tion 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 number of 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 chapter is organized as follows. In Section 2.2 we introduce our modeling assumptions, notation, and observability and cross-validation rules. In Section 2.3 we formulate and prove the complexity of our four PMU placement problems. Section 2.4 presents the approximation algorithms for each problem and Section 2.5 considers our simulation-based evaluation. We conclude with a review of related work (Section 2.6) and concluding remarks (Section 2.7).

2.2 Preliminaries

In this section we introduce notation and underlying assumptions (Section 2.2.1), and define our observability (Section 2.2.2) and cross-validation (Section 2.2.3) rules.

2.2.1 Assumptions, Notation, and Terminology

We model a power grid as an undirected graph $G = (V, E)$. Each $v \in V$ represents a bus. $V = V_Z \cup V_I$, 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 [47]. All other buses are non-zero-injection, which we refer to as injection buses. Each $(u, v) \in E$ is a transmission line connecting buses u and v.

Consistent with the conventions in [7, 12, 15, 34, 44, 45], we assume: PMUs can only be placed on buses and a PMU on a bus measures the voltage phasor at the bus and the current phasor of all transmission lines connected to it. For convenience, we refer to any bus with a PMU as a *PMU node.*

For $v \in V$ define let $\Gamma(v)$ be the set of v's neighbors in G. 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.

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

2.2.2 Observability Rules

We use the simplified observability rules stated by Brueni and Heath [12]:

1. **Observability Rule 1 (O1).** *If node v is a PMU node, then $v \cup \Gamma(v)$ is observed.*

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 $v \cup \Gamma(v)$ is observed.*

Since O2 only applies with zero-injection nodes, the number of zero-injection nodes can greatly affect system observability.

2.2.3 Cross-Validation Rules

Cross-validation formalizes the intuitive notion of placing PMUs "near" each other to allow for measurement error detection. For convenience, we say a PMU is cross-validated even though it is actually the PMU data at a node that is cross-validated. A PMU is *cross-validated* if one of the rules below is satisfied [42]:

1. **Cross-Validation Rule 1 (XV1).** *If two PMU nodes are adjacent, then the PMUs cross-validate each other.*

2. **Cross-Validation Rule 2 (XV2).** *If two PMU nodes have a common neighbor, then the PMUs cross-validate each other.*

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) [41]. When the PMUs are two hops from each other (i.e., have a common neighbor), 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 [41].

2.3 Four NP-Complete PMU Placement Problems

In this section we define four PMU placement problems (FULLOBSERVE, MAXOBSERVE, FULLOBSERVE-XV, and MAXOBSERVE-XV) and prove the NP-Completeness details can be found in [illegible] arXiv TR and e-Energy paper

~~of each.~~ FullObserve-XV and MaxObserve-XV both consider measurement error detection, while FullObserve and MaxObserve do not. In effort to keep this proposal document relatively short, we omit the actual NP-Completeness proofs and instead present proof sketches; We begin this section with a high-level description of the proof strategy we use to prove each problem is NP-Complete (Section 2.3.1). In Section 2.3.2, we state our four PMU placement problems and outline our NP-Completeness proofs for each.

2.3.1 Overview of NPC Proof Strategy

In this section, we outline the proof strategy we use in each of our NP-Completeness proofs. Our proofs follow a similar structure to those proposed by Brueni and Heath [12]. 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_i and c_j nodes, and so the graph is bipartite. P3SAT is a 3-SAT formula such that $G(\phi)$ is planar [30]. For example, P3SAT formula

$$\begin{aligned} \varphi \ = \ & (\overline{v_1} \vee v_2 \vee v_3) \wedge (\overline{v_1} \vee \overline{v_4} \vee v_5) \wedge (\overline{v_2} \vee \overline{v_3} \vee \overline{v_5}) \\ & \wedge (v_3 \vee \overline{v_4}) \wedge (\overline{v_3} \vee v_4 \vee \overline{v_5}) \qquad (2.1) \end{aligned}$$

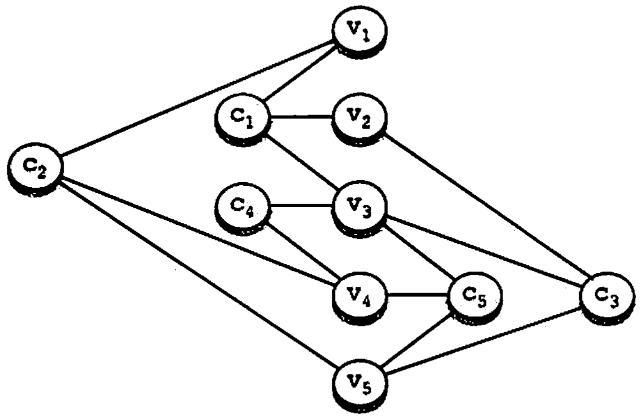
has graph $G(\varphi)$ shown in Figure 2.1(a). Discovering a satisfying assignment for P3SAT is an NPC problem, and so it can be used in a reduction to prove the complexity of the problems we address here.

Following the approach in [12], 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" 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.

