On the Impact of PMU Placement on Observability and Cross-Validation

Technical Report UM-CS-2012-019
Daniel Gyllstrom, Elisha Rosensweig, and Jim Kurose
Department of Computer Science. University of Massachusetts Amherst USA
{dpg, elisha, kurose}@cs.umass.edu

ABSTRACT

Significant investments have been made into deploying phasor measurement units (PMUs) on electric power grids worldwide. PMUs allow the state of the power system – the voltage phasor of system buses and current phasors of all incident transmission lines – to be directly measured. In some cases, it is also possible to infer the voltage and current phasors at neighboring buses and lines. Because PMUs are expensive, it is typically not possible to deploy enough PMUs to observe all phasors in a grid network [3, 6].

In this paper, we prove the NP-Completeness of four problems relating to PMU placements at a subset of system buses to achieve different goals: FullObserve, MaxObserve, FullObserve-XV, and MaxObserve-XV. FullObserve considers the minimum number of PMUs needed to observe all nodes, while MaxObserve considers the maximum number of buses that can be observed with a given number of PMUs. While the first of these two has been considered in the past, our formulation here generalizes the systems being considered. Next, FullObserve-XV and MaxObserve-XV consider these two problems under the constraints that PMUs must be placed "close" to each other so their measurements can be cross-validated. FullObserve-XV considers observing the entire network, while MaxObserve-XV considers maximizing the number of observed buses under this new constraint.

Motivated by their high complexity, for each problem we investigate the performance of a suitable greedy approximation algorithm for PMU placement. Through simulations, we compare the performance of these algorithms with the optimal placement of PMUs over several IEEE bus systems as well as over synthetic graphs. In our simulations these algorithms yield results that are close to optimal - for all four placement problems, the greedy algorithms yield, on average, a PMU placement that is within 97% of optimal.

1. INTRODUCTION

Significant investments have been made to deploy phasor measurement units (PMUs) on electric power grids worldwide. PMUs provide *synchronized* voltage and current measurements at a sampling rate orders of magnitude higher than the status quo: 10 to 60 samples per second rather than one sample every 1 to 4 seconds. This allows system operators to directly measure the state of the electric power grid in real-time, rather than rely on imprecise state estimation. Consequently, PMUs have the potential to enable an entirely new set of applications for the power grid: protection and control during abnormal conditions, real-time distributed control, postmortem analysis of system faults, advanced state estimators for system monitoring, and the reliable integration of renewable energy resources [1].

An electric power system consists of a set of buses – an electric substation, power generation center, or aggregation of electrical loads – and transmission lines connecting those buses. The state of a power system is defined by the voltage phasor – the magnitude and phase angle of electrical sine waves – of all system buses and the current phasor of all transmission lines. PMUs placed on buses provide real-time measurements of these system variables. However, because PMUs are expensive, they cannot be deployed on all system buses [3][6]. Fortunately, the voltage phasor at a system bus can, at times, be determined (termed *observed* in this paper) even when a PMU is not placed at that bus, by applying Ohm's and Kirchhoff's laws on the measurements taken by a PMU placed at some nearby system bus [3][4]. Specifically, with correct placement of enough PMUs at a subset of system buses, the entire system state can be determined.

In this work, we study two sets of PMU placement problems. The first problem set consists of FullObserve and MaxObserve, and considers maximizing the observability of the network via PMU placement. FullObserve considers the minimum number of PMUs needed to observe all system buses, while MaxObserve considers the maximum number of buses that can be observed with a given number of PMUs. A bus is said to be *observed* if there is a PMU placed at it or if its voltage phasor can be estimated using Ohm's or Kirchhoff's Law. Although FullObserve is well studied [3, 4, 9, 11, 14], existing work considers only networks consisting solely of zero-injection buses, an unrealistic assumption in practice, while we generalize the problem formulation to include mixtures of zero and non-zero-injection buses. Additionally, our approach for analyzing FullObserve provides the foundation with which to present the other three new (but related) PMU placement problems.

The second set of placement problems considers PMU placements that support PMU error detection. PMU mea-

surement errors have been recorded in actual systems [13]. One method of detecting these errors is to deploy PMUs "near" each other, thus enabling them to *cross-validate* eachother's measurements. FullObserve-XV aims to minimize the number of PMUs needed to observe all buses while insuring PMU cross-validation, and MaxObserve-XV computes the maximum number of observed buses for a given number of PMUs, while insuring PMU cross-validation.

We make the following contributions in this paper:

- We formulate two PMU placement problems, which (broadly) aim at maximizing observed buses while minimizing the number of PMUs used. Our formulation extends previously studied systems by considering both zero and non-zero-injection buses.
- We formally define graph-theoretic rules for PMU cross-validation. Using these rules, we formulate two additional PMU placement problems that seek to maximize the observed buses while minimizing the number of PMUs used under the condition that the PMUs are cross-validated.
- We prove that all four PMU placement problems are NP-Complete. This represents our most important contribution.
- Given the proven complexity of these problems, we evaluate heuristic approaches for solving these problems. For each problem we describe a greedy algorithm, and prove that each greedy algorithm has polynomial running time.
- Using simulations, we evaluate the performance of our greedy approximation algorithms over synthetic and actual IEEE bus systems. We find that the greedy algorithms yield a PMU placement that is, on average, within 97% optimal. Additionally, we find that the cross-validation constraints have limited effects on observability: on average our greedy algorithm that places PMUs according to the cross-validation rules observes only 5.7% fewer nodes than the same algorithm that does not consider cross-validation.

The rest of this paper is organized as follows. In Section 2 we introduce our modeling assumptions, notation, and observability and cross-validation rules. In Section 3 we formulate and prove the complexity of our four PMU placement problems. Section 4 presents the approximation algorithms for each problem and Section 5 considers our simulation-based evaluation. We conclude with a review of related work (Section 6) and concluding remarks (Section 7).

2. PRELIMINARIES

In this section we introduce notation and underlying assumptions (Section 2.1), and define our observability (Section 2.2) and cross-validation (Section 2.3) rules.

2.1 Assumptions, Notation, and Terminology

Consistent with the conventions in [3, 4, 5, 11, 14, 15], we make the following assumptions about PMU placements and buses. First, a PMU can only be placed on a bus. Second, a PMU on a bus measures the voltage phasor at the bus and the current phasor of all transmission lines connected to it.

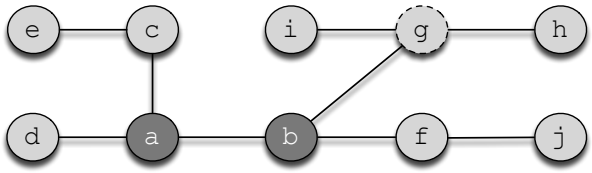


Figure 1: Example power system graph. PMU nodes (a, b) are indicated with darker shading. Injection nodes have solid borders while zero-injection nodes (g) have dashed borders.

We model a power grid as an undirected graph $G = (V, E)$. Each $v \in V$ represents a bus. A bus is either an electrical substation, a power generation center, or an aggregation of loads. $V = V_Z \cup V_I$, where V_Z is the set of all zero-injection buses and V_I is the set of all non-zero-injection buses. A bus is zero-injection if it has no load nor generator [16]. All other buses are non-zero-injection. For simplicity, we refer to non-zero-injection buses as injection buses in the remainder of the paper. Each $(u, v) \in E$ is a transmission line connecting buses u and v. Figure 1 is an example of a power system modeled as such an undirected graph.

Using the same notation as Brueni and Heath [4], we define two Γ functions. For $v \in V$ let $\Gamma(v)$ be the set of v's neighbors in G, and $\Gamma[v] = \Gamma(v) \cup \{v\}$. A PMU placement $\Phi_G \subseteq V$ is a set of nodes at which PMUs are placed, and $\Phi_G^R \subseteq V$ is the set of observed nodes for graph G with placement Φ_G (see definition of observability below). $k^* = \min\{|\Phi_G| : \Phi_G^R = V\}$ denotes the minimum number of PMUs needed to observe the entire network. Where the graph G is clear from the context, we drop the G subscript.

For convenience, we refer to any node with a PMU as a *PMU node*. Additionally, for a given PMU placement we say that set $W \subseteq V$ is observed if all nodes in W are observed, and if $W = V$ we refer to the graph as *fully observed*.

2.2 Observability Rules

We use the simplified observability rules elegantly formulated by Brueni and Heath [4]. We restate the rules here:

1. **Observability Rule 1 (O1)**. *If node v is a PMU node, then $\Gamma[v]$ is observed. Formally, if $v \in \Phi_G$, then $\Gamma[v] \subseteq \Phi_G^R$.*
2. **Observability Rule 2 (O2)**. *If a zero-injection node, v, is observed and $\Gamma(v)\backslash\{u\}$ is observed for some $u \in \Gamma(v)$, then $\Gamma[v]$ is observed. Formally, if $v \in \Phi_G^R \cap V_Z$ and $|\Gamma(v) \cap (V - \Phi_G^R)| \leq 1$, then $\Gamma[v] \subseteq \Phi_G^R$.*

Consider the example in Figure 1, where the shaded nodes are PMU nodes and g is the only zero-injection node. Nodes $a - d$ are observed by applying O1 at the PMU at a, and nodes a, b, f and g are observed by applying O1 at b. e cannot be observed via c because c does not have a PMU (O1 does not apply) and is an injection node (O2 does not apply). Similarly, j is not observed via f. Finally, although $g \in V_Z$, O2 cannot be applied at g because g has two unobserved neighbors i, h, so they remain unobserved.

