PHASOR MEASUREMENT UNIT PLACEMENTS FOR COMPLETE OBSERVABILITY USING LINEAR-TIME, QUADRATIC-TIME, AND SUBQUADRATIC-TIME HEURISTICS

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Oluwasijibomi Saula

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Electrical and Computer Engineering

April 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

PHASOR MEASUREMENT UNIT PLACEMENTS FOR COMPLETE OBSERVABILITY

USING LINEAR-TIME, QUADRATIC-TIME AND SUBQUADRATIC-TIME HEURISTICS

By

Oluwasijibomi Saula

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Saula, Oluwasijibomi, M.S., Department of Electrical and Computer Engineering, College of Engineering and Architecture, North Dakota State University, April 2010. Phasor Measurement Unit Placements for Complete Observability using Linear-Time, Quadratic-Time, and Subquadratic-Time Heuristics. Major Professor: Dr. Samee Ullah Khan.

A phasor measurement unit (PMU) is considered to have the potential to improve the efficiency of electric power systems by monitoring, control, and protection. Through measurements of all bus voltages, incoming and outgoing currents, and by subsequent calculation of all phase angles, employing PMUs on every substation in a power system will allow complete observation of a power system. However, having a PMU on every substation may not be economically feasible. Therefore, methodologies must be devised that can monitor a system with the minimum possible number of PMUs. In this paper, we propose six graph theoretical PMU placement heuristics. The proposed heuristics overcome the previous approaches in terms of scalability and execution time. The proposed heuristics are thoroughly compared and benchmarked using standard IEEE bus networks ranging from 14 to 300 buses, and a 2,383 bus system.

ACKNOWLEDGMENTS

I would like to thank Dr. Samee Ullah Khan for his assistance, advice and help in writing this thesis. I also thank Dr. Rajendra Katti, Dr. Rajesh Kavaserri, and Dr. Warren Shreve for their contributions to this thesis and my graduate education. Greatly appreciated are the precursory efforts of Garrett Kropp, Brady Brodsho, Joshua Adamek on the thesis subject; I am sincerely thankful to Garrett Kropp for his persistent timely assistance.

To my family, the Saulas, please accept my gratitude for all the support, encouragement, advice, and prayers all through the years leading to this thesis submission. I am especially appreciative of Apostle Mr. Shields and Pastor Mrs. Shields, for their kind and gentle directions, and the encouraging words of advice and wisdom.

I am most grateful to my Counselor and Teacher, my Lord and Savior Jesus Christ, who since my birth has led me thus far. Now and forever, I am thankful to You.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

A Phasor Measurement Unit (PMU) is a measurement device which utilizes a global positioning system (GPS) and allows for the acquisition of synchronized phasor measurements, specifically voltage and current phasor measurements. In addition to measuring voltage and current magnitudes, the phase angle can also be calculated directly, and phasor measurements can be presented in a continuous format. As PMUs are dynamic state estimators, these non-static estimators allow monitoring of a power system by calculating state estimates using continuous streams of voltage and current phasor measurements. By virtue of these continuous streams of phasor measurements, the synchronization of PMUs has become an advantage over traditional analog meters. In addition, PMUs are attractive for their ability to improve bad data detection [22], provide better accuracy for iterative state estimation algorithms [4], stability control [23], and disturbance monitoring [24]. PMUs are also favored in system protection schemes while also playing an important role in post-mortem analysis of a power system. With installed PMUs providing synchronized phasor measurements, the standards of power system monitoring, control, and protection of a power system is enhanced [25].

Synchronization in a power system is vital and can be accomplished by using state estimators that reside at a comptroller. The comptroller monitors all measurements and other information received from substations and meters. By continual monitoring, state estimators are able to provide approximate voltage, current, and power measurements of a power system. Estimators may also detect errors in data and make corrections where necessary. One disadvantage of using traditional analog meters is the inherent time dependency [4]. This demerit is visible when disturbances, such as power fluctuations, occur. Traditional analog meters are not guaranteed to trigger at the time of disturbance and could result in defective monitoring of power

systems. Thankfully, the proper application of PMUs prevents such measurement omissions from taking place.

By utilizing the current state of a power system with sufficient but necessary knowledge on the measurement set and their distribution, the power system becomes *observable* (measurable) [4]. PMUs installed on all of the nodes can provide complete observability but PMUs on every substation may not be economically feasible [3]. When a PMU is deployed on a substation, that substation is termed a *directly observable substation.* The substation is termed a *calculated substation* when such substation does not have a PMU installed but is observable by other PMUs installed on neighboring substations. A system is said to be *completely observable* when all of the substations are either directly observed or calculated. If any of the substations are not observed, then that system is defined as *unobservable.* [4]. For instance, consider a power system that is composed of seven substations as described in Fig. 1.

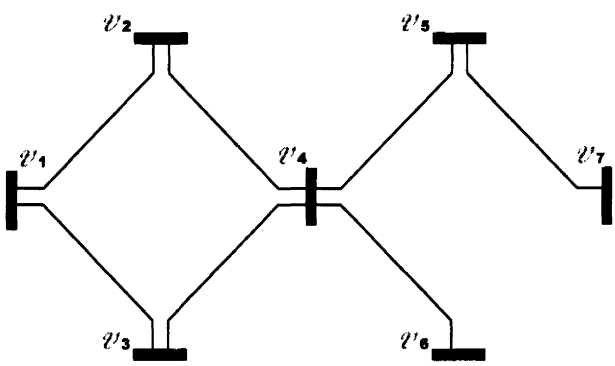


Figure 1: An example power system.

Assume that PMUs are installed on substations v_1, v_4, and v_7. We can differentiate directly observable substations from calculated substations:

1. The PMU on v_1 makes the substations v_2 and v_3 visible.

2. The PMU on v_4 make the substations v_2, v_3, v_5, and v_6 visible.

3. The PMU on v_7 makes the substation v_5 visible.

Therefore, v_1, v_4, and v_7 are designated as directly observable substations and the remaining substations (v_2, v_3, v_5, and v_6) as calculated substations.

A system is termed as completely observable if and only if all of the substations are visible. Therefore, by choosing an optimum position to install PMUs, we can achieve complete observability along with the reduced deployment cost. The main contributions of this paper are as follows:

1. We formulate a graph theoretical PMU placement problem. This formulation is considerably different and comprehensible than the previous formulations.

2. We design and analyze a wide range of PMU placement heuristics, such as edge selection, greedy, and A-star. These heuristics are proven to be linear and sub-quadratic in scalability.

3. Because of the wide range of selection, the heuristics proposed in this paper are easily comprehensible and extend able to the well-known set covering problem [9], vertex covering problem [9], and quadratic assignment problem [12].

4. The execution time of the proposed heuristics also is computationally acceptable.

5. To standardize our results, we benchmark our proposed heuristics on the publicly available IEEE bus networks (14 bus system, 30 bus system, 57 bus system, 118 bus system, and 300 bus system) [14]. Additionally, the proposed heuristics are benchmarked on a 2383 bus network that was generated at North Dakota State University by concatenation of the aforementioned bus systems.

The remainder of this paper is organized as follows. In Section 2, we review previous work. The problem formulation will be described in Section 3. Then in Section 4 and Section 5, we present our proposed heuristics and their simulation results, respectively. Finally, we conclude the paper in Section 6.

