Hicks and Smith created a restricted problem where PMUs are placed at certain locations and prohibited from being placed at other locations. This solution, however, is no longer restricted to the integers and may be slightly greater from the minimum number of PMUs required. To find an exact solution, graph theorists commonly perform power domination via a combination of zero forcing code originally developed by Jason Grout and updated by Jephian Lin, and power domination code developed by Brian Wissman [1; 2; 3; 4; 10]. This method is an exhaustive search for a minimum PDS, and we will call this the *JL-BW algorithm*.

The PDT serves as a drop-in replacement and extension for the JL-BW algorithm written in Python. We implement preprocessing techniques used in graph theoretic proofs that have previously not been implemented in software, including: graph contraction to reduce propagation time, leveraging zero forcing forts to restrict the solution space, and assigning a qualitative score to sets in the remaining solution space to determine minimum PDSs more efficiently. Additionally, the PDT iterates over connected components of input graphs to more effectively determine PDSs of disconnected graphs. Parallel compute methodologies are also implemented to fully leverage computational resources. To demonstrate the PDT's utility: on random graphs with 120 vertices we see an average 19 times run-time improvement over the JL-BW algorithm. The amount of improvement is dependent on the graph structure, and is repeatable on random graphs of other sizes as well as standard test networks.

This paper will describe the PDT. Section 2 will provide the graph theoretic definitions and terminology required in this paper. Section 3 will define the power domination algorithm and related concepts. Section 4 will describe the process that the PDT uses to find a minimum PDS with brief discussions on the run-time for each step. Section 5 is a survey of run-time analysis, demonstrating the efficiency of the PDT over the JL-BW algorithm. Finally, Section 6 gives examples of interfacing with the PDT.

2. GRAPH THEORY. An electrical power grid can be represented as a *graph* G, which consists of two sets: a set of *vertices* (busses), $V(G)$, and a set of unordered pairs of vertices called *edges* (transmission

`powerdominationtoolbox` version 2

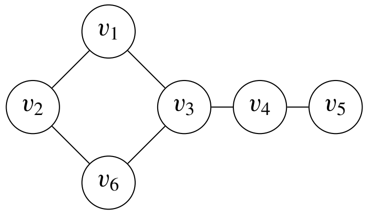


Figure 1. The tadpole graph

lines), $E(G)$, usually written as v_1v_2 for vertices v_1 and v_2. An edge is *incident* to the vertices it contains. Two vertices are *adjacent* (neighbors) if there exists an edge between them. The *closed neighborhood* of a vertex v is the set consisting of v together with all neighbors of v and is written $N[v]$. Similarly, the closed neighborhood of a set of vertices S is the set S together with all neighbors of members of S and is written $N[S]$. The degree of a vertex v is the number of vertices adjacent to v and is written $deg_G(v)$. When the graph is understood, the subscript is omitted. A vertex with degree one is called a *leaf* and a vertex with degree zero is called an *isolated vertex*.

For a graph G, a structure H within G is called a *subgraph*, written $H \subseteq G$, when H is a graph with

$$V(H) \subseteq V(G) \quad \text{and} \quad E(H) \subseteq E(G).$$

A set $A \subseteq V(G)$ generates a *vertex-induced subgraph*, $G[A]$, where

$$V(G[A]) = A \quad \text{and} \quad E(G[A]) = \{xy \in E(G) : x, y \in A\}.$$

A *path* in a graph G has vertices $\{v_1, v_2, \dots, v_n\} \subseteq V(G)$, so that $\{v_iv_{i+1} : 1 \leq i \leq n-1\} \subseteq E(G)$, and is usually written as $v_1v_2 \dots v_n$. The *length* of a path is the number of edges it contains. A graph G is *connected* if there exists at least one path between any two distinct vertices. For a vertex x in $V(G)$, if G is a connected graph but $G[V(G) \setminus \{x\}]$ is not connected, then x is a *cut vertex*.

For a graph G and edge $xy \in E(G)$, an *edge contraction* (contraction) on xy adds a new vertex z to G such that z is adjacent to any vertex adjacent to either x or y and removes x and y from G as well as any edges incident to x or y.

For a graph G with subset $X \subseteq V(G)$, the *entrance of* X in G, written $\partial(X)$, is the set of vertices not in X but adjacent to at least one vertex in X. A *fort*, F, in a graph G is a nonempty subset of vertices for which no vertex in $\partial(F)$ is adjacent to exactly one vertex in F [8]. A *terminal fort* with corresponding cut vertex v, denoted F_v, is a fort in which $\partial(F) = \{v\}$. A C_4 *fort*, denoted $F_{x,y} = \{x, y\}$ with $\partial(F_{x,y}) = \{a, b\}$, consists of nonadjacent vertices x, y satisfying $\deg_G(x) = \deg_G(y) = 2$ so that $G[\{a, b, x, y\}]$ is a C_4. Note that a terminal fort is denoted by its cut vertex while a C_4 fort is denoted by the vertices contained within the fort. By way of example, the sets $\{v_1, v_2, v_6\}$ and $\{v_1, v_6\}$ are both forts of the tadpole graph in Figure 1 with supports $\{v_3\}$ and $\{v_2, v_3\}$ respectively. The set $\{v_1, v_2, v_6\}$ is a terminal fort with corresponding cut vertex v_3, and would be denoted F_{v_3}. The set $\{v_1, v_6\}$ is a C_4 fort and would be denoted F_{v_1,v_6}.

3. POWER DOMINATION. In the power domination algorithm outlined by Haynes et al. [9], vertices are either *unobserved* or *observed* by a PMU. The *power domination algorithm* with input $S \subseteq V(G)$ is as follows:

(1) (*Domination step*) Each vertex in S, or adjacent to a vertex in S, is observed.

(2) (*Zero forcing step*) While there exists an observed vertex adjacent to exactly one unobserved vertex, the unobserved vertex becomes observed.

For a graph G and subset $S \subseteq V(G)$, we identify the set of observed vertices in the graph G after applying the power domination algorithm as $\mathrm{Obs}(G; S)$. A *power dominating set* (*PDS*) is any subset $S \subseteq V(G)$ where $\mathrm{Obs}(G; S) = V(G)$. The *power domination number* of a graph G is the cardinality of a minimum PDS, written as $\gamma_P(G)$. Note, there may be many minimum PDSs for a particular graph.

We also leverage the following terminology from [5]: for a graph G and subset $X \subseteq V(G)$, a *power dominating set subject to* X is any subset $S \subseteq V(G)$ containing X that is also a power dominating set. In a connected graph with maximum degree at least 3, a minimum power dominating set can be chosen in which each vertex has degree at least 3 [9, Observation 4]. In a restricted power domination problem, certain vertices are already observed by the existing PMUs and may have no unobserved neighbors. This means that some vertices provide no additional observations when a PMU is placed on them. Combining this with a restriction to vertices with degree at least 3, we define *active vertices* to be vertices that have degree at least 3 and have unobserved neighbors with respect to the restricted power domination problem subject to X.