Since O2 only applies with zero-injection nodes, more nodes are likely observed when nodes are zero-injection. For example, consider the case where c and f are *zero-injection* nodes. $a - d$, g and f are still observed as before, as O1

makes no conditions on the node type. Additionally, since $c, f \in V_Z$ and each has a single unobserved neighbor, we can apply O2 at each of them to observe e, j, respectively. We evaluate the effect of increasing the number of zero-injection nodes on observability in our simulations (Section 5.2).

2.3 Cross-Validation Rules

From Vanfretti et al. [13], PMU measurements can be cross-validated when: (1) a voltage phasor of a non-PMU bus can be computed by PMU data from two different buses or (2) the current phasor of a transmission line can be computed from PMU data from two different buses. [1] Although it is the PMU data that is actually being cross-validated, for convenience, we say a PMU is cross-validated. A PMU is *cross-validated* if one of the rules below is satisfied [13]:

1. **Cross-Validation Rule 1 (XV1)**. *If two PMU nodes are adjacent, then the PMUs cross-validate each other. Formally, if $u, v \in \Phi_G$, $u \in \Gamma(v)$, then the PMUs at u and v are cross-validated.*

2. **Cross-Validation Rule 2 (XV2)**. *If two PMU nodes have a common neighbor, then the PMUs cross-validate each other. Formally, if $u, v \in \Phi_G$, $u \neq v$ and $\Gamma(u) \cap \Gamma(v) \neq \emptyset$, then the PMUs at u and v are cross-validated.*

In short, the cross-validation rules require that *the PMU is within two hops of another PMU*. For example, in Figure 1, the PMUs at a and b cross-validate each other by XV1.

XV1 derives from the fact that both PMUs are measuring the current phasor of the transmission line connecting the two PMU nodes. XV2 is more subtle. Using the notation specified in XV2, when computing the voltage phasor of an element in $\Gamma(u) \cap \Gamma(v)$ the voltage equations include variables to account for measurement error (e.g., angle bias) [12]. When the PMUs are two hops from each other, there are more equations than unknowns, allowing for measurement error detection. Otherwise, the number of unknown variables exceeds the number of equations, which eliminates the possibility of detecting measurement errors [12].

3. PROBLEM FORMULATIONS AND NP-COMPLETENESS PROOFS

In this section we define four PMU placement problems and prove the NP-Completeness of each. We begin with a general overview of NP-Completeness, as well as a high-level description of our proof strategy in this paper (Section 3.1). In the remainder of Section 3 we present and prove the NP-Completeness of four PMU placement problems, in the following order: FullObserve (Section 3.2), MaxObserve (Section 3.3), FullObserve-XV (Section 3.4), and MaxObserve-XV (Section 3.5).

In all four problems defined in this paper, we are only concerned with computing the voltage phasors of each bus (i.e., observing the buses). Using the values of the voltage phasors, Ohm's Law can be easily applied to compute the current phasors of each transmission line. Also, we consider networks with both injection and zero-injection buses. For similar proofs for purely zero-injection systems, see our Technical Report [8].

[1] Vanfretti et al. [13] use the term "redundancy" instead of cross-validation.

3.1 NP-Completeness Overview and Proof Strategy

Before proving that our PMU placement problems are NP-Complete (abbreviated NPC), we provide some background on NP-Completeness. NPC problems are the hardest problems in complexity class $\mathcal{NP}$. It is generally assumed that solving NPC problems is hard, meaning that any algorithm that solves an NPC problem has exponential running time as function of the input size. It is important to clarify that despite being NPC, a *specific* problem instance might be efficiently solvable. This is either due to the special structure of the specific instance or because the input size is small, yielding a small exponent. For example, in Section 5 we are able to solve FullObserve for small IEEE bus topologies due to their small size. Thus, by establishing that our PMU placement problems are NPC, we claim that there *exist* bus topologies for which these problems are difficult to solve (i.e., no known polynomial-time algorithm exists to solve those case).

To prove our problems are NPC, we follow the standard three-step reduction procedure. For a decision problem Π, we first show $\Pi \in \mathcal{NP}$. Second, we select a known NPC problem, denoted Π', and construct a polynomial-time transformation, f, that maps any instance of Π' to an instance of Π. Finally, we must ensure that for this f, $x \in \Pi' \Leftrightarrow f(x) \in \Pi$ [7].

Next, we outline the proof strategy we use throughout the paper. In Sections 3.2 through Section 3.5 we use slight variations of the approach presented by Brueni and Heath in [4] to prove the problems we consider here are NPC. In general we found their scheme to be elegantly extensible for proving many properties of PMU placements.

In [4], the authors prove NP-Completeness by reduction from planar 3-SAT (P3SAT). A 3-SAT formula, ϕ, is a boolean formula in conjunctive normal form (CNF) such that each clause contains at most 3 literals. For any 3-SAT formula ϕ with the sets of variables $\{v_1, v_2, \ldots, v_r\}$ and clauses $\{c_1, c_2, \ldots, c_s\}$, $G(\phi)$ is the bipartite graph $G(\phi) = (V(\phi), E(\phi))$ defined as follows:

$$\begin{aligned} V(\phi) &= \{v_i \mid 1 \leq i \leq r\} \cup \{c_j \mid 1 \leq j \leq s\} \\ E(\phi) &= \{(v_i, c_j) \mid v_i \in c_j \ \ or \ \ \overline{v_i} \in c_j\}. \end{aligned}$$

Note that edges pass only between v and c nodes, and so the graph is bipartite. P3SAT is a 3-SAT formula s.t. $G(\phi)$ is planar [10]. For example, P3SAT formula

$$\begin{aligned} \varphi &= (\overline{v_1} \vee v_2 \vee v_3) \wedge (\overline{v_1} \vee \overline{v_4} \vee v_5) \wedge (\overline{v_2} \vee \overline{v_3} \vee \overline{v_5}) \\ &\quad \wedge (v_3 \vee \overline{v_4}) \wedge (\overline{v_3} \vee v_4 \vee \overline{v_5}) \end{aligned} \quad (1)$$

has graph $G(\varphi)$ shown in Figure 2. Discovering a satisfying assignment for P3SAT is an NPC problem, and so it can be used in a reduction to prove the complexity of the problems we address here. Note that in this work we will use φ to denote a specific P3SAT formula, while ϕ will be used to denote a generic P3SAT formula.

Following the approach in [4], for P3SAT formula, ϕ, we replace each variable node and each clause node in $G(\phi)$ with a specially constructed set of nodes, termed a *gadget*. In this work, all variable gadgets will have the same structure, and all clause gadgets have the same structure (that is different from the variable gadget structure), and we denote the resulting graph as $H(\phi)$. In $H(\phi)$, each *variable* gadget has a subset of nodes that semantically represent assigning "True"

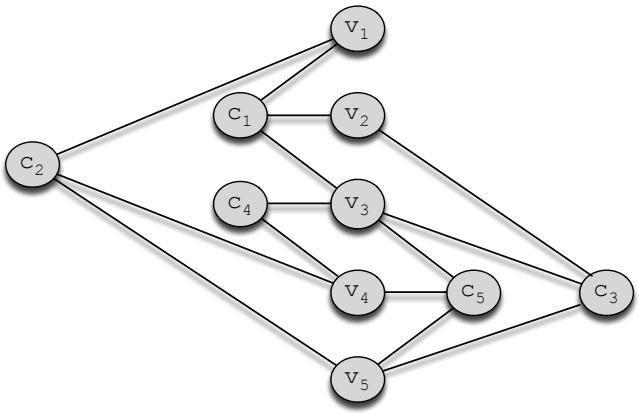
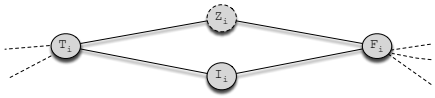
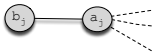


Figure 2: $G(\varphi) = (V(\varphi), E(\varphi))$ formed from φ in Equation (1)



(a) Variable gadget V_i used in Theorem 3.1 and Theorem 3.4. The dashed edges are connections to clause gadgets. Z_i is a zero-injection node and all other nodes are injection nodes.



(b) Clause gadget C_j used in Theorem 3.4. The dashed edges are connections to variable gadgets. a_j and b_j are both injection nodes.

Figure 3: Gadgets used in Theorem 3.1 and Theorem 3.4.

to that variable, and a subset of nodes that represent assigning it "False". When a PMU is placed at one of these nodes, this is interpreted as assigning a truth value to the P3SAT variable corresponding with that gadget. Thus, we use the PMU placement to determine a consistent truth value for each P3SAT variable. Also, clause gadgets are connected to variable gadgets at either "True" or "False" (but never both) nodes, in such a way that the clause is satisfied if and only if *at least one* of those nodes has a PMU.

We also note that while we assume $G(\phi)$ is planar, we make no such claim regarding $H(\phi)$, though in practice all graphs used in our proofs are indeed planar. The proof of NPC rests on the fact that solving the underlying ϕ formula is NPC.

In what follows, for a given PMU placement problem Π, we prove Π is NPC by showing that a PMU placement in $H(\phi)$, Φ, can be interpreted semantically as describing a satisfying assignment for ϕ iff $\Phi \in \Pi$. Since P3SAT is NPC, this proves Π is NPC as well.