CHAPTER 2. RELATED WORK

For maximal benefits of PMUs in a power system, site selection of PMUs to be installed in a power system must be properly addressed. With proper PMU placements, the observability of a power system is improved. Power system observability allows a state estimator to determine the unique state solution of a power system given the network's measurements and topology. For optimal state estimation of a power system, full network observability is usually preferred. An existent constraint on PMU placement algorithms is the ensuing costs of placed PMUs. Since the cost of placed PMUs is directly proportional to the number of PMUs placed, PMU locations chosen judiciously will aptly minimize the number of PMUs placed and the resulting cost. Given a set of measurements, we are assured of full network observability (coverage) when at least one measurement spanning tree of full rank can be formed [15]. With the aim of full network observability and minimal number of employed PMUs, optimal placements of PMUs is necessary.

It should be noted that the problem of minimizing the monetary cost of PMUs employed in a power system is distinct to the problem of minimizing the number of PMUs assigned for full network coverage. The problem of minimizing the monetary cost of PMUs is attempted by the authors in [1]. In [1], the authors attempt to solve the basic problem statement considering the presence of conventional measuring units in power system; the proposed solution method is named Augmented Bus Merging (ABM). A solution is also proposed to the problem of minimizing the cost of PMUs where loss of single or multiple PMUs is expected; the proposed solution method is named Local Redundancy (LR).

In this paper, we focus on the problem of minimizing the number of PMUs required for full network coverage (full observability) of a power system; we also refer to this problem as the Optimal PMU Placement problem (OPP). The OPP problem

descends from classical graph theoretical optimization problems such as set covering, vertex covering, and quadratic assignment, and has been attempted by various works.

An approach presented in [5] applies simulated annealing (SA) to the OPP problem, however, SA is a non optimal technique and suffers from heavy calculation burden in reaching near optimal solutions. In [15], a modification was made to SA in an attempt to reduce SA's computational burden. Another method, Tabu Search (TS) was presented to reduce search space drastically. Unfortunately, TS produced results that were non-optimal even in small studies [16]. In [17], the OPP problem is approached using sorting genetic algorithm (NSGA). Complex as NSGA is, its applicability is restricted by the size of the problem. The authors of [1] remark on the time consumed by various approaches discussed above. The authors of [1] therefore present a time efficient technique. The strategy presented is called Immunity Genetic Algorithm (IGA). IGA is an amalgamation of the genetic algorithm (GA) and immune algorithm (IA). The GA and IA are algorithms developed in the fashion of natural biological processes and implemented using statistical methods. The added advantage of IGA is its utilization of local information and avoidance of repetitive or fruitless tasks in crossovers and mutations.

In [6], the authors propose a solution to the OPP problem by utilizing a binary search algorithm. The authors of [6] also provide a means of retaining full observability in the event of a single PMU outage in a power system. Benchmarking the PMU placement method in [6] was accomplished by running the binary search algorithm on the IEEE 14-bus, IEEE 24-bus, IEEE 30-bus, and the New England 39-bus test systems. In most of these case studies, the binary search algorithm in [6] provided optimal solutions, some of which were improvements over solutions presented by the authors in [2]. In regards to execution time, the binary search algorithm in [1] was applied to a 298-bus test system. An optimal solution is produced. Nevertheless, an

execution time of 91 min is incurred on a single Intel Pentium 4 3.6-GHz CPU with 1 GB RAM.

For a power system to be observable, authors in [5] reported that PMUs need to be installed at 1/5 to 1/3 of the number of system buses. In [19], the authors define incomplete observability as a PMU placement situation in which a specified number of PMUs and corresponding locations are insufficient in obtaining full network observability. A novel concept called "depth-of-unobservability" is also introduced to limit the distance between observed and unobserved buses. As such, for a depth-of-one unobservability, there is one unobserved bus linked to calculated buses; for a depth-of-two unobservability, there are two unobserved buses connected to calculated buses. A tree search placement technique is employed to execute PMU placement in accordance to desired depth-of-unobservability. It is noted that a decreasing number of PMUs is needed to maintain higher depths-of-unobservability.

In [2], integer programming is applied to the OPP problem, with consideration of convention measurement units already existent in the power system. According to authors of [21], some draw backs of the solution proposed in [2] exist: computed results are error prone due to numerical roundoff errors. In [18], the author presents a generalized integer linear programming algorithm. There is a consideration for power systems with conventional measurements and for power systems without conventional measurements. The author of [1] also formulates the problem to accommodate the novel concept of depth-of-unobservability introduced in [19]. Specifically, the authors of [18] solve the OPP problem for depth-of-one-unobservability and depth-of-two-unobservability. The integer linear programming approach developed in [18] was based on Matlab's binary integer programming.

An integer programming strategy is developed by authors in [2] for the OPP problem. In [20], it is discovered that the strategy in [2] becomes non-linear when

power injection measurements or conventional power flow exist. In [17], genetic algorithm (GA), a PMU placement algorithm utilizing the Pareto-optimal paradigm is presented. GA solves the OPP problem by searching the solution space for a set of Pareto-optimal solutions. The solutions in the Pareto-optimal set are equally optimal when all PMU placement objectives are considered. The advantage of GA is the provision of the complete Pareto-optimal front instead of a single point solution. However, the computational complexity of GA limits its application to OPP problems of large system buses. This complexity in computation is attributed to correction of infeasible solutions by GA.

The authors of [21] present a method of optimal PMU placement that factors the outage of single branches and single PMU losses (contingency conditions). As expected, the placement strategy resulted in greater PMU placement numbers than that required for full network observability. In comparison with [2], the required number of placement sites computed by the proposed method is less. Other improvements over [2] are better conditional numbers, more accurate state estimations, complete network observability in contingency conditions.

Authors of Ref [4] propose a matrix reduction technique that transforms a real power system into arrays of buses, branches, and injections. The transformation is applicable to the conception of simple placement heuristics; however, the transformation technique is quite computationally intensive. In [27], an OPP solution technique based on linear integer programming is presented and contingency conditions such as line outages and measurement losses are considered. Execution times were minimal and optimal solutions were found in [27], yet parallel solution techniques for the OPP were not taken into account.

Clearly, differing solutions have been developed for the OPP problem. Yet, while our proposed problem formulations and heuristics do not rely on conventional

instrumentations (such as injection nodes), we approach the OPP problem with coherent computationally efficient heuristics which effectively obtain optimal PMU placements.

CHAPTER 3. PROBLEM FORMULATION

Consider a power system represented as a graph $G(V, E)$. Let V represent the set of vertices (substations of a power system) of a graph. The set of vertices can be denoted as $\{v_1, v_2, v_3, v_4, \cdots, v_n\}$, where $v_i \in V$.

Let E represent a set of edges (transmission lines in a power system) of a graph. For n substations, there may be m transmission lines (edges) connecting all of the substations. These edges can be represented by $e_{jk} \in E$, where j and k represent the vertices of the edge.

A vertex v_i is termed *visible*, if there exists a PMU on v_i or on one of v_i's neighboring vertices. Therefore, we can write an $n \times n$ matrix that represents the visibility of a given graph when certain number of PMUs are placed on the vertices. This matrix is termed as a *visibility matrix*, denoted by $\mathcal{E}$.

To illustrate the concept of the visibility matrix, consider the power system represented in the Fig. 1. Assume that the power system has a PMU placed on vertex v_4. The visibility matrix of the power system is given as follows:

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\left(\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array} \right)
\end{array}
, \ \mathcal{E} = 0
$$

Because the example matrix $\mathcal{E}$ depicts the visibility of the underlying graph, the elements of $\mathcal{E}$ are boolean. From the example power system we can interpret

that substations v_2, v_3, v_5, and v_6 must be visible when the power system has PMU installed on substation v_4. For that reason the example matrix $\mathcal{E}$ has those elements that are being made visible by the PMU installed on v_4 equal to one. For instance: (a) if we consider v_2, then the elements (v_2, v_4) and (v_4, v_2) are equal to one; (b) If we consider v_7, then the elements (v_7, v_4) and (v_4, v_7) are equal to zero. Moreover, we say that the whole graph is visible if and only if each column (or each row) has at least a single element equal to one. Because the visibility matrix is always a square bijectional matrix the condition must either be observed on the rows or the columns and not both.