4. OPTIMIZATIONS. The PDT implements optimizations to the power domination process in four ways:

(1) Contract the input graph.

(2) Leverage zero forcing forts.

(3) Sort the solution space.

Steps 1–3 are done as preprocessing steps and are followed by:

(4) Distribute the search for a minimum PDS across parallel compute resources.

The following sections describe each of these optimizations in turn. Additionally, we will provide short discussions in each subsection on the run-time impact of implementing the given optimization. The discussion centers around a collection of graphs we provide alongside the PDT, and describe in more detail in Section 5.3. For the purposes of these discussions, this data set is a collection of 600 random graphs with 100 graphs each on 20, 40, 60, 80, 100, and 120 vertices.

4.1. ***Contracting the graph.*** Due to the nature of the zero forcing step, we have an opportunity to reduce propagation time via contraction. Paths on vertices with degree less than 3 are contracted via Algorithm 1. We will then show that this contraction yields a graph with minimum PDSs that are also minimum PDSs of the input graph.

Algorithm 1. Graph Contraction Algorithm

Input: A graph G

Output: A contracted graph G'

1 **for** *H a connected component of $G[\{v \in V(G) : \deg_G(v) < 3\}]$* **do**
2   **if** *H is a path terminating in two leaves in G, or H is a cycle* **then**
3   Contract the path or cycle in G corresponding to H to an isolated vertex in G.
4   **end**
5   **if** *H is a path terminating in vertices adjacent in G to distinct vertices x, y with $\deg_G(x) \geq 3$ and $\deg_G(y) \geq 3$* **then**
6   Contract the path in G corresponding to H to a single degree 2 vertex.
7   **end**
8   **if** *H is a path terminating in a leaf and a vertex adjacent in G to a vertex x with $\deg_G(x) \geq 3$* **then**
9   Contract the path in G corresponding to H to a leaf.
10   **end**
11   **if** *H is a path terminating in vertices adjacent in G to a single vertex x with $\deg_G(x) \geq 3$* **then**
12   Contract the path in G corresponding to H to a pair of adjacent degree 2 vertices.
13   **end**
14 **end**

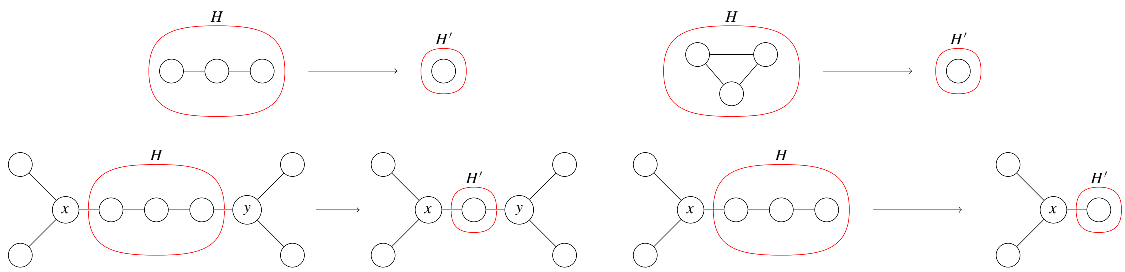


Figure 2. Cases where Algorithm 1 contracts H to a single vertex. Contraction when H is a path (top left). Contraction when H is a cycle (top right). Contraction when H terminates in vertices adjacent to distinct vertices in G, x and y, both with degree at least 3 (bottom left). Contraction when H terminates in a leaf and a vertex adjacent to a vertex, x, in G with degree at least 3 (bottom right).

Algorithm 1 contracts subgraphs of the input graph to either a single vertex, or a pair of adjacent vertices. Figure 2 provides examples of when Algorithm 1 contracts H to a single vertex which corresponds to the conditional statements on lines 2, 4, and 6. Figure 3 provides the example of when Algorithm 1 contracts H to a pair of adjacent vertices which corresponds to the conditional statement on line 8.

A minimum power dominating set for the contracted graph resulting from Algorithm 1 corresponds to a minimum power dominating set of the input graph.

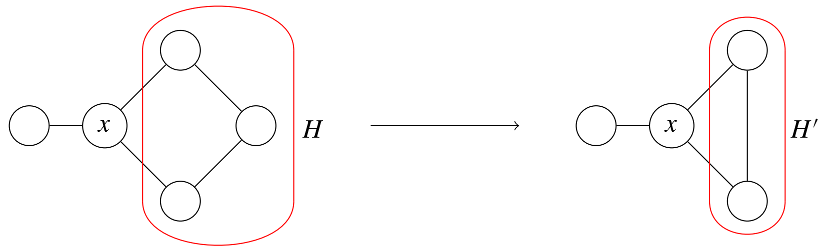


Figure 3. Case where Algorithm 1 contracts H to a pair of adjacent vertices: when H terminates in vertices adjacent to a unique vertex in G with degree at least 3.

$\|V(G)\|$	20	40	60	80	100	120
time to determine G'	2.320×10^{-4}	4.357×10^{-4}	6.440×10^{-4}	6.907×10^{-4}	1.251×10^{-3}	1.661×10^{-3}

Table 1. Average time (in seconds) the PDT uses to calculate G' for graphs of varying sizes.

Theorem 4.1. *Let G be a graph and let G' be the result of contracting G via Algorithm 1. A minimum power dominating set of G' corresponds to a minimum power dominating set of G. Therefore, $\gamma_P(G) = \gamma_P(G')$.*

Proof. We will begin with a minimum power dominating set S of G and demonstrate a corresponding minimum power dominating set of G'. For any vertex $v \in S$, if $v \notin V(G')$, then v is a degree 1 or 2 vertex in G that was contracted to create G'. Note that as S is minimum, there cannot be 2 adjacent such vertices in S. Replace each vertex v with its corresponding contracted vertex and call this set S'. The power domination process on G' with initial set S' proceeds analogously to the power domination process on G with initial set S, with removed observations along the contracted paths.

Next, consider a power dominating set S' of G'. For any vertex $x \in S'$, if $x \notin V(G)$, then x is the result of contracting degree 1 or 2 vertices in G. Replace each such x with one of the corresponding vertices and call the resulting set S. The power domination process on G with initial set S proceeds analogously to the power domination process on G' with initial set S', with added observations along the noncontracted paths.

We have shown corresponding minimum power dominating sets of the same size for G and G' and so $\gamma_P(G) = \gamma_P(G')$. □

Contracting the graph is done in linear time, while the propagation steps it eliminates from the power domination process is potentially exponential. This is a novel method of the PDT that dramatically improves run-time on graphs that demonstrate substructures of long chains compared to the JL-BW algorithm. Table 1 outlines the average run-time to contract random graphs of varying size.