3.2 The FullObserve Problem

The FullObserve problem was previously discussed in the literature (e.g., the PMUP problem in [4], and the PDS problem in [9]) for purely zero-injection bus systems. Here we consider networks with mixtures of injection and zero-injection buses, and modify the NPC proof for PMUP in [4] to handle this mixture.

FullObserve Optimization Problem:

- Input: Graph $G = (V, E)$ where $V = V_Z \cup V_I$ and $V_Z \neq \emptyset$. [2]
- Output: A placement of PMUs, Φ_G, such that $\Phi_G^R = V$ and Φ_G is minimal.

FullObserve Decision Problem:

- Instance: Graph $G = (V, E)$ where $V = V_Z \cup V_I$, $V_Z \neq \emptyset$, k PMUs such that $k \geq 1$.
- Question: Is there a Φ_G such that $|\Phi_G| \leq k$ and $\Phi_G^R = V$?

Theorem 3.1. FullObserve *is NP-Complete.*

Proof Idea: We introduce a problem-specific variable gadget. We show that in order to observe all nodes, PMUs must be placed on variable gadgets, specifically on nodes that semantically correspond to True and False values that satisfy the corresponding P3SAT formula.

For our first problem, we use a single node as a clause gadget denoted a_j, and the subgraph shown in Figure 3(a) as the variable gadget. Note that in the variable gadget, all the nodes are injection nodes except for Z_i. For this subgraph, we state the following simple lemma:

Lemma 3.2. *Consider the gadget shown in Figure 3(a), possibly with additional edges connected to T_i and/or F_i. Then (a) nodes I_i, Z_i are not observed if there is no PMU on the gadget, and (b) all the nodes in the gadget are observed with a single PMU iff the PMU is placed on either T_i or F_i.*

Proof. (a) If there is no PMU on the gadget, O1 cannot be applied at any of the nodes, and so we must resort to O2. We assume no edges connected to I_i, Z_i from outside the gadget, and since $T_i, F_i \in V_I$, we cannot apply O2 at them, which concludes our proof.

(b) In one direction, if we have a PMU placed at T_i, from O1 we can observe Z_i, I_i. Since Z_i is zero-injection and one neighbor, T_i has been observed, from O2 at Z_i we can observe F_i. The same holds for placing a PMU at F_i, due to symmetry.

In the other direction, by placing a PMU at I_i (Z_i) we observe T_i and F_i via O1. However, since $F_i, T_i \notin V_Z$, O2 cannot be applied at either of them, so Z_i (I_i) will not be observed. □

Proof of Theorem 3.1. We start by arguing that FullObserve $\in \mathcal{NP}$. First, nondeterministically select k nodes in which to place PMUs. Using the rules specified in Section 2.2, determining the number of observed nodes can be done in linear time.

To show FullObserve is NP-hard, we reduce from P3SAT. Let ϕ be an arbitrary P3SAT formula with variables $\{v_1, v_2, \ldots, v_r\}$ and the set of clauses $\{c_1, c_2, \ldots, c_s\}$, and $G(\phi)$ the corresponding planar graph. We use $G(\phi)$ to construct a new graph $H_0(\phi) = (V_0(\phi), E_0(\phi))$ by replacing each variable

[2] We include the condition that $V_Z \neq \emptyset$ because otherwise FullObserve reduces to Vertex-Cover, making the NP-Completeness proof trivial.

node in $G(\phi)$ with the variable gadget shown in Figure 3(a). The clause nodes consist of a single node (i.e., are the same as in $G(\phi)$). We denote the node corresponding to c_j as a_j. All clause nodes are injection nodes. In the remainder of this proof we let $H := H_0(\phi)$. In total, V_Z contains all Z_i nodes for $1 \leq i \leq r$, and all other nodes are in V_I. The edges connecting clause nodes with variable gadgets express which variables are in each clause: for each clause node a_j, $(T_i, a_j) \in E_0(\phi) \Leftrightarrow v_i \in c_j$, and $(F_i, a_j) \in E_0(\phi) \Leftrightarrow \overline{v_i} \in c_j$. As a result, the following observation holds:

Observation 3.3. *For a given truth assignment and a corresponding PMU placement, a clause c_j is satisfied iff a_j is attached to a node in a variable gadget with a PMU.*

The resulting graph for the example given in Figure 2 is shown in Figure 4. Nodes with a dashed border are zero-injection nodes. [3] The corresponding formula for this graph, φ, is satisfied by truth assignment A_φ: $\overline{v_1}, \overline{v_2}, v_3, \overline{v_4}$, and $\overline{v_5}$ are True. This corresponds to the dark shaded nodes in Figure 4. While this construction generates a graph with very specific structure, in Section 3.6, we detail how to extend our proof to consider graphs with a wider range of structures.

With this construct in place, we move on to our proof. We show that ϕ is satisfiable if and only if $k = r = |\Phi_H|$ PMUs can be placed on H such that $\Phi_H^R = V$.

($\Rightarrow$) Assume ϕ is satisfiable by truth assignment A_ϕ. Then, consider the placement Φ_H such that for each variable gadget V_i, $T_i \in \Phi_H \Leftrightarrow v_i = True$ in A_ϕ, and $F_i \in \Phi_H \Leftrightarrow v_i = False$. From Lemma 3.2(b) we know that all nodes in variable gadgets are observed by such a placement. From Observation 3.3, all clause nodes are observed because our PMU assignment is based on a satisfying assignment. Thus, we have shown that $\Phi_H^R = V$.

($\Leftarrow$) Suppose there is a placement of r PMUs, Φ_H, such that $\Phi_H^R = V$. From Lemma 3.2(a) we know that for each V_i with no PMU, at least two nodes are not observed, so each V_i must have a PMU placed in it. Since we have only r PMUs, that means one PMU per gadget. From Lemma 3.2(b) we know this PMU must be placed on T_i or F_i, since otherwise the gadget will not be fully observed. Note that these nodes are all in V_I.

Since we assume the graph is fully observed, all a_j are observed by Φ_H. Because we just concluded that PMUs are placed only on injection nodes in the variable gadgets, each clause node a_j can only be observed via application of O1 at T_i/F_i nodes to which it is attached – specifically, a_j is attached to a node with a PMU. From Observation 3.3 this means that all clauses are satisfied by the semantic interpretation of our PMU placement, which concludes our proof. □

3.3 The MaxObserve Problem

MaxObserve is a variation of FullObserve: rather than consider the minimum number of PMUs required for full system observability, MaxObserve finds the maximum number of nodes that can be observed using a fixed number of PMUs.

MaxObserve Optimization Problem:

- Input: Graph $G = (V, E)$ where $V = V_Z \cup V_I$, k PMUs such that $1 \leq k < k^*$.
- Output: A placement of k PMUs, Φ_G, such that $|\Phi_G^R|$ is maximum.

MaxObserve Decision Problem:

- Instance: Graph $G = (V, E)$ where $V = V_Z \cup V_I$, k PMUs such that $1 \leq k < k^*$.
- Question: For a given $m < |V|$, is there a Φ_G such that $|\Phi_G| \leq k$ and $m \leq |\Phi_G^R| < |V|$?

[3] Throughout this paper, nodes with dashed borders denote zero-injection nodes.

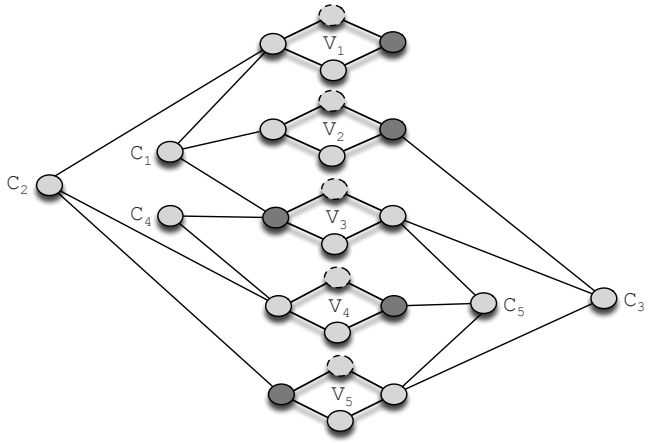


Figure 4: Graph $G = (V, E) = H_1(\varphi)$ formed from φ formula in Theorem 3.1 proof. Nodes with a dashed border are zero-injection nodes.

Theorem 3.4. MaxObserve *is NP-Complete.*

Proof Idea: First, we construct problem-specific gadgets for variables and clauses. We then demonstrate that any solution that observes m nodes must place the PMUs only on nodes in the variable gadgets. Next we show that as a result of this, the problem of observing m nodes in this graph reduces to Theorem 3.1.

Proof. MaxObserve $\in \mathcal{NP}$ using the same argument in the proof for Theorem 3.1.

Next, we reduce from P3SAT as in the proof for Theorem 3.1, where ϕ is an arbitrary P3SAT formula. We create a new graph $H_1(\phi) = (V_1(\phi), E_1(\phi))$ which is identical to $H_0(\phi)$ from the previous proof, except that each clause node in $H_0(\phi)$ is replaced with the clause gadget shown in Figure 3(b), comprising of two injection nodes. As before, the edges connecting clause nodes with variable gadgets express which variables are in each clause: for each clause node a_j, $(T_i, a_j) \in E_1(\phi) \Leftrightarrow v_i \in c_j$, and $(F_i, a_j) \in E_1(\phi) \Leftrightarrow \overline{v_i} \in c_j$. Note that Observation 3.3 holds here as well.