Based on the above condition, we can formally state the PMU placement problem as: *"Find the minimum number of PMUs that can provide full observability for the entire power system."*

Let X be defined as an $n \times 1$ matrix that represents the PMU placement of a graph (power system). An entry $x_i \in X$ is equal to one when there is a PMU on a substation and zero otherwise. The PMU placement problem for an n-substation power system can be stated as:

$$\min \sum_{i=1}^{n} x_i,$$

where

$$x_i = \begin{cases} 1 & \text{if PMU is installed} \\ 0 & \text{otherwise} \end{cases}$$

such that $\mathcal{E} = 1$.

It is our belief that the PMU placement problem is closely related to the set covering problem [9]. The set covering problem is proven to be NP complete even for the simplest case of identifying a cover for three sets. Because of this similarity we also suspect that PMU is in the class NP. An optimal solution for a large-scale power

system would be inconceivable. Therefore, we must design efficient and effective heuristics.

CHAPTER 4. HEURISTICS

Before we detail our heuristics, we must note that in the generalized case there can be no polynomial time algorithm to tackle the PMU placement problem [9]. Therefore, we must determine theoretical bounds on the performance of each of the proposed heuristic with a hypothetically conceived optimal solution.

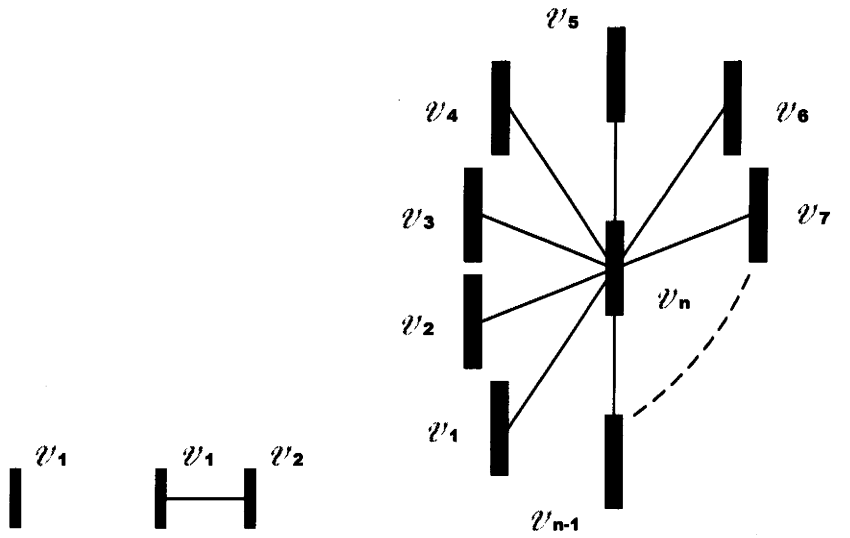
The oracle to derive the performance bounds for each of the heuristics are standard techniques utilized in the analysis of approximation algorithms [13]. Let $\mathcal{O}(\cdot)$ be the *traversal order* that defines the number of traversals that a particular heuristic or an optimal solution must perform to attain complete visibility. Let K represent the solution ratio of a heuristic approach to the optimal solution. Let Q define the traversal order ratio of a heuristic approach to the optimal solution.

We begin the derivation of performance bounds of a heuristic compared to an optimal solution by first describing the worst-case scenarios (see Fig. 2). Let us consider a power system having only one substation (see Fig. 2(a)). In the optimal case, the solution would be obtained by a single traversal and there would be only one PMU. Because of singularity, the heuristic approach also would achieve both the traversal order and the solution (number of PMUs) as one. The values of K and Q are determined subsequently:

$$K = \frac{\text{search order of a heuristic approach}}{\text{search order of the optimal approach}} = \frac{\mathcal{O}(1)}{\mathcal{O}(1)} = 1,$$

$$Q = \frac{\text{solution quality of a heuristic approach}}{\text{solution quality of the optimal approach}} = \frac{1}{1} = 1.$$

When a bi-vertex graph (see Fig. 2(b)) is considered, the optimal solution would be obtained with a single traversal and the PMU could be placed on either one of the vertices. Therefore, both the traversal order and the solution is again equal to one. Unlike the optimal solution, a heuristic approach must traverse both of the vertices



(a) A single vertex graph. (b) A bi-vertex graph. (c) An n-vertex graph

Figure 2: Worst-case scenarios.

to ascertain enough knowledge to install a single PMU. For, the case of a bi-vertex graph, the values of K and Q are given below:

$$K = \frac{\mathcal{O}(1)}{\mathcal{O}(1)} = 1,$$

$$Q = \frac{2}{1} = 2.$$

When an n-vertex graph (see Figure 2(c)) is traversed, say by a cognizant optimal approach, both the traversal order and the solution will once again be equal to one. When a PMU is placed at the vertex v_n the underlying graph becomes completely visible. On the other hand, a heuristic approach may not be as cognizant as the optimal approach. Therefore, a heuristic approach must traverse all of the n vertices to compute an informed solution. In the generalized case, the traversal order and the solution (number of PMUs) to attain complete visibility will vary from heuristic to heuristic.

Below, we detail the six proposed heuristics. Each heuristic is described by outlining the pseudo-code, a brief description, and theoretical analysis (traversal order and solution).

4.1. Greedy

Algorithm 4.1: GREEDY( )

[h!]**Input** :$G(V, E)$

Output :V'

Initialization :$V' = \emptyset$

while $\mathcal{E} = 0$

do
$$\begin{cases} \mathcal{P} = argmax_i(deg(v_i)),\ \forall\ v_i \in V \\ \text{update}(\mathcal{E}) \\ V' \leftarrow V' \cup \mathcal{P} \end{cases}$$

print(V');

The greedy PMU placement heuristic (Algorithm 4.1) takes as an input a graph $G(V, E)$ (the power system). The output of this greedy algorithm would be the set of vertices V' that must have a PMU installed such that the underlining graph is completely observable. The algorithm iteratively places a PMU on the vertex that has the highest degree (incoming or out going edges) until the visibility condition is satisfied ($\mathcal{E} \neq 0$). Because the underlying graph is bidirectional, we only must consider either the incoming or outgoing edges, and not both. This visibility condition is verified by updating the visibility matrix $\mathcal{E}$. To illustrate the updating process, consider the visibility matrix of the example power system as depicted in Figure 1. Because of the greedy approach always chooses the vertex with the maximum degree, the first PMU will always be installed on vertex v_4 that has a degree equal to four.

The visibility matrix will become:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}, \mathcal{E} = 0$$

Because $\mathcal{E} = 0$, the greedy heuristic (Algorithm 4.1) iterates and must place another PMU. There are many possibilities that can be circumvented by narrowing down the potential locations of the PMUs that would result in maximum visibility. Note that v_1, v_2, v_3, and v_5 all have a degree equal to two. However, v_2, v_3, and v_5 are already covered by the PMU placed on v_4. Therefore, installing a PMU on any one of the already covered substations will not increase visibility. The next vertex for selection will be v_1. After placing the PMU on v_1 the visibility matrix would be as follows:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}, \mathcal{E} = 0$$

Finally, the PMU must be placed on v_7, and the visibility matrix would become:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}, \ \mathcal{E} = 1$$

4.2. Single vertex selection

Because the execution time of the greedy heuristic is dictated by the vertex sorting module, the greedy heuristic may become computationally expensive over large-scale power systems. To speed up the PMU placement process, we propose a very simple vertex selection heuristic (Algorithm 4.2).