4.2. ***Leveraging forts.*** Forts were first utilized in zero forcing by Fast and Hicks in [8]. They were generalized to power domination Bozeman et al. in [5]. Hicks and Smith exploited forts to find minimum power dominating sets in their integer linear program method [14]. We will also use forts in order to find a minimum power dominating set while considering significantly fewer cases than other brute force methods.

Proposition 4.2 [5, Proposition 4.3]. *Let G be a graph and F be any fort of G. If S is a power dominating set of G, then $S \cap N[F] \neq \varnothing$.*

That is, any power dominating set must intersect nontrivially with the closed neighborhood of every fort, or equivalently, that any power dominating set must intersect nontrivially with $F \cup \partial(F)$. In special cases, we can utilize this result to determine which vertices from the closed neighborhood of a fort are in *some* minimum power dominating set.

We begin by leveraging certain terminal forts.

Theorem 4.3. *Let G be a connected graph with distinct terminal forts F_x and F_y with $F_x \subseteq \mathrm{Obs}(G; \{x\})$ and $F_y \subseteq \mathrm{Obs}(G; \{y\})$. Then, $y \in F_x$ or $x \in F_y$ if and only if $\gamma_P(G) = 1$.*

Proof. Let G be a connected graph with distinct terminal forts F_x and F_y where $x, y \in V(G)$ are cut vertices, $\partial(F_x) = \{x\}$, $\partial(F_y) = \{y\}$, $F_y \subseteq \mathrm{Obs}(G; \{y\})$, $F_y \subseteq \mathrm{Obs}(G; \{y\})$.

($\Rightarrow$) Assume that $\gamma_P(G) = 1$ and let $S = \{v\}$ be a minimum power dominating set.

If $v = x$, assume for eventual contradiction that $y \notin F_x$. As F_x is a terminal fort, x is the only vertex of $V(G) \setminus F_x$ adjacent to any vertex in F_x and so $F_y \cap F_x = \varnothing$. By Proposition 4.2, it must be the case that $\{x\} \cap N[F_y] = \{y\}$, which contradicts that F_x and F_y are distinct terminal forts.

Now suppose $v \neq x$. By Proposition 4.2, $v \in N[F_x]$. Hence, $v \in F_x$. During the power domination process starting at $\{v\}$, any vertex in $V(G) \setminus N[F_x]$ must be observed via a zero forcing step from x. Thus, $G[V(G) \setminus N[F_x]]$ is empty or a path. Hence, $\{x\}$ must be a power dominating set. If $y \notin F_x$ and $V(G) \setminus N[F_x]$ is empty or a path, we obtain that $x \in F_y$, as otherwise F_y is not a fort. Therefore, $x \in F_y$ or $y \in F_x$.

($\Leftarrow$) Now assume $y \in F_x$ or $x \in F_y$. Without loss of generality, say $y \in F_x$. Now y is a cut vertex, so let H_x be the connected component of $G - y$ containing x and there exists a connected component of $G - y$ not containing x where a is a neighbor of y. Call this connected component H_a. Since $a \neq x$ and $\{x\} = \partial(F_x)$, it must follow that $a \in F_x$. Similarly, $H_a \subseteq F_x$. Moreover, as $F_x \subseteq \mathrm{Obs}(G; \{x\})$ and $x \notin H_a$, it must be the case that a becomes observed as the result of a zero forcing step from y. This zero forcing step can only occur if a is the only neighbor of y not in H_x. Hence the only components of $G - y$ are H_x and H_a. Following the zero forcing process and that $H_a \subseteq F_x \subseteq \mathrm{Obs}(G; \{x\})$, we obtain that H_a must be a path. Since H_a is a path, which is not a fort, and $\partial(F_y) = \{y\}$, it must be that $F_y = H_x$. Therefore, $\{y\}$ is a power dominating set of G and $\gamma_P(G) = 1$. □

The next result follows from the contrapositive of Theorem 4.3: if there are multiple distinct terminal forts, the power domination number is greater than 1 if and only if the terminal forts do not intersect. By Proposition 4.2, any power dominating set must intersect nontrivially with the closed neighborhood of each fort, and since these entrance vertices observe the entire fort, there is some minimum power dominating set containing all of these terminal fort entrances.

Corollary 4.4. *For a connected graph G with $\gamma_P(G) > 1$, there exists a minimum power dominating set S that contains the entrance of every terminal fort F_x satisfying $F_x \subseteq \mathrm{Obs}(G; \{x\})$.*

We now leverage C_4 forts in the contracted graph G' resulting from applying Algorithm 1. This allows us to find more than just C_4 forts in the original graph G, as the nonadjacent degree 2 vertices could have resulted from contracting many degree 2 vertices in G. In this contracted graph, the only remaining C_4 forts have support vertices with degree at least 3.

Observation 4.5 [9, Observation 4]. If G is a connected graph with maximum degree at least 3, then G contains a minimum power dominating set in which every vertex has degree at least 3.

Utilizing Observation 4.5 and Proposition 4.2, the following holds.

Corollary 4.6. *Given a graph G, there exists a minimum power dominating set of G, say S, such that for every C_4 fort $F_{x,y}$ in the contracted graph G' associated with Algorithm 1, $S \cap F_{x,y} = \varnothing$ and $S \cap \partial(F_{x,y}) \neq \varnothing$.*

Corollaries 4.4 and 4.6 can be used to determine a set of vertices that are in *some* minimum power dominating set of a given graph.

Observation 4.7. For any connected graph G with $\gamma_P(G) \neq 1$ with corresponding contracted graph G' from Algorithm 1, there exists a minimum power dominating set S of G' such that

(1) for every terminal fort F_v in G' satisfying $F_v \subseteq \mathrm{Obs}(G'; \{v\})$, $S \cap F_v = \varnothing$ and $v \in S$, and

(2) for every C_4 fort $F_{x,y}$ in G', $S \cap F_{x,y} = \varnothing$ and $S \cap \partial(F_{x,y}) \neq \varnothing$.

That is, for any connected input graph G and corresponding contracted graph G' from Algorithm 1 with $\gamma_P(G)] \neq 1$, there exists a minimum power dominating set S of G containing the entrance of every terminal fort in G' and at least one of the 2 entrance vertices of any C_4 fort in G'.

We will call the entrance vertices of terminal forts in a contracted graph G' *preferred vertices*, and let $\mathrm{Pref}(G) = \{v : F_v$ is a terminal fort of the contracted graph $G'\}$. We will call the entrance vertices of C_4 forts in G' *paired entrance vertices* and $\mathrm{Pev}(G) = \{\partial(F_{x,y}) : F_{x,y}$ is a C_4 fort of the contracted graph $G'\}$. Preferred vertices give a lower bound on the power domination number.