We are now ready to show MaxObserve is NP-hard. For convenience, we let $H := H_1(\phi)$. Recall ϕ has r variables and s clauses. Here we consider the instance of MaxObserve where $k = r$ and $m = 4r + s$, and show that ϕ is satisfiable if and only if $r = |\Phi_H|$ PMUs can be placed on H such that $m \leq |\Phi_H^R| < |V|$. In Section 3.6 we discuss how to extend this proof for any larger value of m and different $\frac{|V_Z|}{|V_I|}$ ratios.

($\Rightarrow$) Assume ϕ is satisfiable by truth assignment A_ϕ. Then, consider the placement Φ_H such that for each variable gadget V_i, $T_i \in \Phi_H \Leftrightarrow v_i = True$ in A_ϕ, and $F_i \in \Phi_H \Leftrightarrow v_i = False$. In the proof for Theorem 3.1 we demonstrated such

a placement will observe all nodes in $H_0(\phi) \subset H_1(\phi)$, and using the same argument it can easily be checked that these nodes are still observed in $H_1(\phi)$. Each b_j node remains unobserved because each $a_j \in V_I$ and consequently O2 cannot be applied at a_j. Since $|H_0(\phi)| = 4r + s = m$, we have observed the required nodes.

($\Leftarrow$) We begin by proving that any solution that observes m nodes must place the PMUs only on nodes in the variable gadgets. By construction, each PMU is either on a clause gadget or a variable gadget, but not both. Let $0 \leq t \leq r$ be the number of PMUs on clause gadgets, we wish to show that for the given placement $t = 0$. First, note that *at least* $\max(s-t, 0)$ clause gadgets are without PMUs, and that for each such clause (by construction) at least one node (b_i) is not observed. Next, from Lemma 3.2(a) we know that for each variable gadget without a PMU, at least two nodes are not observed.

Denote the *unobserved* nodes for a given PMU placement as Φ_H^-. Thus, we get $|\Phi_H^-| \geq 2t + \max((s-t), 0)$. However, since m nodes are observed and $|V| - m \leq s$, we get $|\Phi_H^-| \leq s$, so we know $s \geq 2t + \max((s-t), 0)$. We consider two cases:

- $s \geq t$: then we get $s \geq t + s \Rightarrow t = 0$.
- $s < t$: then we get $s \geq 2t$, and since we assume here $0 \leq s < t$ this leads to a contradiction and so this case cannot occur.

Thus, the r PMUs must be on nodes in variable gadgets. Note that the variable gadgets in $H_1(\phi)$ have the same structure as in $H_0(\phi)$. We return to this point shortly.

Earlier we noted that for each clause gadget without a PMU, the corresponding b_j node is unobserved, which comes to s nodes. To observe $m = 4r + s$ nodes, we will need to observe all the remaining nodes. Thus, we have reduced the problem to that of observing all of $H_0(\phi) \subset H_1(\phi)$. Our proof for Theorem 3.1 demonstrated this can only be done by placing PMUs at nodes corresponding to a satisfying assignment of ϕ, and so our proof is complete. □

3.4 The FullObserve-XV Problem

FullObserve-XV Optimization Problem:

- Input: Graph $G = (V, E)$ where $V = V_Z \cup V_I$.
- Output: A placement of PMUs, Φ_G, such that $\Phi_G^R = V$, and Φ_G is minimal under the condition that each $v \in \Phi_G$ is cross-validated according to the rules specified in Section 3.4.

FullObserve-XV Decision Problem:

- Instance: Graph $G = (V, E)$ where $V = V_Z \cup V_I$, k PMUs such that $k \geq 1$.
- Question: Is there a Φ_G such that $|\Phi_G| \leq k$ and $\Phi_G^R = V$ under the condition that each $v \in \Phi_G$ is cross-validated?

Theorem 3.5. FullObserve-XV *is NP-Complete.*

Proof Idea: We show FullObserve-XV is NP-hard by reducing from P3SAT. We create a single-node gadget for clauses (as for FullObserve) and the gadget shown in Figure 5 for each variable. Each variable gadget here comprises of two disconnected components, and there are two T_i and two F_i nodes, one in each component. First, we show that each variable gadget must have 2 PMUs for the entire graph to be observed, one PMU for each subgraph. Then, we show that cross-validation constraints force PMUs to be placed on both T nodes or both F nodes. Finally, we show how to use the PMU placement to derive a satisfying P3SAT truth assignment.

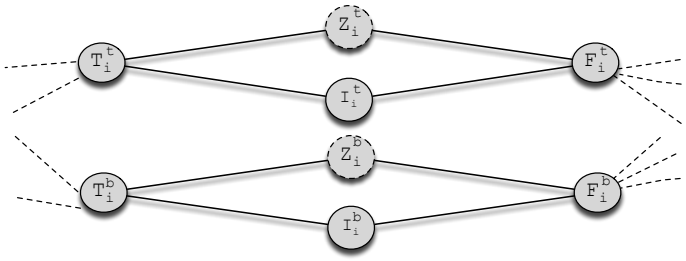


Figure 5: Variable gadget used in Theorem 3.5 proof. The gadget has two disconnected subgraphs, where the superscript, t, denotes nodes in the upper subgraph and superscript, b, indexes nodes in the lower subgraph. The dashed edges are connections to clause gadgets.

Lemma 3.6. *Consider the gadget shown in Figure 5, possibly with additional nodes attached to T_i and/or F_i nodes. (a) nodes I_i^t, Z_i^t are not observed if there is no PMU on V_i^t, and (b) all the nodes in V_i^t are observed with a single PMU iff the PMU is placed on either T_i^t or F_i^t. Due to symmetry, the same holds when considering V_i^b.*

Proof. The proof is straightforward from the proof of Lemma 3.2, since both V_i^t and V_i^b are identical to the gadget from Figure 3(a), which Lemma 3.2 refers to. □

Proof of Theorem 3.5. First, we argue that FullObserve-XV $\in \mathcal{NP}$. Given a FullObserve-XV solution, we use the polynomial time algorithm described in our proof for Theorem 3.1 to determine if all nodes are observed. Then, for each PMU node we run a breadth-first search, stopping at depth 2, to check that the cross-validation rules are satisfied.

To show FullObserve-XV is NP-hard, we reduce from P3SAT. Our reduction is similar to the one used in Theorem 3.1. We start with the same P3SAT formula ϕ with variables $\{v_1, v_2, \ldots, v_r\}$ and the set of clauses $\{c_1, c_2, \ldots, c_s\}$.

For this problem, we construct $H_2(\phi)$ in the following manner. We use the single-node clause gadgets as in $H_0(\phi)$, and as before, the edges connecting clause nodes with variable gadgets shown in Figure 5 express which variables are in each clause: for each clause node a_j, $(T_i^t, a_j), (T_i^b, a_j) \in E_1(\phi) \Leftrightarrow v_i \in c_j$, and $(F_i^t, a_j), (F_i^b, a_j) \in E_1(\phi) \Leftrightarrow \overline{v_i} \in c_j$. For notational simplicity, we shall use H to refer to $H_2(\phi)$. Note that once again, by construction Observation 3.3 holds for H.

Moving on, we now show that ϕ is satisfiable if and only if $k = 2r$ PMUs can be placed on H such that H is fully observed under the condition that all PMUs are cross-validated, and that $2r$ PMUs are the minimal bound for observing the graph with cross-validation.

($\Rightarrow$) Assume ϕ is satisfiable by truth assignment A_ϕ. For each $1 \leq i \leq r$, if $v_i = True$ in A_ϕ we place a PMU at T_i^b and at T_i^t of the variable gadget V_i. Otherwise, we place a PMU at F_i^b and at F_i^t of this gadget. From the fact that A_ϕ

is satisfying and Observation 3.3, we know the PMU nodes in V_i must be adjacent to some clause node[4], making T_i^t (F_i^t) two hops away from T_i^b (F_i^b). Therefore, all PMUs are cross-validated by XV2.

Assignment Φ_H observes all $v \in V$: from Lemma 3.6(b) we know the assignment fully observes all the variable gadgets. From Observation 3.3 we know all clause nodes are adjacent to a node with a PMU, so they are observed via O1, which concludes this direction of the theorem.

($\Leftarrow$) Suppose Φ_G observes all nodes in H under the condition that each PMU is cross-validated, and that $|\Phi_H| = 2r$. We want to show that ϕ is satisfiable by the truth assignment derived from Φ_H. We do so following a similar method as for the previous Theorems.

From Lemma 3.6(a) we know that each component in each variable gadget must have at least one PMU in order for the entire graph to be observed. Since we have $2r$ PMUs and $2r$ components, each component will have a single PMU. This also means there are no PMUs on clause gadgets.

From Lemma 3.6(b) we know that full observability will require PMUs be on either T or F nodes in each variable gadget. As a result, cross-validation constraints require for each variable gadget that both PMUs are either on T_i^t, T_i^b or F_i^t, F_i^b. This is because any T_i^t (F_i^t) is four hops or more away from any other T/F node. Since we assume the clause nodes are all observed and we know no PMUs are on clause nodes, from Observation 3.3 this means the PMU placement satisfies all clauses, which concludes our proof.

3.5 The MaxObserve-XV Problem

MaxObserve-XV Optimization Problem:

- Input: Graph $G = (V, E)$ where $V = V_Z \cup V_I$ and k PMUs such that $1 \leq k < k^*$.
- Output: A placement of k PMUs, Φ_G, such that $|\Phi_G^R|$ is maximum under the condition that each $v \in \Phi_G$ is cross-validated according to the rules specified in Section 3.4.