Algorithm 4.2: SINGLE VERTEX( )

[h!]**Input** :$G(V, E)$

Output :V'

Initialization :$V' = \emptyset$

while $\mathcal{E} = 0$

do $\begin{cases} e_{ij} \leftarrow RAND(E) \\ V' = V' \cup RAND(v_i, v_j) \\ \textbf{update}(\mathcal{E}) \end{cases}$

print(V');

Similar to the greedy heuristic (Algorithm 4.1), the single vertex selection heuristic (Algorithm 4.2) iteratively builds the solution until the visibility constraint is satisfied ($\mathcal{E} = 1$). The solution is built by taking an input graph $G(V, E)$ and randomly selecting an edge $e_{jk} \in E$. A PMU will be placed on one of the vertices (v_i and v_j) of the selected edge. The visibility matrix is updated similar to the approach undertaken in the greedy heuristic (Algorithm 4.1). Moreover, we also maintain the set of edges E so that we can select a unique edge in each iteration. This edge selection process continues until the entire power system is observable.

To illustrate the process of a single vertex selection, consider the visibility matrix of the example power system depicted in Figure 1. The edges e_{jk} will be chosen at random and a PMU will be placed on vertices v_i or v_j. Assume the first PMU is randomly chosen to be placed on v_5 from selecting the edge e_{4-5}. The visibility matrix will become:

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\left(\begin{array}{ccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0
\end{array} \right)
\end{array}
, \ \mathcal{E} = 0
$$

With $\mathcal{E}$ being equal to zero, the single vertex heuristic iterates and must place another PMU. Because the edge e_{4-5} was chosen, e_{4-5} can be eliminated from the selection process. Because v_5 and v_7 are observable, the edge e_{5-7} also can be eliminated. The next PMU may be randomly chosen to be placed on v_1 from the

edge selection e_{1-2}. The visibility matrix will become:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}, \mathcal{E} = 0$$

Again, $\mathcal{E} = 0$ so the heuristic must iterate and place another PMU. Because of the only choice left, the PMU must be placed on v_6, and the visibility matrix would become:

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}, \mathcal{E} = 1$$

4.3. Double vertex selection

Although the execution time of the single vertex algorithm is superior to the greedy algorithm, the single vertex algorithm may perform poorly in a worst-case scenario, such as the n-vertex graph (see Fig. 2(c)). If any vertex other than v_n is selected,

then we need to repeatedly select vertices to attain complete observability. The selection process may result in worse case performance. To prevent such unsatisfactory performance, a natural extension to Algorithm 4.2 is presented in Algorithm 4.3.

Algorithm 4.3: BOTH NODES( )

[h!]**Input** :$G(V, E)$

Output :V'

Initialization :$V' = \emptyset$

while $\mathcal{E} = 0$

$$\textbf{do} \begin{cases} e_{ij} \leftarrow RAND(E) \\ V' = V' \cup v_i \cup v_j \\ \textbf{update}(\mathcal{E}) \end{cases}$$

print(V');

The Algorithm 4.3 selects an edge from the edge matrix and places the PMUs on both of the selected vertices. Although the traversal order in the Algorithm 4.3 also is given by the output matrix V', yet the heuristic may provide an optimum solution for the n-vertex graph (see Figure 2(c)). After placing PMUs, the output matrix and the visibility matrix will be updated along with the edge matrix E so that no edge can be selected twice. This process is repeated until $\mathcal{E} = 1$. To corroborate the selection process we can consider the n-vertex graph (see Figure 2(c)). The entire system will be visible when we install PMUs on both vertices of an edge.

To illustrate the process of a double vertex selection, consider the visibility matrix of the power system depicted in Figure 1. The edges e_{jk} may be chosen at random and a PMU is placed on both vertices v_i and v_j. The first two PMUs must be chosen to be placed on v_3 and v_4 from selecting the edge e_{3-4}. As a result, the

visibility matrix will become:

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\left(\begin{array}{ccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array} \right)
\end{array}
, \ \mathcal{E} = 0
$$

Because $\mathcal{E} = 0$, the double vertex heuristic iterates and must place at least two more PMUs. Because the edge e_{3-4} was chosen, e_{3-4} can be eliminated from the selection process. Because v_1, v_2, v_3, v_4, v_5, and v_6 are observable, the edges e_{1-2}, e_{1-3}, e_{2-4}, e_{4-5}, and e_{4-6} also can be eliminated. The next two PMUs are then chosen to be placed on v_5 and v_7 from the edge selection e_{5-7}. That being the only choice, the visibility matrix will become:

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\left(\begin{array}{ccccccc}
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1
\end{array} \right)
\end{array}
, \ \mathcal{E} = 1
$$

Table 1 summarizes the search and solution ratio of all the proposed heuristics

compared to the optimal approach for the generalized n-vertex graph. Because of our simple oracle, the reader will find it a trivial exercise to derive the aforementioned search and solution ratios. Moreover, because the Greedy, Single Vertex, and Double Vertex heuristics, all have their corresponding search ratios less than or equal to n, the heuristics belong to the class linear-time.

Table 1: Search order ratio and solution ratio.

Heuristics	Q	K
Greedy	n	1
Single Vertex	V'	depends on V'
Double Vertex	V'	depends on V'

4.4. A-Star Algorithm

The A-star heuristic (hereafter referred as A*) is a quadratic-time, best-fit first graph search that finds the least cost path from the initial vertex to the goal vertex [8]. A cost function f computes each vertex's associated cost. The value f for a vertex v_i that is the estimated cost of the cheapest solution through v_i, is computed as:

$$f(v_i) = h(v_i) + g(v_i), \tag{1}$$

where $g(v_i)$ is the search-path cost from the initial vertex to the current vertex v_i and $h(v_i)$ termed the heuristic is a lower-bound estimate of the path cost from v_i to the goal vertex.

For the PMU problem studied in this paper, the A* algorithm uses the vertex coverage as heuristic and PMUs placed as the cost. Applying Eq. (1) for the PMU problem will bias the result towards $g(v_i)$ and may not properly represent the system. To maintain a weight balance, the ratio of the heuristic to cost may be an appropriate measure. Such a ratio represents the cost per coverage at any point in the power system.

$$f(v_i) = \frac{h(v_i)}{g(v_i)}. \tag{2}$$

The vertex corresponding to the maximum ratio value of the entire tree is selected. When a vertex is chosen, the paths from that vertex will be searched. This process continues until full coverage is achieved. The A* algorithm will always find the optimal solution but may not work for large-scale problems due to time and memory issues [8].

Algorithm 4.4: ASTAR( )

[h!]**Input** :$G(V, E)$

Output :V'

Initialization :$V' = \emptyset$

for each vertex $v_i \in V$

$f(v_i) \leftarrow \frac{h(v_i)}{g(v_i)}$

repeat

Assign vertex with maximum f value to x

update($\mathcal{E}$)

for each child of x with vertices v_i'

$f(v_i') \leftarrow \frac{h(v_i')}{g(v_i')}$

until $\mathcal{E} = 1$

print(V');

The Algorithm 4.4 takes an input graph (the power system). The output of the A* algorithm V' is the set of vertices where a PMU has been installed such that the system is completely observable. A* starts at a "zero step" where no vertices have been searched. The first step searches the set of vertices v_i and computes f. Each vertex has an associated heuristic cost (coverage of the system, $h(v_i)$) and for every PMU placed, there is a distance cost, $g(v_i)$. The ratio of the heuristic to the

distance cost is computed and the maximum value is selected and assigned a variable x. The visibility matrix $\mathcal{E}$ is updated for that path explored and the children of x are searched. The process continues until $\mathcal{E}$=1. The path explored that completely observes the system is the optimal solution.