Observation 4.8. For any connected graph G with $\gamma_P(G) > 1$,

$$\gamma_P(G) \geq |\mathrm{Pref}(G)|.$$

In practice, the PDT first contracts the graph as in Algorithm 1, then determines preferred vertices for each connected component, as shown in Algorithm 17. The conditional on line 6 corresponds to catching the edge case where $\gamma_P(G) = 1$, which is nested in the loop on line 5. The loop on line 5 locates terminal forts, including type I and type II forts as defined by Hicks and Smith [14] and more general terminal forts as shown in Figure 4. Paired entrance vertices are located with Algorithm 9, which leverages fast methods for determining chordless cycles within a graph. The run-time to calculate $\mathrm{Pref}(G)$ and $\mathrm{Pev}(G)$ is given in Table 2.

Algorithm 2. Algorithm to determine Pref(G)

```
Input: A graph G
Output: Pref(G)
Pref(G) ← ∅
for v ∈ V(G) where v is a cut vertex do
    if v is adjacent to at least 2 leaves then
        Pref(G) ← Pref(()G) ∪ {v}
    end
    for H a connected component of G[V(G) \ {v}], and |V(H)| > 1 do
        if H ⊆ Obs(G; {v}) then
            if Pref(G) = ∅ then
                if V(G) ⊆ Obs(G; {v}) then
                    return {v}
                end
            end
            Pref(G) ← Pref(()G) ∪ {v}
        end
    end
end
return Pref(G)
```

Algorithm 3. Algorithm to determine the entrance of forts associated with induced C_4 subgraphs of the contracted graph

```
Input: A graph G
Output: Pev(G)
G′ ← Algorithm 1(G)
A ← {v ∈ V(G′) : deg_G v = 2}
B ← N_G′[A]
Pev(G) ← ∅
for H a connected component of G′[B] do
    if C_4 ⊆ H then
        Pev(G) ← Pev(G) + {v ∈ V(H) : deg_G(v) > 2}
    end
end
```

4.3. ***Determining qualitative scores.*** The PDT sorts the potential power dominating sets in the solution space to more optimally locate power dominating sets. This is done by maximizing the number of vertices observed after considering the restricted power domination problem on G' subject to Pref(G'). Define

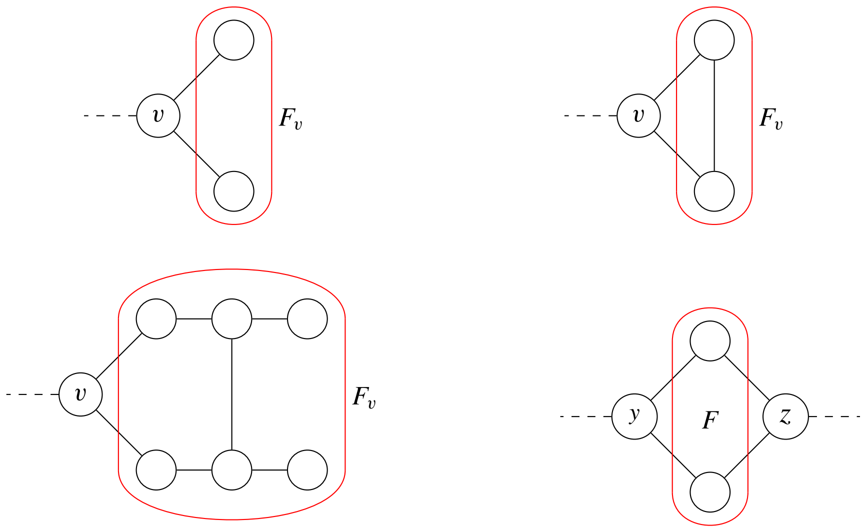


Figure 4. Forts located by the PDT indicated in red. Dashed edges represent connection(s) from the entrance vertices to the remainder of the graph. A pair of leaves, F_v, indicated by Hicks and Smith as Type I forts [14] (top left). A terminal C_3, F_v, indicated by Hicks and Smith as Type II forts [14] (top right). An example of a terminal fort, F_v, not described by Hicks and Smith, that is located by the PDT (bottom left). A fort associated with an induced C_4, F, indicated by Hicks and Smith as Type III forts [14] (bottom right).

$\lvert V(G)\rvert$	20	40	60	80	100	120
time to determine $\mathrm{Pref}(G)$	4.898×10^{-4} s	1.234×10^{-3} s	2.046×10^{-3} s	2.501×10^{-3} s	2.703×10^{-3} s	2.831×10^{-3} s
time to determine $\mathrm{Pev}(G)$	4.225×10^{-4} s	9.010×10^{-4} s	1.385×10^{-3} s	1.487×10^{-3} s	1.656×10^{-3} s	1.905×10^{-3} s

Table 2. Average run-time (in seconds) the PDT uses to calculate $\mathrm{Pref}(G)$ and $\mathrm{Pev}(G)$.

the *qualitative score*, of a vertex v as

$$Q(v) := \mathrm{Obs}(G; \mathrm{Pref}(G) \cup \{v\}), \tag{1}$$

Formulating $Q(v)$ in this way affords another opportunity for optimization. If $\gamma_P(G) \neq |\mathrm{Pref}(G)|$ and $\max(Q(v) : v \in V(G)) = |V(G)|$, then $\gamma_P(G) = |\mathrm{Pref}(G)| + 1$ and $\mathrm{Pref}(G) \cup \{v\}$ is a minimum PDS for any $v \in V(G)$ such that $Q(v) = |V(G)|$.

The PDT iteratively adds vertices to a potential PDS from largest to smallest qualitative score. Figure 5 provides $Q(v)$ for each active vertex in the Zim graph. The PDT checks subsets with higher total qualitative score first. The JL-BW algorithm does no such preprocessing and checks subsets lexicographically by vertex label.

4.4. ***Distribution across multiple threads.*** Using an exhaustive search algorithm to determine the power domination number of a graph, necessarily, requires determining $\mathrm{Obs}(G; S)$ for sets $S \subseteq V(G)$ with $|S| < \gamma_P(G)$. While the PDT dramatically lowers this number of sets S, the number of sets to analyze

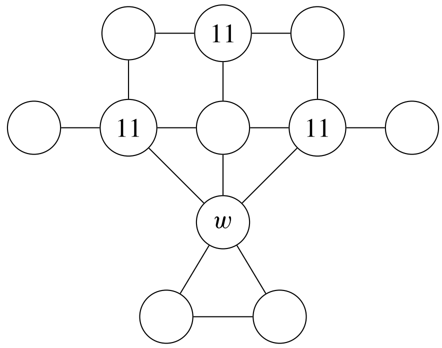


Figure 5. Zim graph with qualitative score for each nonpreferred vertex with degree at least 3 displayed. Vertex w is the only preferred vertex.

with the PDT still grows factorially as the power domination number grows. Checking each set S can be viewed as an independent process and so the PDT utilizes parallelization.