MaxObserve-XV Decision Problem:

- Instance: Graph $G = (V, E)$ where $V = V_Z \cup V_I$, k PMUs such that $1 \leq k < k^*$, and some $m < |V|$.
- Question: Is there a Φ_G such that $|\Phi_G| \leq k$ and $m \leq |\Phi_G^R| < |V|$ under the condition that each $v \in \Phi_G$ is cross-validated?

Theorem 3.7. MaxObserve-XV *is NP-Complete.*

Proof Idea: We show MaxObserve-XV is NP-hard by reducing from P3SAT. Our proof is a combination of the NP-hardness proofs for MaxObserve and FullObserve-XV. From a P3SAT formula, ϕ, we create a graph $G = (V, E)$ with the clause gadgets from MaxObserve (Figure 3(b)) and the variable gadgets from FullObserve-XV (Figure 5). The edges in G are identical the ones the graph created in our reduction for FullObserve-XV.

We show that any solution that observes $m = |V| - s$ nodes must place the PMUs exclusively on nodes in the variable gadgets. As a result, we show 1 node in each clause gadget – b_j for clause C_j – is not observed, yielding a total s unobserved nodes. This implies all other nodes must be observed, and thus reduces our problem to the scenario considered in Theorem 3.5, which is already proven.

Proof. MaxObserve-XV is easily in $\mathcal{NP}$. We verify a MaxObserve-XV solution using the same polynomial time algorithm described in our proof for Theorem 3.5.

We reduce from P3SAT to show MaxObserve-XV is NP-hard. Our reduction is a combination of the reductions used for MaxObserve and FullObserve-XV. Given a P3SAT formula, ϕ, with variables $\{v_1, v_2, \ldots, v_r\}$ and the set of clauses $\{c_1, c_2, \ldots, c_s\}$, we form a new graph, $H_3(\phi) = (V(\phi), E(\phi))$ as follows. Each clause c_j corresponds to the clause gadget from MaxObserve (Figure 3(b)) and the variable gadgets from FullObserve-XV (Figure 5). As in Theorem 3.5, we refer to the upper subgraph of variable gadget, V_i, as V_i^t and the lower subgraph as V_i^b. Also, we denote here $H := H_3(\phi)$.

Let $k = 2r$ and $m = 8r + s = |V| - s$. As in our NP-hardness proof for MaxObserve, m includes all nodes in H except b_j of each clause gadget. We need to show that ϕ is satisfiable if and only if $2r$ cross-validated PMUs can be placed on H such that $m \leq |\Phi_H^R| < |V|$.

($\Rightarrow$) Assume ϕ is satisfiable by truth assignment A_ϕ. For each $1 \leq i \leq r$, if $v_i = True$ in A_ϕ we place a PMU at T_i^b and at T_i^t of the variable gadget V_i. Otherwise, we place a PMU at F_i^b and at F_i^t of this gadget. In either case, the PMU nodes in V_i must be adjacent to a clause node, making T_i^t (F_i^t) two hops away from T_i^b (F_i^b)[5]. Therefore, all PMUs are cross-validated by XV2.

This placement of $2r$ PMUs, Φ_H, is exactly the same one derived from ϕ's satisfying instance in Theorem 3.5. Since Φ_H only has PMUs on variable gadgets, all a_j nodes are observed by the same argument used in Theorem 3.5. Thus, at least $8r + s$ nodes are observed in H. Because no PMU in Φ_H is placed on a clause gadget, C_j, and O2 cannot be applied at a_j since $a_j \in V_I$, we know that no b_j is observed. We conclude that exactly m nodes are observed with Φ_H.

($\Leftarrow$) We begin by proving that any solution that observes m nodes must place the PMUs only on nodes in the variable gadgets. Assume that there are $1 < t \leq r$ variable gadgets without a PMU. Then, at most t PMUs are on nodes in clause gadgets, so *at least* $\max(s - t, 0)$ clause gadgets are without PMUs. We want to show here that for $m = 8r + s$, $t = 0$.

To prove this, we rely on the following observations:

- As shown in Theorem 3.5, a variable gadget's subgraph with no PMU has at least 2 unobserved nodes.
- In any clause gadget C_j, b_j nodes cannot be observed if there is no PMU somewhere in C_j.

Thus, given some t, $|\Phi_H^-| \geq 2t + \max(s - t, 0)$, where Φ_H^- denotes the unobserved nodes in H. Since $|V| - m \leq s$, we know $|\Phi_H^-| \leq s$ and thus $s \geq 2t + \max(s - t, 0)$. We consider two cases:

- $s \geq t$: then we get $s \geq s + t \Rightarrow t = 0$.
- $s < t$: then we get $s \geq 2t$, and since we assume here $0 \leq s < t$ this leads to a contradiction and so this case cannot occur.

[4]Each variable must be used in at least a single clause, or it is not considered part of the formula. If there is a variable that has no impact on the truth value of ϕ, we always place the PMUs on two nodes (both T or both F) that are adjacent to a clause node.

[5]See previous note on FullObserve-XV

Thus, we have concluded that the $2r$ PMUs must be on variable gadgets, leaving all clause gadgets without PMUs. We now observe that for each clause gadget C_j, such a placement of PMUs cannot observe nodes of type b_j, which amounts to a total of s unobserved nodes – the allowable bound. This means that all other nodes in H must be observed in order for the requirement to be met. Specifically this is exactly all the nodes in $H_2(\phi)$ from the Theorem 3.5 proof. Since PMUs can only be placed on variable gadgets – all of which are included $H_2(\phi)$ – we have reduced the problem to the problem in Theorem 3.5. We use the Theorem 3.5 proof to determine that all clauses in ϕ are satisfied by the truth assignment derived from Φ_H. □

3.6 Proving NPC for additional topologies

A quick review of our NPC proofs reveals that the graphs are carefully constructed regarding our selection of $|V_Z|, |V_I|$ and (where relevant) m. From a purely theoretical standpoint this is sufficient to prove that the class of problems is NPC. However, we argue that the NPC of these problems holds for a much wider range of topologies. To support this claim, in this section we show that slight adjustments to the variable and/or clause gadgets can generate a wide selection of graphs – changing $|V_Z|, |V_I|$ and (where relevant) m and $m/|V|$ – in which the same proofs from Section 3.2 - Section 3.5 can be applied. We present the outline for new gadget constructions and leave the detailed analysis to the reader.

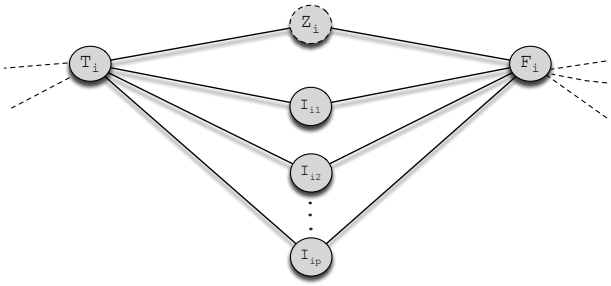
The *number of injection nodes* $|V_I|$ for each of our four problem definitions can be increased by introducing new variable gadgets. For FullObserve and MaxObserve, we use the variable gadget shown in Figure 6(a) in place of the original variable gadget (Figure 3(a)). Our proofs for Theorem 3.1 and Theorem 3.4 can remain largely unchanged because the same PMU placement described in each NP-Completeness proof observes these newly introduced nodes. [6] For FullObserve-XV and MaxObserve-XV we increase $|V_I|$ using the variable gadget shown in Figure 6(b). The PMU placements described in the proofs for Theorem 3.5 and Theorem 3.7 observe all newly introduced nodes in Figure 6(b).

Similarly, the *number of zero-injection nodes* $|V_Z|$ can modified by changing the variable gadgets. FullObserve and MaxObserve – using the variable gadget shown in Figure 7(a) – and FullObserve-XV and MaxObserve-XV – using the variable gadget shown in Figure 7(a) – are easily extended to include more zero-injection nodes. By repeatedly applying O2 at the newly introduced zero-injection nodes, all variable gadget nodes are observed using the same PMU placement described in the NP-Completeness proofs for each problem. For this reason, our proofs only require slight modifications.

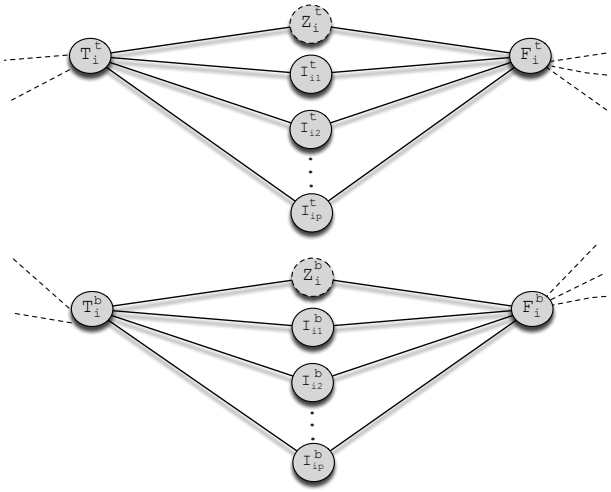
In the MaxObserve-XV and MaxObserve proofs we demonstrated NPC for $m = |V| - s$. In order to increase the size of $|V|$ while keeping m the same, we replace each clause gadget, C_j for $1 \leq j \leq s$, with a new clause gadget, C'_j, shown in Figure 8. Note that all C'_j nodes are injection nodes. [7] In this new clause gadget, placing a PMU on any node but a_j results in the observation of at most 3 nodes. Using this simple insight, we can easily argue that more nodes are always observed by placing a PMU on the variable gadget rather than at a clause gadget. Then, we can argue that PMUs are only placed on variable gadgets and finally leverage the argument from Theorem 3.4 to show MaxObserve is NP-Complete for any $\frac{m}{|V|}$. A similar argument can be made for MaxObserve-XV.