To illustrate how the A* works, consider the 5 bus system depicted in Figure 3. A tree graph is shown to display the process of A* on the power system in Figure 4.

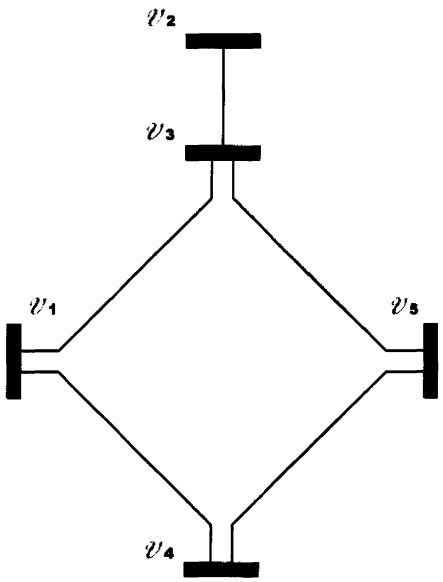


Figure 3: An example 5 bus power system.

The search-tree nodes include partial assignment of PMUs to nodes and the value of f (the cost of the partial assignment). The assignment of PMUs to m vertices is indicated by an m-digit string, a_0, a_1, ..., $a_m - 1$, where:

$$a_i = \begin{cases} 1 & \text{if PMU is installed,} \\ 0 & \text{otherwise.} \end{cases}$$

The root node includes the set of all uncovered vertices [00000]. Next we expand all five possibilities of placing the first PMU and calculate the corresponding f values

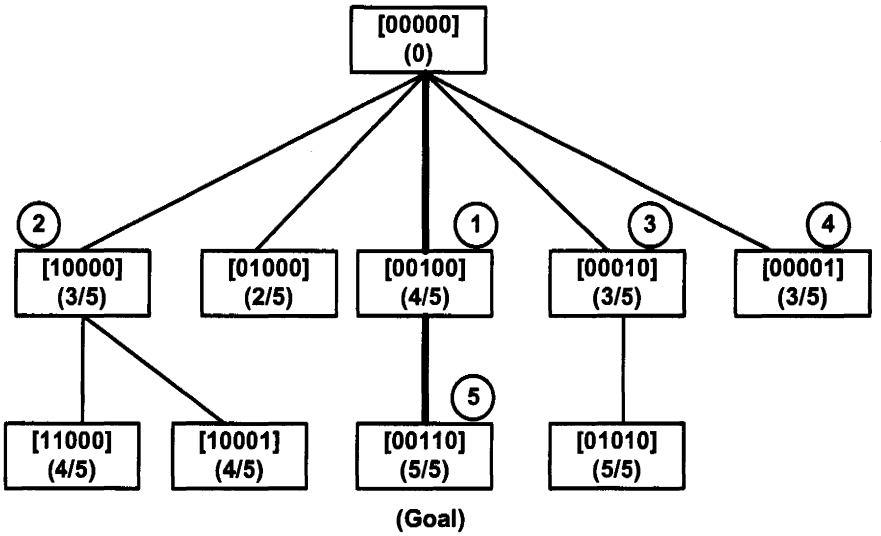


Figure 4: A* algorithm example.

as in Figure 4. A visibility matrix is calculated for each expansion. Placing a PMU on v_3 (node [00100]) results in $f(v_4)$ being equal to 4/5. The $g(v_i)$ value for any node is equal to five, which is the number of vertices in a given power system. The $h(v_3)$ value is equal to four, which is the number of vertices that are visible as a result of placing a PMU at v_2. Because 4/5th is the highest ratio of the entire list of candidates, nodes [10000], [01000], [00100], [00010] and [00001], the A* algorithm expands node [00100]. This expansion is achieved by calculating the $f(v_i)$ ratios of the remaining unobserved vertices, in this case v_4.

Since $\mathcal{E} = 0$, the A* algorithm must expand the first node with the next highest ratio. The algorithm searches on a depth basis, starting at the depth of zero (node [00000]) and increasing until the entire tree is searched. The first node found with the next highest ratio v of 3/5 is v_1 (node [10000]). The A* algorithm expands v_1 to create node [11000] and node [10001], and then calculates the f value for the nodes. Again $\mathcal{E} = 0$, so the A* algorithm expands another node with the next highest ratio

value, in this case node [00010]. The A* algorithm expands v_4 to create node [01010]. Because node [00110] was previously calculated when node [00100] was expanded on, the node is not recalculated.

Attached to some of the nodes are numbers in circles. These represent the sequence in which the nodes are chosen for expansion. The bold lines represent the path that leads to the optimal solution. The process continues until $\mathcal{E} = 1$ that is achieved when node [00110] is expanded. Therefore the system is completely observable. The node [00110] represents full coverage at the minimum cost, and therefore referred to as the goal node.

4.5. Parallel A-Star Algorithm

To further the speed of execution, a parallel version of the A* was developed. The parallel A*, described in algorithm 4.6, allows multiple concurrent expansions of nodes with the best cost per coverage ratios. With parallel expansion of nodes, a quicker arrival at the goal node is expected. The parallel A*, given n number of processors, assigns a processor as the *root* processor and $n-1$ processors as *expansion* processors. The root processor selects and distributes to expansion processors the node for expansion. Expansion processors receive a node N_v, as given by the root processor, and expand such node to output the child nodes of the inputted node N_v. The root processor gathers the child nodes from expansion processors examining for goal nodes. Once one of the child nodes is ascertained to be a goal node, such goal node is outputted as the PMU placement solution. In the case that no goal node is found amongst the received child nodes, the child nodes are added to the OpenSet from which the root processor will make the next $n-1$ node selections for expansion. Figure 5 illustrates how the parallel A* works, with 4 processors.

Algorithm 4.5: PARALLEL ASTAR( )

Input : $G(V,E)$; $P = \{p_0, p_1, \cdots, p_{n-1}\}$

Output : Goal node

Initialization :$V = \emptyset$

while $\mathcal{E} \neq 1$ {

- **for each** vertex $v_i \in V$
- $f(v_i) \leftarrow \frac{h(v_i)}{g(v_i)}$
- **for each** processor $p_i \in P$ $(i \geq 1)$
  - **do** {
    - processor p_0 sends the i^{th}
    - best open node to p_i
- **for each** processor $p_i \in P$ $(i \geq 1)$
  - **do** {
    - p_i expands i^{th} best open node
    - p_i sends the child nodes to p_0
- processor p_i receives child nodes from $p_1, \cdots, p_{n-1}$
- **if** goal node found: output goal node and quit

}

In Figure 5, the parallel A* begins at a "zero step" where no vertices have been searched, and generates [10000], [01000], [00100], [00010], and [00001] as expansion results. The root processor (p_0) selects the nodes with the first and second best cost per coverage ratio; [00100] is given to processor 2, [00010] is given to processor 2, and [10000] is sent to processor 3. After concurrent expansion by processors p_1, p_2, and p_3, [00110] is found by processor p_1, and is the resulting goal node.

4.6. A* Pruning Methods

Various pruning techniques may to be used to increase efficiency of the A* algorithm to find (sub)optimal solutions. These pruning techniques belong to the class subquadratic-time heuristics [10].