The multiprocessing library in Python is implemented to distribute the search for a minimum PDS across available CPU resources. This parallelization requires computational overhead, so the PDT checks the first 50,000 subsets that may be PDSs on a single compute thread. Once the number of subsets to check grows, however, it becomes advantageous to parallelize the search for a minimum PDS with a given number of PMUs. The PDT can dynamically determine the number of processes to leverage, or it can be set explicitly to use a predetermined number of processes. Common personal computers can facilitate the search for a minimum PDS across approximately 10 processes while allowing the user to continue using the computer for light tasks.

4.5. ***The PDT algorithm.*** We now discuss the algorithm that the PDT uses to find a minimum PDS of an input graph G. The PDT calculates G' and the parameters $\text{Pref}(G')$, $\text{Pev}(G')$, $\text{Obs}(G'; \text{Pref}(G'))$, and active vertices. The PDT then iterates over the connected components of G', restricts the predetermined parameters to the connected component, and calls Algorithm 30 with these parameters as additional input to determine a minimum PDS of the component. The PDT allows the user to directly call this algorithm as `PDT_minpds_connected`, and determines G', $\text{Pref}(G')$, $\text{Pev}(G')$, $\text{Obs}(G'; \text{Pref}(G'))$, and active vertices if not provided by the user. Let us inspect Algorithm 30 as if the user called it directly.

Line 1 is the application of the graph contraction algorithm. If there are no vertices with degree more than 2, then any vertex is a PDS, and hence an arbitrary vertex is returned on line 4.

The PDT then calculates $\text{Pref}(G')$, and $\text{Obs}(G; \text{Pref}(G'))$ and stores these observed vertices in the set B on line 5. If $B = V(G)$, then $\text{Pref}(G')$ is a PDS and is then returned on line 8. Lines 9 and 10 then locate active vertices for the restricted power domination problem on G' subject to $\text{Pref}(G')$ and determine $Q(v)$ for each active vertex. If $\max(Q(v)) = |V(G')|$, then the PDT has located a minimum power dominating set with a PMU on each vertex in $\text{Pref}(G')$ and a vertex with $Q(v) = |V(G')|$. The PDT returns this minimum PDS on line 13.

If a PDS has not been located at this point, $\gamma_P(G') \geq |\operatorname{Pref}(G')|+2$. The PDT then determines $\operatorname{Pev}(G')$ and φ. If $\varphi - |\operatorname{Pref}(G)| > 2$, more than 2 additional vertices are required to form a PDS. These two cases are covered by $i = \max\{2, \varphi - |\operatorname{Pref}(G)|\}$ on line 14.

The PDT then begins checking sets of size $|\operatorname{Pref}(G')| + i$ for a minimum PDS. Line 17 sets up the combination of additional vertices, C, to be added to $\operatorname{Pref}(G')$ to create a subset S, which is formed on line 18. The PDT enforces that S intersects nontrivially with each pair of entrance vertices on line 19. Determining the observed vertices in $\operatorname{Obs}(G; \operatorname{Pref}(G') \cup C)$ on line 20 is given by repeated application of the zero forcing step to the set $\operatorname{Obs}(G; \operatorname{Pref}(G)) \cup N[C]$. If the resulting set is equal to $V(G')$, then the minimum PDS S is returned on line 22. If all B with $|B| = i$ are exhausted, then i is incremented on line 23 and the PDT returns to the loop starting on line 16.

5. RUN-TIME ANALYSIS. We now compare the JL-BW algorithm and the PDT in two ways: empirically, and with run-time examples on both random graphs with varying size and common IEEE test systems.

5.1. ***Empirical comparison.*** For a graph G, we can compare the number of subsets strictly smaller than $\gamma_P(G)$ checked by both the JL-BW algorithm and the PDT. Let N and N' represent this number for the JL-BW algorithm and the PDT respectively:

$$N := \sum_{i=1}^{\gamma_P(G)-1} \binom{|V(G)|}{i}.$$

$$N' := 1 + a + \sum_{i=\max(2,\varphi)}^{\gamma_P(G)-p-1} \binom{a-\varphi}{i} 2^{\varphi-p}.$$

Where G' is the contracted graph, $p = |\operatorname{Pref}(G')|$, φ is as in Section 4.2, and a is the number of active vertices with respect to the restricted power domination problem on G' subject to $\operatorname{Pref}(G')$. There is 1 case from checking if $\operatorname{Pref}(G')$ is a PDS and a cases from determining if any vertex satisfies $Q(v) = |V(G')|$. We then check sets containing i additional vertices from the set of active vertices that intersect nontrivially with each of the paired entrance vertices. Observe that N' is often lower than N due to the prevalence of preferred vertices, paired entrance vertices, and nonactive vertices.

5.2. ***IEEE test systems.*** Returning to the original problem of the 2003 power grid failure, we consider the IEEE 39 bus test system that represents a historic model of the New England power grid as available in the pandapower Python module [15]. This graph is displayed in Figure 6 and has power domination number 5. The JL-BW algorithm evaluates the $N = 92,170$ subsets to determine $\gamma_P(G) > 4$ and locates a minimum PDS of size 5 in an average of 3.566 seconds. The PDT contracts to a graph on 36 vertices and finds 3 preferred vertices. The contracted graph is shown in Figure 6. By considering the restricted power domination problem on G' subject to $\operatorname{Pref}(G')$, the PDT indicates 11 active vertices. The PDT evaluates $N' = 12$ subsets to determine $\gamma_P(G) > 4$ and locates a minimum PDS of size 5 in an average of 2.673×10^{-3} seconds.

Algorithm 4. The PDT algorithm for determining a minimum PDS of a *connected* graph

```
Input: A connected graph G
Output: A minimum PDS of G
G′ ← Algorithm 1(G)
if ∄v ∈ V(G) with deg_G(v) > 2 then
    S ← {v} for any v ∈ V(H)
    return S
end
B ← Obs(G′; Pref(G′))
if B = V(G′) then
    S ← Pref(G′)
    return S
end
U ← V(G′) \ B
A ← {v ∈ V(G′) : deg_G′(v) > 2 and N[v] ∩ U ≠ ∅}
if max(Q(v) = |V(G′)|) then
    S ← Pref(G′) ∪ {v} for some v that maximizes Q(v)
    return S
end
i ← max(2, φ − |Pref(G′)|)
PDS ← false
while not PDS do
    for C ⊆ A where |C| = i do
        S ← Pref(G′) ∪ C
        if S ∩ R ≠ ∅ for each R ∈ Pev(G′) then
            if Obs(G′; S) = V(G′) then
                PDS ← true
                return S
            end
        end
    end
    i ← i + 1
end
```

The IEEE 118 bus test system [6] shown in Figure 7 is also available in the pandapower module and has $\gamma_P(G) = 8$. The JL-BW algorithm would need to evaluate the $N \approx 5.620 \times 10^{10}$ subsets to determine $\gamma_P(G)$, however the search was terminated after a week and no minimum PDS was located. The PDT locates the single preferred vertex, contracts the graph to a graph with 115 vertices, locates