[6] The PMU on a T_i or F_i node observes $I_{i1}, I_{i2}, \ldots, I_{ip}$ via O1.

[7] Other modifications exist for the clause gadgets that do not involve solely injection nodes, with similar results.



(a) Modified variable gadget used in FullObserve and MaxObserve, containing additional injection nodes: $I_{i1}, I_{i2}, \ldots I_{ip}$.

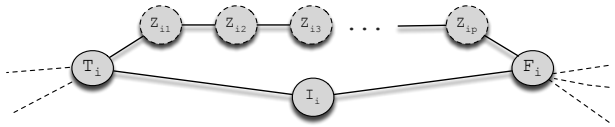


(b) Modified variable gadget used in FullObserve-XV and MaxObserve-XV. Each disconnected subgraph has additional injection nodes: nodes $I^t_{i1}, I^t_{i2}, \ldots I^t_{ip}$ are added to the upper subgraph and nodes $I^b_{i1}, I^b_{i2}, \ldots I^b_{ip}$ are included in the bottom subgraph.

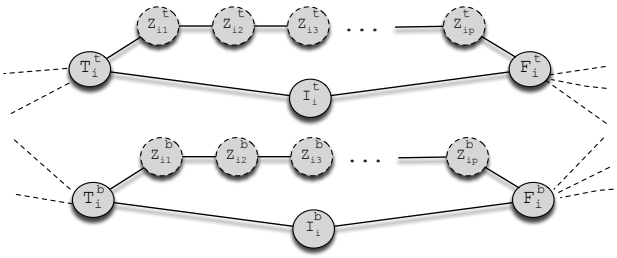
Figure 6: Figures for variable gadget extensions to include more injection nodes described in Section 3.6. The dashed edges indicate connections to clause gadget nodes.

4. APPROXIMATION ALGORITHMS

Because all four placement problems are NPC, we propose greedy approximation algorithms for each problem, which iteratively add a PMU in each step to the node that observes the maximum number of new nodes. We present two such algorithms, one which directly addresses MaxObserve (`greedy`) and the other MaxObserve-XV (`xvgreedy`). `greedy` and `xvgreedy` can easily be used to solve FullObserve and FullObserve-XV, respectively, by selecting the appropriate k value to ensure full observability. We prove these algorithms have polynomial complexity (i.e., they are in $\mathcal{P}$), making them feasible tools for approximating optimal PMU



(a) Modified variable gadget used in FullObserve and MaxObserve, containing additional injection nodes: $Z_{i1}, Z_{i2}, \ldots Z_{ip}$.



(b) Modified variable gadget used in FullObserve-XV and MaxObserve-XV. Each disconnected subgraph has additional injection nodes: the upper subgraph includes nodes $Z^t_{i1}, Z^t_{i2}, \ldots Z^t_{ip}$ and nodes $Z^b_{i1}, Z^b_{i2}, \ldots Z^b_{ip}$ are added in the bottom subgraph.

Figure 7: Figures for variable gadget extensions to include more non-injection nodes described in Section 3.6. The dashed edges indicate connections to clause gadget nodes.

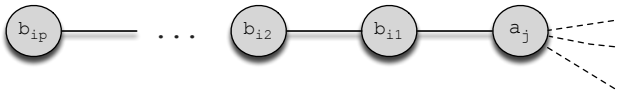


Figure 8: Extended clause gadget, C'_j, used in Section 3.6. All nodes are injection nodes.

placement.

`greedy` **Algorithm**. We start with $\Phi = \emptyset$. At each iteration, we add a PMU to the node that results in the observation of the maximum number of new nodes. The algorithm terminates when all PMUs are placed. [8] The pseudo-code for `greedy` is given in Algorithm 1.

Theorem 4.1. *For input graph $G = (V, E)$ and k PMUs* `greedy` *has $O(dkn^3)$ complexity, where $n = |V|$ and d is the maximum degree node in V.*

Proof. The procedure to determine the number of nodes observed by a candidate PMU placement spans steps $6-18$. [9] First, we apply O1 at each PMU node (steps $6-9$). O1 takes $O(d)$ time to be applied at a single node. Because $|\Phi_G| \leq k$, the total time to apply O1 is $O(dk)$.

Then, we iteratively apply O2 (steps $10-18$), terminating when no new nodes are observed. Like O1, applying O2 at a single node takes $O(d)$ time. In each iteration, if possible we apply O2 at each $v \in (V_Z \cap \Phi^R_G)$ (steps $13-16$). It total, the *loop* spanning steps $10-18$ repeats at most $O(n)$ times. This occurs when only a single new node is observed in each iteration. The *for* loop spanning steps $12-17$ repeats $O(n)$ times. We conclude that O2 evaluation for each set of candidate PMU locations takes $O(dn^2)$ time.

In order to determine the placement of each PMU, we try all possible PMU placements among nodes without a PMU. We place the PMU at the node that observes the maximum number of new nodes. This corresponds to Steps $4-23$, in which the *for* loop iterates $O(n)$ times. Thus the complexity of Steps $4-23$ is $O(dn^3)$.

Finally, the outer most *for* loop (Steps $2-25$) iterates k times: one iteration to determine the greedy placement of each PMU. We conclude that the complexity of `greedy` is $O(dkn^3)$. □

[8] This is the same greedy algorithm proposed by Aazami and Stilp [2].

[9] In this proof, step i refers to the i^{th} line in Algorithm 1.

Algorithm 1 `greedy` with input $G = (V, E)$ and k PMUs

```
1: Φ_G ← ∅
2: for k iterations do
3:   maxObserved ← 0
4:   for each v ∈ (V − Φ_G) do
5:     numObserved ← 0
6:     for each u ∈ (Φ_G ∪ {v}) do
7:       add PMU to u
8:       apply O1 at u and update numObserved
9:     end for
10:    repeat
11:      flag ← False
12:      for each w ∈ (V − (Φ_G ∪ {v})) do
13:        if w ∈ (V_Z ∩ Φ^R_G) and w has 1 unobserved
           neighbor then
14:          apply O2 at w and update numObserved
15:          flag ← True
16:        end if
17:      end for
18:    until flag = False
19:    if numObserved > maxObserved then
20:      greedyNode ← v
21:      maxObserved ← numObserved
22:    end if
23:  end for
24:  Φ_G ← Φ_G ∪ {greedyNode}
25: end for
```

`xvgreedy` **Algorithm**. `xvgreedy` is almost identical to `greedy`, except that PMUs are added in pairs such that the selected pair observe the maximum number of nodes under the condition that the PMU pair satisfy one of the cross-validation rules. We provide the pseudo code for `xvgreedy` in Algorithm 2.

Theorem 4.2. *For input graph $G = (V, E)$ and k PMUs* `xvgreedy` *has $O(kdn^3)$ complexity, where $n = |V|$ and d is the maximum degree node in V.*

Proof. The only difference between `xvgreedy` and `greedy` is that `xvgreedy` only considers pairs of cross-validated nodes. For this reason, step 4 in Algorithm 2 does not appear in Algorithm 1. We can find all pairs of cross-validated nodes in $O(d^2n)$ time. We do so by implementing a breadth-first search at each $v \in (V - \Phi_G)$ but stopping at a depth of 2. This takes $O(d^2)$ time for each node and since $O(n)$ searches are executed, step 4 takes $O(d^2n)$ time.

Because all other parts of Algorithm 1 and Algorithm 2 are nearly identical – Algorithm 2 adds PMUs in pairs while Algorithm 1 adds PMUs one-at-a-time – we are able to directly

Algorithm 2 `xvgreedy` with input $G = (V, E)$ and k PMUs

```
1: Φ_G ← ∅
2: for k iterations do
3:    maxObserved ← 0
4:    C ← all cross-validated node pairs in (V − Φ_G)
5:    for each {v_1, v_2} ∈ C do
6:       numObserved ← 0
7:       for each u ∈ (Φ_G ∪ {v_1, v_2}) do
8:          add PMU to v_1 and v_2
9:          apply O1 at u and update numObserved
10:      end for
11:      repeat
12:         flag ← False
13:         for each w ∈ (V − (Φ_G ∪ {v_1, v_2})) do
14:            if w ∈ (V_Z ∩ Φ_G^R) and w has 1 unobserved
               neighbor then
15:               apply O2 at w and update numObserved
16:               flag ← True
17:            end if
18:         end for
19:      until flag = False
20:      if numObserved > maxObserved then
21:         greedyNodes ← {v_1, v_2}
22:         maxObserved ← numObserved
23:      end if
24:   end for
25:   Φ_G ← Φ_G ∪ greedyNodes
26: end for
```

apply the analysis from Theorem 4.1 in this proof. Therefore, we conclude the complexity of `xvgreedy` is $O(k(d^2n + dn^3)) = O(dkn^3)$. □

5. SIMULATIONS

Topologies. We evaluate our approximation algorithms using simulations over IEEE topologies as well as synthetic ones. As is standard practice in the literature [3, 5, 11, 14], we use IEEE bus systems 14, 30, 57, and 118 [10]. The bus system number indicates the number of nodes in the graph (e.g., bus system 57 has 57 nodes). Synthetic graphs are then generated based on each of these topologies, and are used to quantify the performance of our greedy approximations. We use synthetic topologies in order to establish the statistical significance of our results.