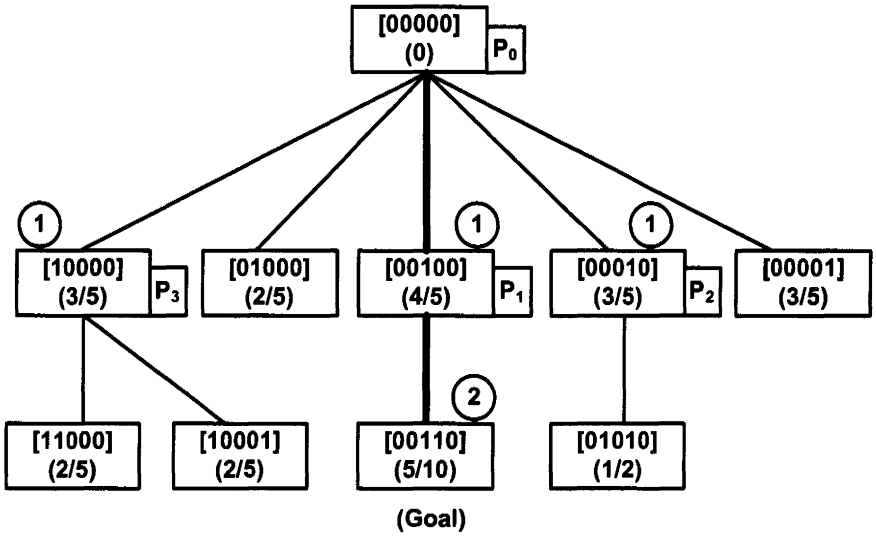


Figure 5: Parallel A* algorithm example.

The first method is a depth optimization. This optimization searches the A* tree until a certain depth d is reached (number of PMUs placed). Only the path with the maximum ratio from a search is considered in future expansions. The selection of a depth of the tree will decrease the execution time and return (sub)optimal solutions.

The second method is a restricted list optimization. This optimization maintains an m length list. Only the ratios of highest value ratios are stored in this list. This technique may decrease time spent searching for the best ratio because there is a finite number of possibilities to sort through.

4.7. Distance-Level Algorithms (DLA)

A family of PMU placement algorithms, called Distance-Level Algorithms (DLA), has been developed with the aim of obtaining PMU placement solutions in a graceful manner. The goal of distance-level algorithms is the minimal assignment of PMUs in a power system, thereby providing full measurability of the network. A host of

concepts have been derived from graph theory and will be presented firstly.

In Figure 6 we see an arbitrary graph G. From graph theory, a graph G is connected if there exists a $u - v$ path between every two vertices of G [26]. By inspection, it is clear that for every two substations in a power system (substation u, and substation v), there exists a $u - v$ path, symbolized by the power line connection. Consequently, we can abstract a power system as a connected graph. Also in graph theory, there exists the subjects of domination sets, and domination number of a graph G. The dominating set S of G is a set of vertices in $V(G)$ that dominate every vertex in $V(G) - S$. The domination number of a graph G , is the minimal cardinality of the dominating sets of G, and is symbolized as $\gamma(G)$. Another utilized concept, from graph Theory, is the distance $d(u, v)$ between two vertices u and v. $d(u, v)$ is defined as the minimum of the lengths of the $u - v$ paths of G [26].

Algorithm 4.6: DISTANCE-LEVEL CONSTRUCTION (DLC)( )

Input : G, V_r (root vertices)

Output : $DLC(G)$

for each $v \in V(G)$

do $\{ DL[v] = -1$

for each $v \in V_r$

do $\{ DL[v] = 0$

for each $v \in V(G)$

do $\begin{cases} \textbf{if } DL[v] == -1 : \\ DL[v] = DL[Pt(v)] + 1 \end{cases}$

Based on the preceding graph theory concepts, we define the Distance-Level Construction (DLC) as the assignment of distance levels to all the vertices of a connected graph G. Algorithm 4.6 outlines the pseudocode of the DLC. A distance level graph, the product of the DLC, is shown in Figure 7.

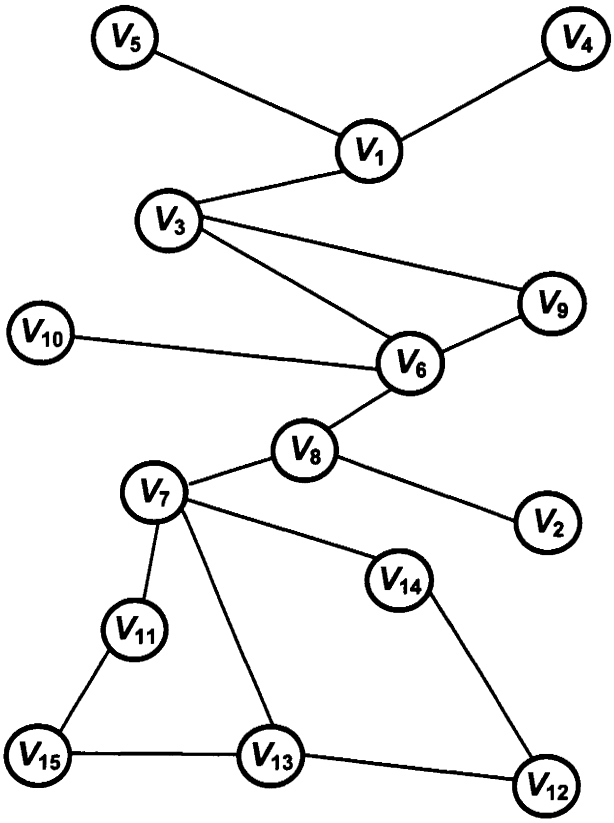


Figure 6: An arbitrary graph G.

We see G', in Figure 7, a DLC of G ($DLC(G)$), showing the distance levels from vertex v_1. We define the *root vertex (vertices)* as such vertex that belongs to *distance-level 0* (dl_0). In Figure 7 v_1 is the root vertex. Vertices v_3, v_4, v_5 belong to dl_1, while v_6 and v_9 belong to dl_2, and so on. Given the $SDLC(G)$ of a graph G, and a distance-level dl_i, we label dl_{i+1} as the child-level to dl_i. In the same manner, given the $DLC(G)$ of a graph G, and a distance-level $dl_i (i \geq 1)$, we label dl_{i-1} as the parent-level to dl_i.

Assume that a vertex $v \subseteq V(G)$ is assigned to dl_i, vertices $v_1, v_2 \subseteq V(G)$ are assigned to dl_{i+1}, and v is adjacent to v_1, v_2. Vertex v is named the parent-vertex of v_1, v_2; similarly, v_1, v_2 are labeled as children-vertices of vertex v. Now, assume that a vertices v_i, v_j and $v_k \subseteq V(G)$ belong to dl_i. Also, assume that v_i are adjacent v_j,