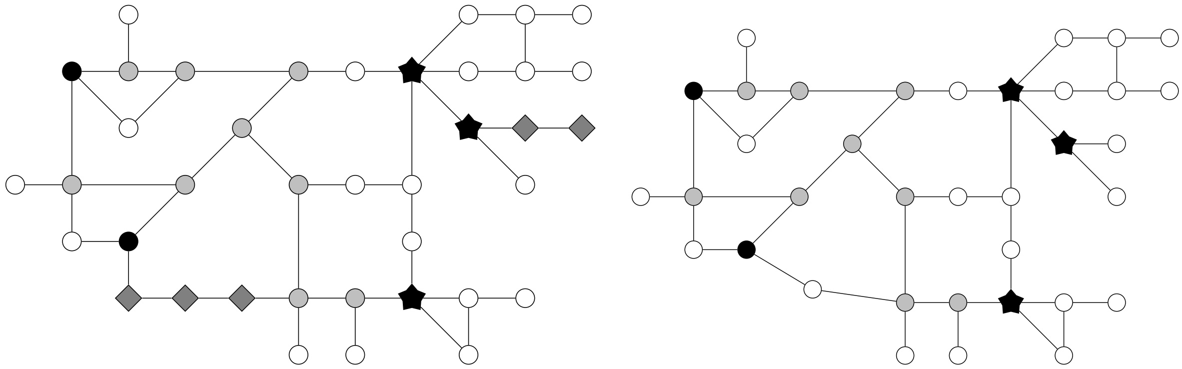


Figure 6. New England power grid as a graph with 39 vertices (left). The PDT removes the dark gray diamond vertices to contract the graph into the graph (right). Further, the PDT locates 3 preferred vertices (black stars) and 11 active vertices 3 (light gray and black circles). The minimum PDS of size 5 returned by the PDT is indicated by solid black vertices.

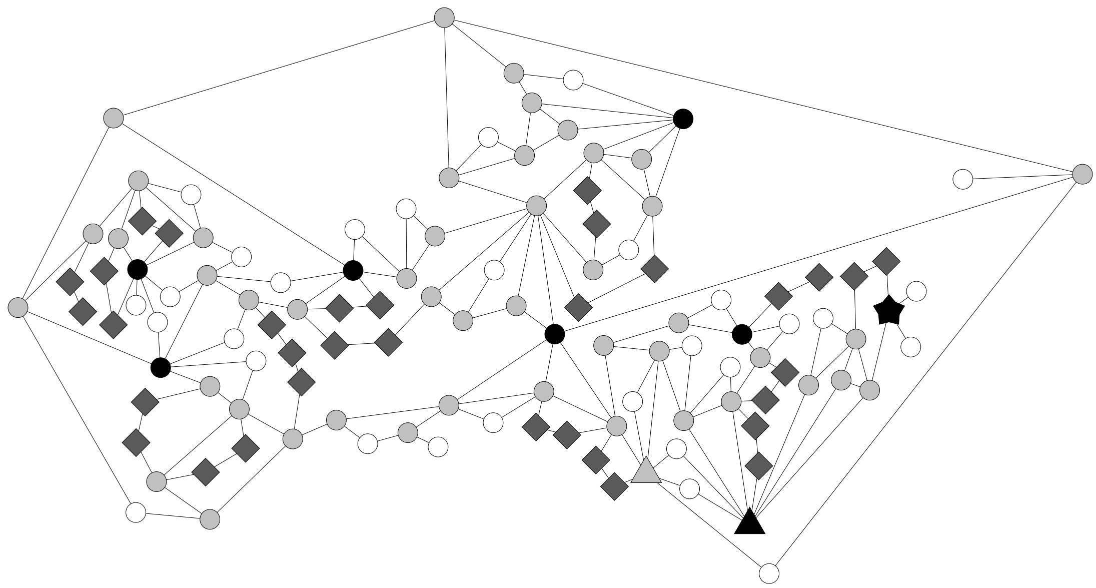


Figure 7. IEEE 118 vertex test system as a graph. The PDT contracts the dark gray diamond vertices to yield G' with 101 vertices. The PDT indicates 1 preferred vertex (black star), 1 set of paired entrance vertices (triangles), and 54 active vertices (light gray and black circles and triangles). The minimum PDS of size 8 returned by the PDT is indicated by solid black vertices.

54 active vertices, and finds one set of paired entrance vertices. By leveraging the PDT and 32 threads, $N' \approx 4.650 \times 10^7$ subsets are evaluated to determine $\gamma_P(G) > 7$ and to locate a minimum PDS of size 8 in 1.098×10^2 seconds. This exemplifies the drastic run-time improvements of the PDT over the JL-BW algorithm.

Table 3 shows the average run-time on various other IEEE test systems up to 300 vertices. Table 5 $|V(G)|$, $|V(G')|$, $|\operatorname{Pref}(G')|$, and $|\operatorname{Pev}(G')|$ for other IEEE test systems up to the 1354 vertex system.

number of vertices	$\gamma_P(G)$	average JL-BW algorithm time	average PDT time
5	1	$\mathbf{2.087 \times 10^{-5}}$ s	1.526×10^{-4} s
6	1	$\mathbf{2.791 \times 10^{-5}}$ s	1.643×10^{-4} s
9	1	$\mathbf{9.541 \times 10^{-5}}$ s	7.172×10^{-4} s
11	2	$\mathbf{1.518 \times 10^{-4}}$ s	3.194×10^{-4} s
14	2	$\mathbf{1.806 \times 10^{-4}}$ s	1.023×10^{-3} s
24	3	1.161×10^{-2} s	$\mathbf{2.043 \times 10^{-3}}$ s
30	3	8.830×10^{-3} s	$\mathbf{1.920 \times 10^{-3}}$ s
30	3	9.259×10^{-3} s	$\mathbf{1.954 \times 10^{-3}}$ s
33	1	5.945×10^{-4} s	$\mathbf{5.368 \times 10^{-4}}$ s
39	5	3.567 s	$\mathbf{2.733 \times 10^{-3}}$ s
57	3	2.352×10^{-1} s	$\mathbf{1.064 \times 10^{-2}}$ s
89	5	2.023×10^{2} s	$\mathbf{7.776 \times 10^{-3}}$ s
118	8	> 1 week	$\mathbf{1.098 \times 10^{2}}$ s
145	13	N/A	$\mathbf{6.495 \times 10^{5}}$ s*
200	20	N/A	$\mathbf{4.062 \times 10^{-2}}$ s
300	?	N/A	> 1 week

Table 3. Average run-time for determining $\gamma_P(G)$ for various IEEE test systems. The 145 vertex test system, due to it's long run-time, was only tested once where each other graph was tested 20 times unless terminated early. Note that the 200 node system is much faster than the 145 node system due to the PDT locating 17 preferred vertices.