Since observability is determined by the connectivity of the graph, we use the *degree distribution* of IEEE topologies as the template for generating our synthetic graphs. A synthetic topology is generated from a given IEEE graph by randomly "swapping" edges in the IEEE graph. Specifically, we select a random $v \in V$ and then pick a random $u \in \Gamma(v)$. Let u have degree d_u. Next, we select a random $w \notin \Gamma(v)$ with degree $d_w = d_u - 1$. Finally, we remove edge (v, u) and add (v, w), thereby preserving the node degree distribution. We continue this swapping procedure until the original graph and generated graph share *no edges*, and then return the resulting graph.

Evaluation Methods. We are interested in evaluating how close our algorithms are to the optimal PMU placement. Thus, when computationally possible (for a given k) we use brute-force algorithms to iterate over all possible placements of k PMUs in a given graph and select the best PMU placement. When the brute-force algorithm is computationally infeasible, we present only the performance of the greedy algorithm. [11] In what follows, the output of the brute-force algorithm is denoted `optimal`, and when we require cross-validation it is denoted `xvoptimal`.

We present three different simulations in Section 5.1-5.3. In Section 5.1 we consider performance as a function of the number of PMUs, and in Section 5.2 we investigate the performance impact of the number of zero-injection nodes in the network. These two sections use synthetic graphs. We conclude in Section 5.3, where we compare these results to the performance over the actual IEEE graphs.

5.1 Simulation 1: Impact of Number of PMUs

In the first simulation scenario we vary the number of PMUs and determine the number of observed nodes in the synthetic graph. Each data point is generated as follows. For a given number of PMUs, k, we generate a graph, place k PMUs on the graph, and then determine the number of observed nodes. We continue this procedure until $[0.9(\overline{x}), 1.1(\overline{x})]$ – where $\overline{x}$ is the mean number of observed nodes using k PMUs – falls within the 90% confidence interval.

In addition to generating a topology, for each synthetic graph we determined the members of V_I, V_Z. These nodes are specified for the original graphs in the IEEE bus system database. Thus, we randomly map each node in the IEEE graph to a node in the synthetic graph with the same degree, and then match their membership to either V_I or V_Z.

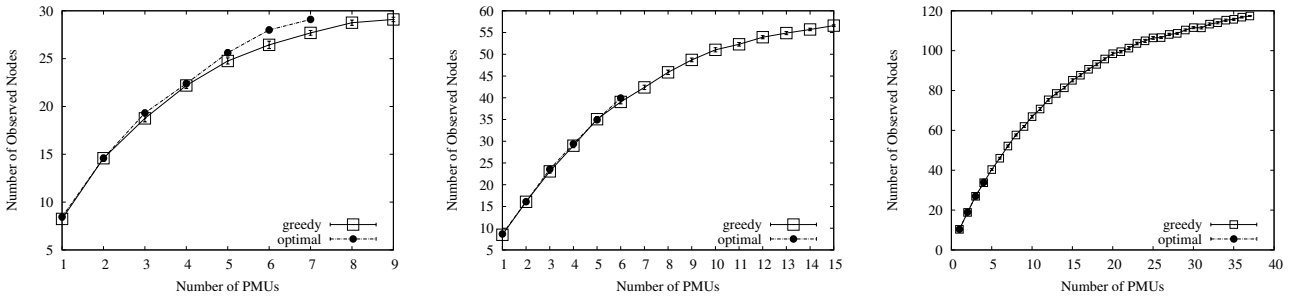
We present here results for solving MaxObserve and MaxObserve-XV. The number of nodes observed given k, using `greedy` and `optimal`, are shown in Figure 9, and Figure 10 shows this number for `xvgreedy` and `xvoptimal`. In both sets of plots we show 90% confidence intervals. We omit results for graphs based on IEEE bus 14 because the same trends are observed.

Our greedy algorithms perform well. On average, `greedy` is within 98.6% of `optimal`, is never below 94% of `optimal`, and in most cases gives the optimal result. Likewise, `xvgreedy` is never less than 94% of `xvoptimal` and on average is within 97% of `xvoptimal`. In about about half the cases `xvgreedy` gives the optimal result. These results suggest that despite the complexity of the problems, a greedy approach can return high-quality results. Note, however, that these statistics do not include performance over large topologies (i.e., IEEE graphs 57, 118) when k is large. It is an open question whether the greedy algorithms used here would do well for larger graphs.

Surprisingly, when comparing our results with and without the cross-validation requirement, we find that the cross-validation constraints have little effect on the number of observed nodes for the same k. Our experiments show that on average `xvoptimal` observed only 5% fewer nodes than `optimal`. Similarly, on average `xvgreedy` observes 5.7% fewer nodes than `greedy`. This suggests that the cost of imposing the cross-validation requirement is low, with the clear gain of ensuring PMU correctness across the network.

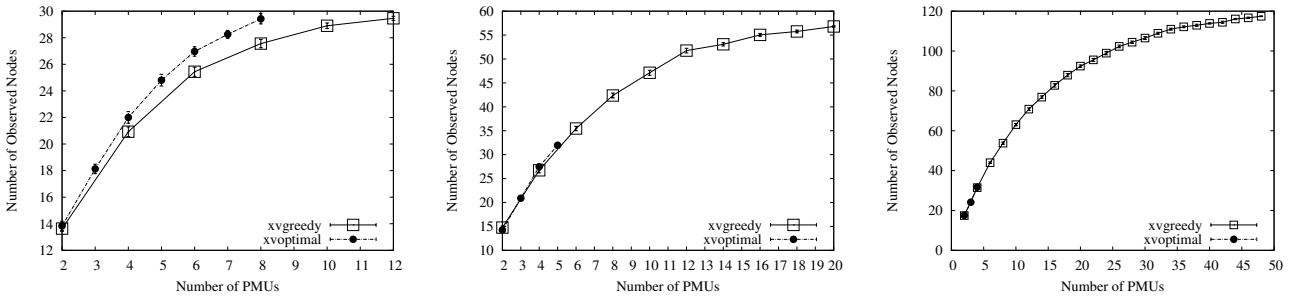
[10] http://www.ee.washington.edu/research/pstca/

[11] Because of the computational cost of the brute-force algorithm for larger topologies (e.g., IEEE bus system 300 and other more complicated systems), we have no reference in which to measure the effectiveness of our greedy approximations for larger scale systems. Therefore, we do not present results for topologies larger than IEEE bus 118.



(a) Graphs based on IEEE bus 30 (b) Graphs based on IEEE bus 57 (c) Graphs based on IEEE bus 118

Figure 9: Mean number of observed nodes over synthetic graphs – using `greedy` and `optimal` – when varying number of PMUs. The 90% confidence interval is shown.



(a) Graphs based on IEEE bus 30 (b) Graphs based on IEEE bus 57 (c) Graphs based on IEEE bus 118

Figure 10: Over synthetic graphs, mean number of observed nodes – using `xvgreedy` and `xvoptimal` – as a function of number of PMUs. The 90% confidence interval is shown.

5.2 Simulation 2: Impact of Number of Zero-Injection Nodes

Next, we examine the impact of the number of zero-injection nodes ($|V_Z|$) on algorithm performance. For each synthetic graph, we run our algorithms for increasing values of $|V_Z|$ and determine the minimum number of PMUs needed to observe all nodes in the graph (k^*). For each $z := |V_Z|$, we select z nodes uniformly at random to be zero-injection, and the rest are in V_I. Because we compute k^* here, we solve FullObserve and FullObserve-XV, rather than MaxObserve and MaxObserve-XV as in Simulation 1.

We generate each data point using a similar procedure to the one described in Section 5.1. For each z, we generate a graph and determine k^*. We then compute $\overline{k^*}$, the mean value of k^* using $|V_Z| = z$. We continue this procedure until $[0.9(\overline{k^*}), 1.1(\overline{k^*})]$ falls within the 90% confidence interval.

Figure 11(a) shows the simulation results for solving FullObserve and FullObserve-XV on synthetic graphs modeled by IEEE bus 57. Results for other topologies considered here (i.e., 14, 30 and 118) followed the same trend and are thus omitted. Due to the exponential running time of `optimal` and `xvoptimal`, we present here only results of our greedy algorithms.

As expected, increasing the number of zero-injection nodes, for both `greedy` and `xvgreedy`, reduces the number of PMUs required for full observability. More zero-injection nodes allow O2 to be applied more frequently (Figure 11(b)), thereby increasing the number of observed nodes without using more PMUs. In fact, we found the relationship between $|V_Z|$ to the greedy estimate of k^* to be linear.