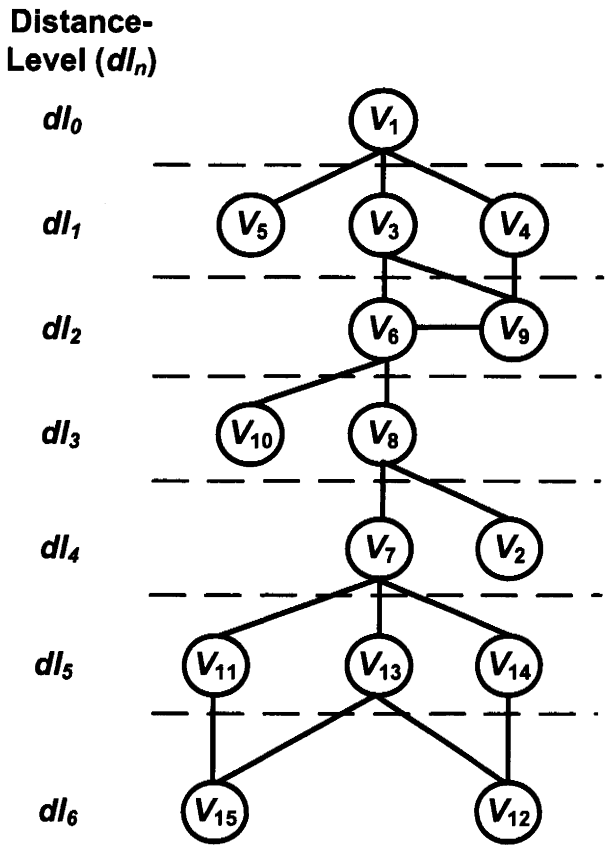


Figure 7: A Distance-leveled Graph.

and that neither v_i nor v_j are adjacent to v_k. Based on the preceding assumptions, we say $\{v_i, v_j, v_k\}$ are peer-vertices on dl_i, while $\{v_i, v_j\}$ are adjacent peer-vertices on dl_i. Figure 8 provides an illustration of parent, children, and peer-vertices as presented.

4.8. Distance-Level Heuristics (DLHs)

We presented the DLC as a fundamental construction for the distance level algorithms. A foundational tool for the Distance Level Algorithms are the distance-level heuristics (DLH). The DLHs are heuristics designed to peruse a given set of vertices and provide, as an output, a single vertex best desired for PMU assignment.

Listed in Table 2 are symbols and the corresponding definitions, as utilized in the descriptions of the various distance-level heuristics.

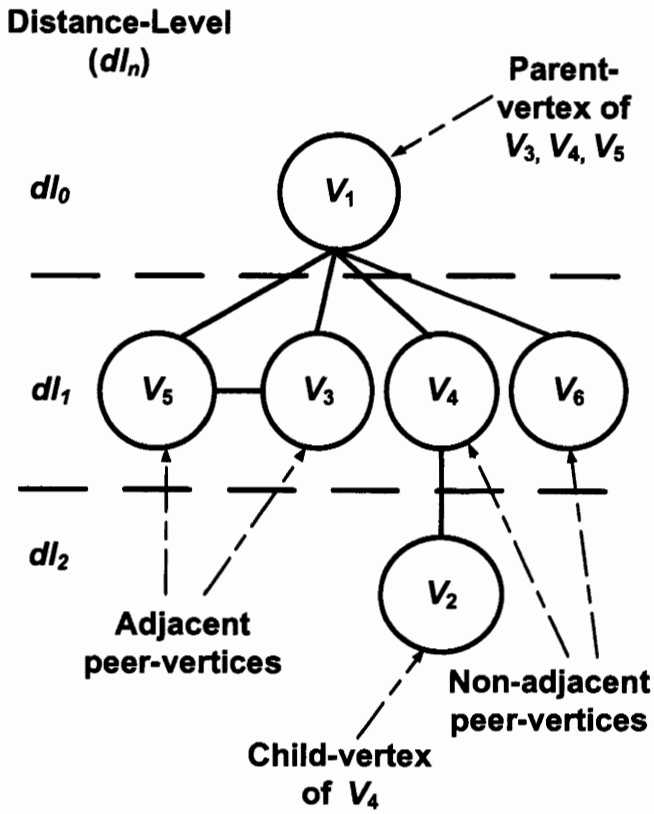


Figure 8: An illustration of parent, children, and peer-vertices.

Described below are four distance-level heuristics, *MAX-C*, *MAXSEQ-CPR*, *MAX-R1L*, *MAX-R2L*:

- *MAX-C*

  Input: $DLC(G)$; dl_i.

  Output: v_{pmu}.

  This heuristic randomly selects a vertex from $V_{max(|N[v]|),\ dl_i}$ as the v_{pmu}.

- *MAXSEQ-CPR*

  Input: $DLC(G)$; dl_i.

  Output: v_{pmu}.

  In the case that $\mid V_{max(n_{ch}),\ dl_i} \mid = 1$, this heuristic selects the only vertex of $V_{max(n_{ch}),\ dl_i}$ as the v_{pmu}. In the event that $\mid V_{max(n_{ch}),\ dl_i} \mid > 1$, from $V_{max(n_{ch}),\ dl_i}$

Table 2: DLA symbols and definitions

$N[v]$	Set of vertices dominated by vertex v.
$\mid N_{pt}[v] \mid$	Number of dominated parent-vertices of vertex v.
$\mid N_{pr}[v] \mid$	Number of dominated peer-vertices of vertex v.
$\mid N_{ch}[v] \mid$	Number of dominated child-vertices of vertex v.
$v_{max(\mid N[ch,dl_i] \mid)}$	Vertex on dl_i with maximal number of dominated child-vertices.
V_{pt}	Set of parent-vertices.
$V_{pt(dl_i)}$	Set of parent-vertices, on dl_i
V_{dl_i}	Set of vertices, on dl_i
$V_{max(n_{ch}),\ dl_i}$	Set of vertices, on dl_i, with maximal number of dominated child-vertices.
$V_{max(n_{ch},\ n_{pr}),\ dl_i}$	Set of vertices, on dl_i, with maximal number of dominated child-vertices and peer-vertices.
$V_{max(\mid N[v] \mid),\ dl_i}$	Set of vertices, on dl_i, with maximal number of dominated vertices.
$V_{dl_i\{n_{pt},n_{pr},n_{ch}\}}$	Set of vertices, on dl_i, with n_{pt} parent-vertices, n_{pr} peer-vertices, and n_{ch} child-vertices.
$Pt[V]$	The set of parent-vertices of vertices in set V
v_{pmu}	Vertex to be assigned a PMU

it selects $V_{max(n_{ch},\ n_{pr}),\ dl_i}$. In the case that $\mid V_{max(n_{ch},\ n_{pr}),\ dl_i} \mid = 1$ the lone vertex of $V_{max(n_{ch},\ n_{pr}),\ dl_i}$ is set as v_{pmu}. In the case that $\mid V_{max(n_{ch},\ n_{pr}),\ dl_i} \mid > 1$, from $V_{max(n_{ch},\ n_{pr}),\ dl_i}$ it selects $V_{max(\mid N[v] \mid),\ dl_i}$. From $V_{max(\mid N[v] \mid),\ dl_i}$ it randomly selects a vertex as the v_{pmu}.

- *MAX-R1L*

  Input: $DLC(G)$; dl_i

  Output: v_{pmu}

  This heuristic randomly selects a vertex from $V_{max(\mid N[dl_i] \mid)}$ as the v_{pmu}

- *MAX-R2L*

  Input: $DLC(G)$; dl_i; dl_{i+1} $(0 \leq i \leq \mid V(G') \mid - 1)$

  Output: v_{pmu}

  This heuristic randomly selects from $Pt[V_{dl_i\{1,\ 0,\ 0\}}]$ a parent-vertex having a single child-vertex that is adjacent to non other but the parent vertex. In the

case that $Pt[V_{dl_i\{1,\ 0,\ 0\}}] = \emptyset$, it constructs from $V_{pt(dl_i)}$ the set $V_{max(n_{ch}),\ dl_i}$ and selects the lone vertex as v_{pmu} if $\mid V_{max(n_{ch}),\ dl_i} \mid = 1$. In the event that $\mid V_{max(n_{ch}),\ dl_i} \mid > 1$, from $V_{max(n_{ch}),\ dl_i}$ it constructs $V_{max(|N[.]|),\ dl_i,\ dl_{i+1}}$. If $\mid V_{max(|N[.]|),\ dl_i,\ dl_{i+1}} \mid = 1$, it selects the lone vertex of $V_{max(|N[.]|),\ dl_i,\ dl_{i+1}}$ as the v_{pmu}; otherwise if $\mid V_{max(|N[.]|),\ dl_i,\ dl_{i+1}} \mid \geq 1$ it randomly selects a parent-vertex from $Pt[V_{max(|N[.]|),\ dl_i,\ dl_{i+1}}]$.