5.3. ***On Erdős–Rényi random graphs with varying size.*** We now compare the run-time of the PDT to the JL-BW algorithm on Erdős–Rényi random graphs. An *Erdős–Rényi random graph* on n vertices is a graph resulting from adding edges between each pair of distinct vertices with a predetermined probability [7]. In this paper, we will consider Erdős–Rényi random graphs with edge probability of 0.05 and a varying number of vertices. While it is not guaranteed that an Erdős–Rényi random graph is connected, we generate Erdős–Rényi random graphs until the resulting graph is connected and store the connected graph. To collect the following run-time data, we determined $\gamma_P(G)$ for each graph 20 times and used the average run-time as the estimated run-time for each graph.

We investigated the impact graph order has on time to find a minimum PDS by testing on connected Erdős–Rényi random graphs with 20, 40, 60, 80, 100, and 120 vertices. This data set is available in graph6 format [12] alongside the PDT and includes a total of 600 Erdős–Rényi random graphs (100 of each order). As expected, when the order of the graph increases, so does the time required to find a minimum PDS. This difference in run-time is shown in Figure 8 and Table 4 gives mean and median values.

In addition to the faster run-times, the PDT yields less variance in run-time.

6. Concluding remarks.

6.1. ***Using the power domination toolbox.*** The primary interface functions of the PDT are the following:

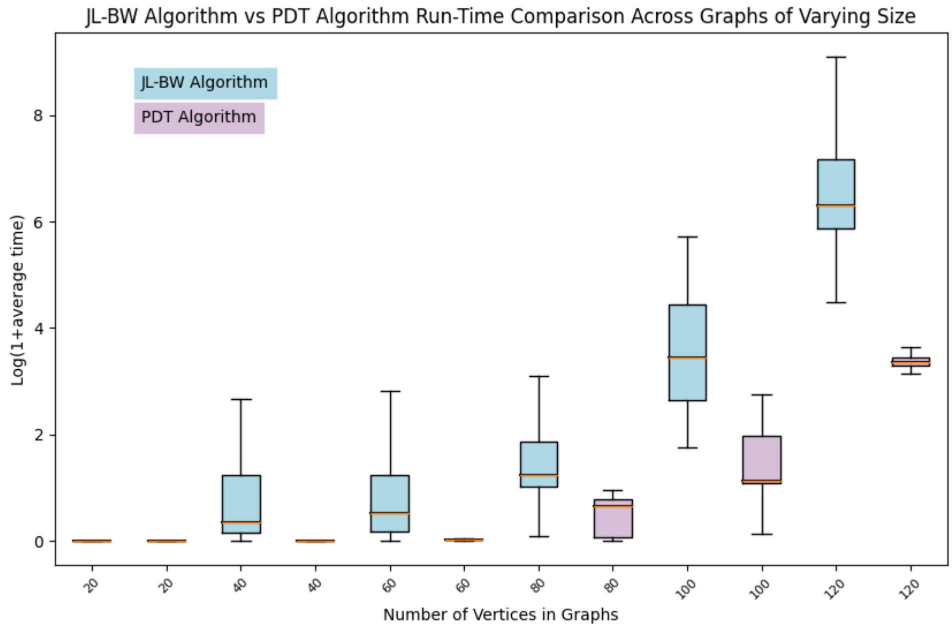


Figure 8. Boxplots of time to determine $\gamma_P(G)$ grouped by algorithm and $|V(G)|$ for each graph G in the Erdős–Rényi data set including 100 graphs with 20, 40, 60, 80, 100, and 120 vertices each.

	algorithm	$\lvert V(G)\rvert = 20$	$\lvert V(G)\rvert = 40$	$\lvert V(G)\rvert = 60$	$\lvert V(G)\rvert = 80$	$\lvert V(G)\rvert = 100$	$\lvert V(G)\rvert = 120$
mean	JL-BW	4.550×10^{-3}	4.646	6.361	1.009×10^{1}	7.216×10^{1}	1.108×10^{2}
	PDT	$\mathbf{6.692 \times 10^{-4}}$	$\mathbf{4.847 \times 10^{-3}}$	$\mathbf{7.233 \times 10^{-2}}$	$\mathbf{6.999 \times 10^{-1}}$	**4.077**	$\mathbf{3.935 \times 10^{1}}$
median	JL-BW	2.661×10^{-3}	4.218×10^{-1}	7.004×10^{-1}	2.416	3.009×10^{1}	5.483×10^{2}
	PDT	$\mathbf{5.833 \times 10^{-4}}$	$\mathbf{2.959 \times 10^{-3}}$	$\mathbf{2.034 \times 10^{-2}}$	$\mathbf{9.241 \times 10^{-1}}$	**2.110**	$\mathbf{2.793 \times 10^{1}}$

Table 4. Average run-time on Erdős–Rényi random graphs with varying number of vertices.

(1) `PDT_pdn(Input_graph, Number_workers) -> int`

This function returns an integer (the power domination number of the input graph) when supplied with a NetworkX graph object. Optionally, the user may supply this function with a number of compute threads to use in the parallelization step. If no number of compute threads are given, then all but one available compute threads are used by default.

(2) `PDT_minpds(Input_graph, Number_workers) -> list`

This function returns a list containing vertex labels of vertices that form a minimum power dominating set for the input graph. Optionally, the user may supply this function with a number of compute threads to use in the parallelization step. If no number of compute threads are given, then all but one available compute threads are used by default. The function signature is as follows:

(3) `CheckForPDSOfSize(Input_graph, Contracted, PreferredVertices, CycleEntrances, ActiveVertices, Blues, Placement_size, Number_workers) -> list`

This function returns a power dominating set of the given size (if one exists) that is subject to the restrictions discussed in Section 4. Power dominating sets returned by this function have no PMUs located on vertices with degree less than 3, no PMUs located on redundant vertices, and PMUs located on all preferred vertices. Optionally, the user may supply: a boolean value for if the input graph is already contracted, the list of preferred vertices, the list of paired entrance vertices, the list of active vertices, and the list of vertices colored blue in the restricted power domination problem subject to $\mathrm{Pref}(G)$. If none of these parameters are supplied, then the PDT will calculate them.

(4) `allpdsofsize(Input_graph, Size) -> list`

This generator yields each power dominating set of a given size. Note that this function is not parallelized and can be used to find all minimum PDSs and not just ones that satisfy the restricted power domination problem on G' subject to $\mathrm{Pref}(G')$.

(5) `parallel_allpds_of_size(Input_graph, Placement_size, Number_workers) -> list of lists`

This function returns a list of all power dominating sets of a given size. This differs from `allpdsofsize` in that each power dominating set is held in memory at a given time. Due to memory constraints, this function may overload personal computer's memory capacity for large power dominating sets. Note that this function leverages parallel computing methodologies.