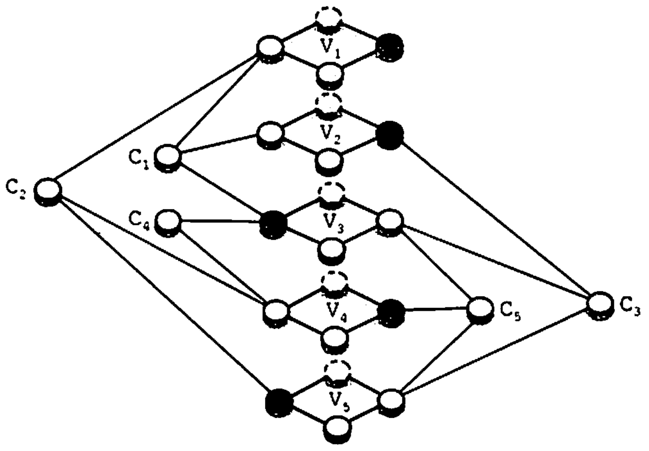
While we assume $G(\phi)$ is planar, we make no such claim regarding $H(\phi)$, though in practice all graphs used in our proofs are indeed planar. The proof of NPC rests on the fact that solving the underlying ϕ formula is NPC.

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.

While the structure of our proofs is adapted from [12], the variable and clause gadgets we use to correspond to the P3SAT formula are novel, thus leading to a different set of proofs. Our work here demonstrates how the work in [12] can be extended, using new variable and clause gadgets, to address a wide array of PMU placement problems.



(a) $G(\varphi)$ formed from φ in Equation (2.1).



(b) Graph formed from φ formula in Theorem 2.1 proof.

Figure 2.1. The figure in (a) shows $G(\varphi) = (V(\varphi), E(\varphi))$ using example formula, φ, from Equation (2.1). (b) shows the new graph formed by replacing each variable node in $G(\varphi)$ – as specified by the Theorem 2.1 proof – with the Figure 2.2(a) variable gadget.

2.3.2 Problem Statements and NPC Proof Sketches

Here we briefly define each of our four PMU placement problems: FULLOBSERVE, MAXOBSERVE, MAXOBSERVE-XV, and FULLOBSERVE-XV. Then, we provide proof sketches (that follow the proof strategy outlined in the previous section) demonstrating that each algorithm is NP-Complete.

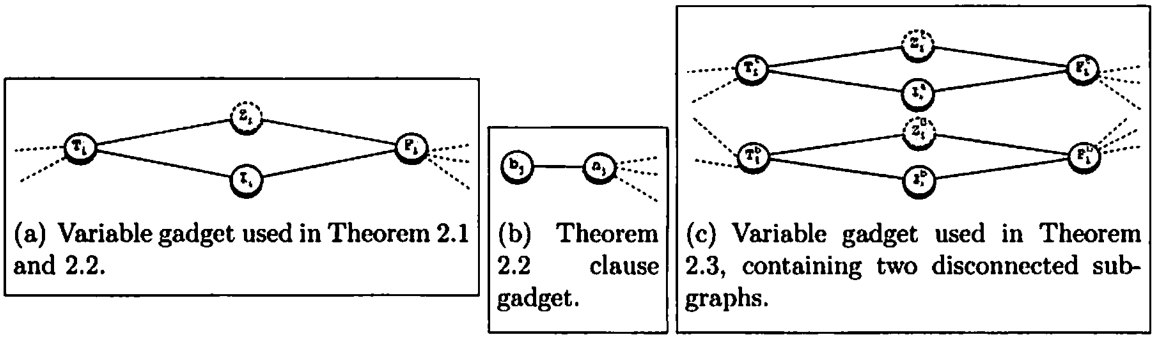
FullObserve Decision Problem:

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 2.1. FULLOBSERVE *is NP-Complete.*

Proof Sketch: We introduce a problem-specific variable gadget shown in Figure 2.2(a) and a single node as the clause 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.


(a) Variable gadget used in Theorem 2.1 and 2.2.

(b) Theorem 2.2 clause gadget.

(c) Variable gadget used in Theorem 2.3, containing two disconnected subgraphs.

Figure 2.2. Gadgets used in Theorem 2.1 - 2.4. Z_i in Figure 2.2(a), Z_i^t in Figure 2.2(c), and Z_i^b in Figure 2.2(c) are the only zero-injection nodes. The dashed edges in Figure 2.2(a) and Figure 2.2(c) are connections to clause gadgets. Likewise, the dashed edges in Figure (b) are connections to variable gadgets. In Figure 2.2(c), superscript, t, denotes nodes in the upper subgraph and superscript, b, indexes nodes in the lower subgraph.

Consider the example P3SAT formula from Equation 2.1, φ, and its corresponding bipartite graph, $G(\varphi)$, shown in Figure 2.1(a). Our proof procedure replaces each variable node in $G(\varphi)$ with the variable gadget shown in Figure 2.2(a), yielding the graph shown in Figure 2.1(b). Nodes with a dashed border are zero-injection nodes. φ is satisfied when literals $\overline{v_1}, \overline{v_2}, v_3, \overline{v_4}$, and $\overline{v_5}$ are all True, yielding a PMU placement with PMUs at each dark shaded node in Figure 2.1(b).