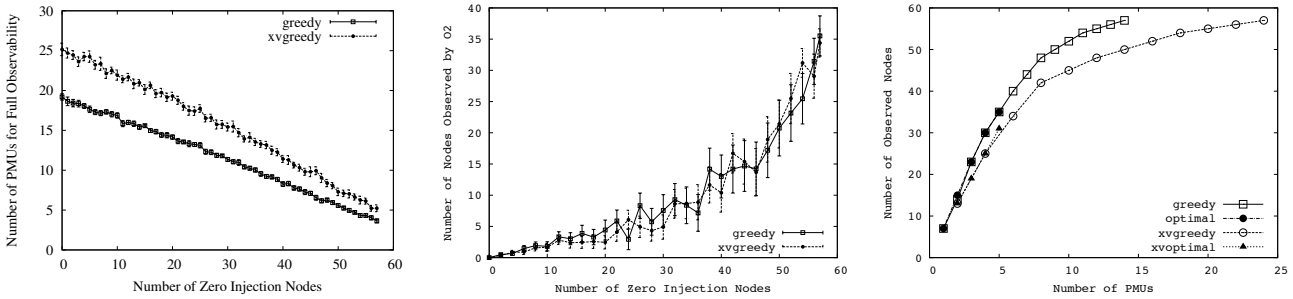
The gap in k^* between `greedy` and `xvgreedy` decreases as z grows. `greedy` and `xvgreedy` observe a similar number of nodes via O2 across all z values: the mean absolute difference in the number of nodes observed by O2 between the two algorithms is only 1.66 nodes (equivalently, less than 3% of observed nodes). Thus, as z grows the number of nodes observed by O2 accounts for an increasing proportion of all observed nodes (Figure 11(b)), causing the gap between `greedy` and `xvgreedy` to shrink.

5.3 Simulation 3: Synthetic vs Actual IEEE Graphs

In this section, we compare our results with the performance over the original IEEE systems. We assign nodes to V_Z and V_I as specified in the IEEE database files. Our results indicate that the trends we observed over the synthetic graphs apply as well to real topologies.

Figure 11(c) shows the number of observed nodes for the `greedy`, `xvgreedy`, `optimal`, and `xvoptimal` algorithms for IEEE bus system 57. `greedy` and `xvgreedy` observe nearly as many nodes as the corresponding optimal solution. In many cases, greedy yields the optimal placement. Similarly, as with the synthetic graphs, the number of PMUs required to observe all nodes decreases linearly as $|V_Z|$ increases. [12]

[12]The same trends are observed using IEEE bus systems 14, 30, and 118.



(a) Simulation 2 (b) Simulation 2 (c) Simulation 3

Figure 11: Figures (a) and (b) correspond to Simulation 2 using synthetic graphs based on IEEE bus 57. The 90% confidence interval is shown in Figure (a) and (b). Figure (a) considers the number of PMUs needed for full observability as a function of $|V_Z|$ and Figure (b) shows the number of nodes observed by O2 for different $|V_Z|$ values. Figure (c) displays the number of observed nodes as a function of the number of PMUs, using IEEE bus 57.

	greedy	xvgreedy	optimal	xvoptimal
Simulation 1	4%	4.6%	6%	7.6%
Simulation 2	9.1%	16.1%	N/A	N/A

Table 1: Mean absolute difference between the computed values from synthetic graphs and IEEE graphs, normalized by the result for the synthetic graph.

Finally, we consider whether node degree distribution is an appropriate feature for generating topologies similar to their IEEE counterparts. To do so, we compare the actual number of observed nodes for synthetic graphs to those over IEEE graphs. Specifically, we take the mean absolute difference between these two values, and normalized by the result for the synthetic graph. For example, let n_k be the mean number of observed nodes using `greedy` over all synthetic graphs with input k, and let $n_{G,k}$ be the output of `greedy` for IEEE graph G and k. We compute $n_{d,k} = (|n_k - n_{G,k}|)/n_k$. Finally, we calculate the mean over all $n_{d,k}$. This process is done for each algorithm we evaluate. The resulting statistics can be found in Table 1. The small average difference between the synthetic and actual IEEE topologies suggests that the node degree distribution of the IEEE graph is an effective feature for generating similar synthetic graphs.

6. RELATED WORK

FullObserve is well-studied [3, 4, 9, 11, 14]. Haynes et al. [9] and Brueni and Heath [4] both prove FullObserve is NPC. However, their proofs make the unrealistic assumption that all nodes are zero-injection. We drop this assumption and thereby generalize their NPC results for FullObserve. Additionally, we leverage the proof technique from Brueni and Heath [4] in all four of our NPC proofs, although our proofs differ considerably in their details.

In the power systems literature, Xu and Abur [14, 15] use integer programming to solve FullObserve, while Baldwin et al. [3] and Mili et al. [11] use simulated annealing to solve the same problem. All of these works allow nodes to be either zero-injection or non-zero-injection. However, these papers make no mention that FullObserve is NPC, i.e., they do not characterize the fundamental complexity of the problem.

Aazami and Stilp [2] investigate approximation algorithms for FullObserve. They derive a hardness approximation threshold of $2^{\log^{1-\epsilon} n}$. Also they prove that in the worst case, `greedy` from Section 4 does no better $\Theta(n)$ of the optimal solution. However, this approximation ratio assumes that all nodes are zero-injection.

Chen and Abur [5] and Vanfretti et al. [13] both study the problem of bad PMU data. Chen and Abur [5] formulate their problem differently than FullObserve-XV and MaxObserve-XV. They consider fully observed graphs and add PMUs to the system to make all existing PMU measurements non-critical (a critical measurement is one in which the removal of a PMU makes the system no longer fully observable). Vanfretti et al. [13] define the cross-validation rules used in this paper. They also derive a lower bound on the number of PMUs needed to ensure all PMUs are cross-validated and the system is fully observable.

7. CONCLUSIONS AND FUTURE WORK

In this work, we formulated four PMU placement problems and proved that each one is NPC. Consequently, future work should focus on developing approximation algorithms for these problems. As a first step, we presented two simple greedy algorithms: `xvgreedy` which considers cross-validation and `greedy` which does not. Both algorithms iteratively add PMUs to the node which observes the maximum of number of nodes.

Using simulations, we found that our greedy algorithms consistently reached close-to-optimal performance. Our simulations also showed that the number of PMUs needed to observe all graph nodes decreases linearly as the number of zero-injection nodes increase. Finally, we found that cross-validation had a limited effect on observability: for a fixed number of PMUs, `xvgreedy` and `xvoptimal` observed only 5% fewer nodes than `greedy` and `optimal`, respectively. As a result, we believe imposing the cross-validation requirement on PMU placements is advised, as the benefits they provide come at a low marginal cost.

There are several topics for future work. The success of

the greedy algorithms suggests that bus systems have special topological characteristics, and we plan to investigate their properties. Additionally, we intend to implement the integer programming approach proposed by Xu and Abur [14] to solve FullObserve. This would provide valuable data points to measure the relative performance of `greedy`.

8. ACKNOWLEDGMENTS

This work is supported by NSF grant CNS-1143655. Also, we thank Luigi Vanfretti, David Bertagnolli, and Dan Brancaccio for helpful discussions about power systems, PMUs, and PMU errors.

9. REFERENCES

[1] Synchro-Phasor Data Validation. Technical report, North-American Synchro-Phasor Initative, October 2010.

[2] A. Aazami and M.D. Stilp. Approximation Algorithms and Hardness for Domination with Propagation. *CoRR*, abs/0710.2139, 2007.

[3] T.L. Baldwin, L. Mili, Jr. Boisen, M.B., and R. Adapa. Power System Observability with Minimal Phasor Measurement Placement. *Power Systems, IEEE Transactions on*, 8(2):707 –715, May 1993.

[4] D. J. Brueni and L. S. Heath. The PMU Placement Problem. *SIAM Journal on Discrete Mathematics*, 19(3):744–761, 2005.

[5] J. Chen and A. Abur. Placement of PMUs to Enable Bad Data Detection in State Estimation. *Power Systems, IEEE Transactions on*, 21(4):1608 –1615, 2006.

[6] J. De La Ree, V. Centeno, J.S. Thorp, and A.G. Phadke. Synchronized Phasor Measurement Applications in Power Systems. *Smart Grid, IEEE Transactions on*, 1(1):20 –27, 2010.

[7] M.R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[8] D. Gyllstrom, J. Kurose, and E. Rosensweig. Max Observability PMU Placement with Cross-Validation. Technical Report UM-CS-2011-014.

[9] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning. Domination in Graphs Applied to Electric Power Networks. *SIAM J. Discret. Math.*, 15:519–529, April 2002.

[10] D. Lichtenstein. Planar Formulae and Their Uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[11] L. Mili, T. Baldwin, and R. Adapa. Phasor Measurement Placement for Voltage Stability Analysis of Power Systems. In *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*, pages 3033 –3038 vol.6, December 1990.

[12] L. Vanfretti. *Phasor Measurement-Based State-Estimation of Electrical Power Systems and Linearized Analysis of Power System Network Oscillations*. PhD thesis, Rensselaer Polytechnic Institute, December 2009.

[13] L. Vanfretti, J. H. Chow, S. Sarawgi, and B. (B.). Fardanesh. A Phasor-Data-Based State Estimator Incorporating Phase Bias Correction. *Power Systems, IEEE Transactions on*, 26(1):111–119, Feb 2011.

[14] B. Xu and A. Abur. Observability Analysis and Measurement Placement for Systems with PMUs. In *Proceedings of 2004 IEEE PES Conference and Exposition, vol.2*, pages 943–946, 2004.

[15] B. Xu and A. Abur. Optimal Placement of Phasor Measurement Units for State Estimation. Technical Report PSERC Publication 05-58, October 2005.

[16] J. Zhang, G. Welch, and G. Bishop. Observability and Estimation Uncertainty Analysis for PMU Placement Alternatives. In *North American Power Symposium (NAPS), 2010*, pages 1 –8, 2010.