Documentation for the other functions contained within the PDT can be located on GitHub as well as examples that act as unit tests for each function.

Moreover, we include functions that afford some preliminary investigation into power domination variations within the PDT, including: failed power domination, k-fault-tolerant power domination, k-PMU-defect-robust power domination, and fragile power domination. The PDT can also be used to investigate the token jumping reconfiguration graph for power domination, or the token addition and removal reconfiguration graph for power domination.

ACKNOWLEDGEMENTS. The authors thank Dr. Mary Flagg of the University of St. Thomas in Houston for her insights on forts and power domination. This project was sponsored, in part, by the Air Force Research Laboratory via the Autonomy Technology Research Center and Wright State University. This research was also supported by Air Force Office of Scientific Research labtask award 23RYCOR004.

Data availability statement. All tests were conducted on a Ryzen 9 5950X with 128 gigabytes of system RAM running Ubuntu 22.04.3. Total system RAM usage never exceeded 8 gigabytes at any time during the tests. The power domination toolbox is available on GitHub.

SUPPLEMENT. The online supplement contains version 2 of the power domination toolbox.

$\lvert V(G)\rvert$	number of vertices with $\deg(v) \geq 3$	$\lvert V(G')\rvert$	average contraction time	$\lvert \mathrm{Pref}(G')\rvert$	average time to calculate $\mathrm{Pref}(G')$	$\lvert \mathrm{Pev}(G')\rvert$	average time to calculate $\mathrm{Pev}(G')$
4	0	1	4.181×10^{-5}	0	4.321×10^{-4}	0	6.275×10^{-5}
5	2	4	8.187×10^{-5}	0	9.654×10^{-4}	0	1.853×10^{-4}
6	6	6	5.170×10^{-5}	0	6.929×10^{-4}	0	7.557×10^{-5}
9	3	9	9.790×10^{-5}	0	2.265×10^{-4}	0	2.263×10^{-4}
11	3	9	1.277×10^{-4}	2	2.525×10^{-4}	0	2.198×10^{-4}
14	7	13	1.442×10^{-4}	0	2.088×10^{-4}	0	5.116×10^{-4}
24	14	23	2.071×10^{-4}	0	3.165×10^{-4}	1	7.595×10^{-4}
30	12	25	2.994×10^{-4}	1	6.621×10^{-4}	0	9.014×10^{-4}
30	12	25	3.121×10^{-4}	1	6.611×10^{-4}	0	9.298×10^{-4}
33	3	9	3.653×10^{-4}	2	4.501×10^{-4}	0	4.096×10^{-4}
39	18	36	3.791×10^{-4}	3	1.293×10^{-3}	0	1.293×10^{-4}
57	24	42	6.231×10^{-4}	0	8.031×10^{-4}	1	7.121×10^{-3}
89	50	84	1.115×10^{-3}	3	2.973×10^{-3}	1	2.738×10^{-3}
118	55	101	1.612×10^{-3}	1	3.422×10^{-3}	1	4.443×10^{-3}
145	102	141	2.225×10^{-3}	4	6.811×10^{-3}	0	4.071×10^{-3}
200	73	176	3.568×10^{-3}	17	1.022×10^{-2}	0	5.235×10^{-3}
300	155	283	6.718×10^{-3}	11	3.818×10^{-2}	2	8.870×10^{-3}
1,354	496	1,233	1.119×10^{-1}	141	6.228×10^{-1}	12	1.237×10^{-1}

Table 5. Parameters for various test networks as available through the pandapower Python module.

REFERENCES.

[1] S. E. Anderson, K. L. Collins, D. Ferrero, L. Hogben, C. Mayer, A. N. Trenk, and S. Walker, "Product throttling for power domination", *Australas. J. Combin.* **85** (2023), 248–272. MR

[2] K. F. Benson, D. Ferrero, M. Flagg, V. Furst, L. Hogben, and V. Vasilevska, "Nordhaus–Gaddum problems for power domination", *Discrete Appl. Math.* **251** (2018), 103–113. MR

[3] K. F. Benson, D. Ferrero, M. Flagg, V. Furst, L. Hogben, V. Vasilevska, and B. Wissman, "Zero forcing and power domination for graph products", *Australas. J. Combin.* **70** (2018), 221–235. MR

[4] B. Bjorkman, C. Bozeman, D. Ferrero, M. Flagg, C. Grood, L. Hogben, B. Jacob, and C. Reinhart, "Power domination reconfiguration", preprint, 2022. arXiv 2201.01798

[5] C. Bozeman, B. Brimkov, C. Erickson, D. Ferrero, M. Flagg, and L. Hogben, "Restricted power domination and zero forcing problems", *J. Comb. Optim.* **37**:3 (2019), 935–956. MR

[6] R. Christie, "118 bus power flow test case", 1993, available at https://labs.ece.uw.edu/pstca/pf118/ieee118cdf.txt.

[7] P. Erdős and A. Rényi, "On random graphs, I", *Publ. Math. Debrecen* **6** (1959), 290–297. MR

[8] C. C. Fast and I. V. Hicks, "Effects of vertex degrees on the zero-forcing number and propagation time of a graph", *Discrete Appl. Math.* **250** (2018), 215–226. MR

[9] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning, "Domination in graphs applied to electric power networks", *SIAM J. Discrete Math.* **15**:4 (2002), 519–529. MR

[10] L. Hogben, J. C.-H. Lin, and B. L. Shader, *Inverse problems and zero forcing for graphs*, Mathematical Surveys and Monographs **270**, Amer. Math. Soc., Providence, RI, 2022. MR

[11] N. M. Manousakis, G. N. Korres, and P. S. Georgilakis, "Taxonomy of PMU placement methodologies", *IEEE Transactions on Power Systems* **27**:2 (2012), 1070–1077.

[12] B. McKay, "Graph formats", Available at http://users.cecs.anu.edu.au/∼bdm/data/formats.html.

[13] A. Muir and J. Lopatto, "Final report on the August 14, 2003 blackout in the United States and Canada: causes and recommendations", 2004.

[14] L. A. Smith and I. V. Hicks, "Optimal sensor placement in power grids: Power domination, set covering, and the neighborhoods of zero forcing forts", preprint, 2020. arXiv 2006.03460

[15] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "Pandapower—An open-source Python tool for convenient modeling, analysis, and optimization of electric power systems", *IEEE Transactions on Power Systems* **33**:6 (2018), 6510–6521.

RECEIVED: 26 Sep 2024 REVISED: 26 Jan 2026 ACCEPTED: 16 Mar 2026

JOHNATHAN KOCH:

johnathankoch01@gmail.com
Applied Research Solutions, Beavercreek, OH, United States

BETH BJORKMAN:

beth.morrison@us.af.mil
Air Force Research Laboratory, WPAFB, OH, United States