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 $m \leq |\Phi_G^R| < |V|$?

Theorem 2.2. MAXOBSERVE *is NP-Complete.*

Proof Sketch: First, we construct problem-specific gadgets for variables (Figure 2.2(a)) and clauses (Figure 2.2(b)). 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 2.1.

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 2.3. FULLOBSERVE-XV *is NP-Complete.*

Proof Sketch: We show FULLOBSERVE-XV is NP-hard by reducing from P3SAT. We create a single-node gadget for clauses (as we did with FULLOBSERVE) and the gadget shown in Figure 2.2(c) 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 use the PMU placement to derive a satisfying P3SAT truth assignment.

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 2.4. MAXOBSERVE-XV *is NP-Complete.*

Proof Sketch: Our proof is a combination of the NP-Completeness proofs for MAXOBSERVE and FULLOBSERVE-XV.

2.4 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~~ that 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.

`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. [1]

`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.

2.5 Simulation Study

Topologies. We evaluate our approximation algorithms with simulations over synthetic topologies generated using real portions of the North American electric power grid (i.e., IEEE bus systems 14, 30, 57, and 118) as templates [2]. The bus system number indicates the number of nodes in the graph (e.g., bus system 57 has 57 nodes). It is standard practice in the literature to only use single IEEE bus systems [7, 15, 34, 44]. We follow this precedent but do not present these results because they are consistent with the trends found using synthetic topologies. Instead, we focus on synthetic topologies because, unlike simulations using single IEEE bus systems, we can establish the statistical significance of the performance of our greedy approximations.

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

[1]The same greedy algorithm is proposed by Aazami and Stilp [3].

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

"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. In what follows, the output of the brute-force algorithm is denoted `optimal`, and when we require cross-validation it is denoted `xvoptimal`.

Simulation Results. 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.

Due to space constraints, we only show plots for solving MAXOBSERVE and MAXOBSERVE-XV using synthetic graphs based on IEEE bus 57. The number of nodes observed given k, using `greedy` and `optimal`, are shown in Figure 2.3(a), and

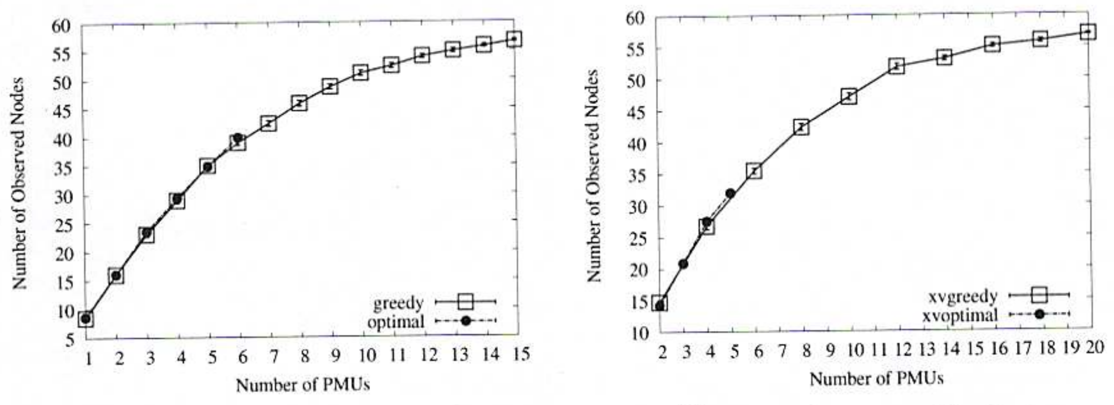
Figure 2.3(b) shows this number for `xvgreedy` and `xvoptimal`. Both plots include the 90% confidence intervals. ~~Note that~~ R results for synthetic graphs generated using IEEE bus 14, 30, and 118 yield the same trends.

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 when k is large. It is an open question whether `greedy` and `xvgreedy` would do well for large k.

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.

2.6 Related Work

FullObserve is well-studied [7, 12, 26, 34, 44]. Haynes et al. [26] and Brueni and Heath [12] 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 [12] in all four of our NPC proofs, although our proofs differ considerably in their details.



(a) `greedy` vs `optimal` (b) `xvgreedy` vs `xvoptimal`

Figure 2.3. Mean number of observed nodes over synthetic graphs based on IEEE bus 57 when varying number of PMUs. The 90% confidence interval is shown.

In the power systems literature, Xu and Abur [44, 45] use integer programming to solve FULLOBSERVE, while Baldwin et al. [7] and Mili et al. [34] 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 [3] 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 2.4 does no better $\Theta(n)$ of the optimal solution. However, this approximation ratio assumes that all nodes are zero-injection.

Chen and Abur [15] and Vanfretti et al. [42] both study the problem of bad PMU data. Chen and Abur [15] 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. [42] define the cross-validation rules used in this ~~paper~~ chapter.

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.

2.7 Conclusions and Future Work

In this chapter, 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. We also 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 [44] to solve FULLOBSERVE. This would provide valuable data points to measure the relative performance of `greedy`.

as part of your thesis?