The Distance-level algorithms create the dominating set of G, V_{pmu}, using any of the distance-level heuristics (DLH) presented above. Given the $DLC(G)$, a distance-level algorithm, directed by the employed DLH, peruses each level(in a bottom-up fashion), selecting favorably adjudged vertices and adding such vertices to the dominating Set V_{pmu} under formulation. In other words, a distance-level algorithm scans the $DLC(G)$, beginning at the highest numbered distance level(dl_{n-1}) and ending at the lowest numbered distance level(dl_0). During these scans, PMUs are assigned to vertices based on the recommendations of the employed DLH.

4.9. Sequential Distance-level Algorithm

Implicit in the function of the DLC is the organization of the vertices of an arbitrary graph. With the introduction of the DLC and the DLH, we now introduce two Distance Level Algorithms. The first distance-level algorithm is the sequential distance-level algorithm, also known as *Seqla*. The Seqla is a distance-level algorithm that is specifically designed for single processor execution. Given a graph G, the Seqla, as detailed in Algorithm 4.7, initially creates the distance-level construction of G. Subsequently, Seqla scans the set of vertices belonging to dl_{n-1}, selecting favorably adjudged vertices and adding such vertices to the dominating Set V_{pmu} under formulation, until all vertices on dl_{n-1} are ascertained covered (either by assigned PMUs or by previously installed conventional measurement units).

Algorithm 4.7: SEQUENTIAL DISTANCE-LEVEL ALGORITHM( )

Input :$G' = DLC(G)$

Output :V_{pmu}

while $V(G')$ is not dominated by V_{pmu}

do
- **while** $V(G', dl_i)$ are not dominated
  - **do**
    - $v_i = DLH(G', dl_i)$
    - $V_{pmu} = V_{pmu} \cup v_i$
    - G' =update(G', V_{pmu})
- $dl_i = dl_{i-1}$

4.10. Parallel Distance-level Algorithm

Algorithm 4.8: PARALLEL DISTANCE-LEVEL CONSTRUCTION. (PDLC)( )

Input : $DLC(G)$

Output : $PDLC(G)$

for each $dl_i \in V(G)$

do map dl_i to discrete sine-wave.

output $PDLC(G)$

In a bid to reduce execution times (particularly for large systems), we developed the parallel distance-level algorithm, also denoted as *Parla*. Parla, detailed in algorithm 4.9, assigns PMUs in similar fashion to Seqla; however, special provision is made for concurrent assignment of PMUs by multiple processors. In Figure 9, an

horizontal view of $DLC(G)$ is depicted, while Figure 11 depicts the parallel distance-level construction of G for two executing processors. The parallel distance-level construction (PDLC) is a special distance-level construction for the Parla. Given the number of processors for parallel execution, the PDLC, as detailed in Algorithm 4.8, constructs a distance-level graph that allows the specified number of processors to execute the Parla seamlessly. The PDLC maps the distance-levels of $DLC(G)$ to a discrete sine wave function parameterized to the specified number of processors (Figure 10). As a result of this mapping, the PDLC of a distance-level graph suitable for parallel processing by the specified number of processors is formed.

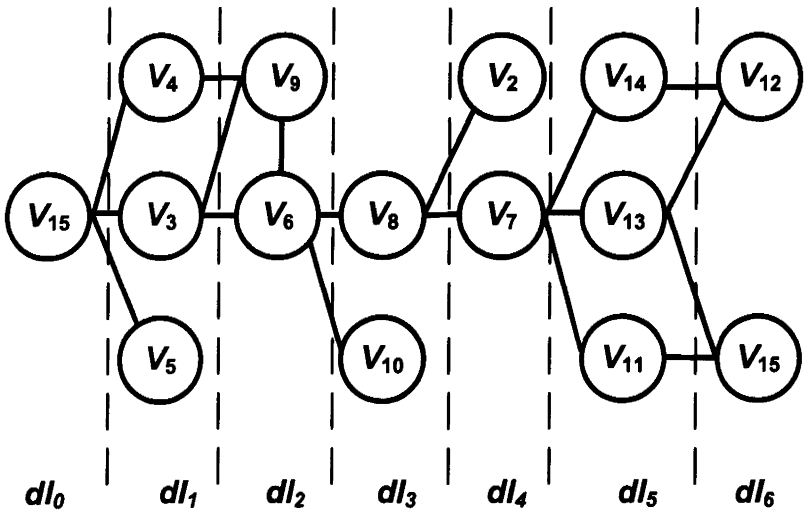


Figure 9: Distance-Level Construction of graph G: a horizontal view

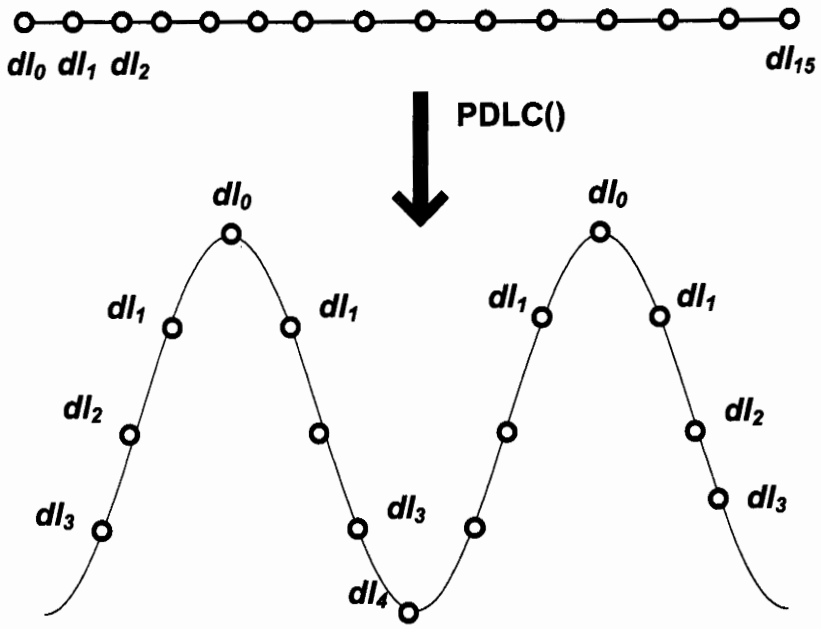


Figure 10: An illustration of distance-levels mapped to sine-wave function.

Algorithm 4.9: PARALLEL DISTANCE-LEVEL ALGORITHM. (PDLA)( )

Input : $PDLC(G)$, $P = \{p_0, p_1, \cdots, p_{n-1}\}$

Output : V_{pmu} (vertices assigned PMUs)

Initialization : $V_{pmu} = \emptyset$

processor p_0 applies $DLH()$ to bottom apexes

processor p_0 updates V_{pmu}

for each processor $p_i \in P$ $(i \geq 1)$

do $\begin{cases} p_i \text{ runs } DLH() \text{ on } G_i \\ p_i \text{ updates } V_{pmu} \end{cases}$

processor p_0 applies $DLH()$ to top apexes

processor p_0 updates V_{pmu}

processor p_0 outputs V_{pmu}

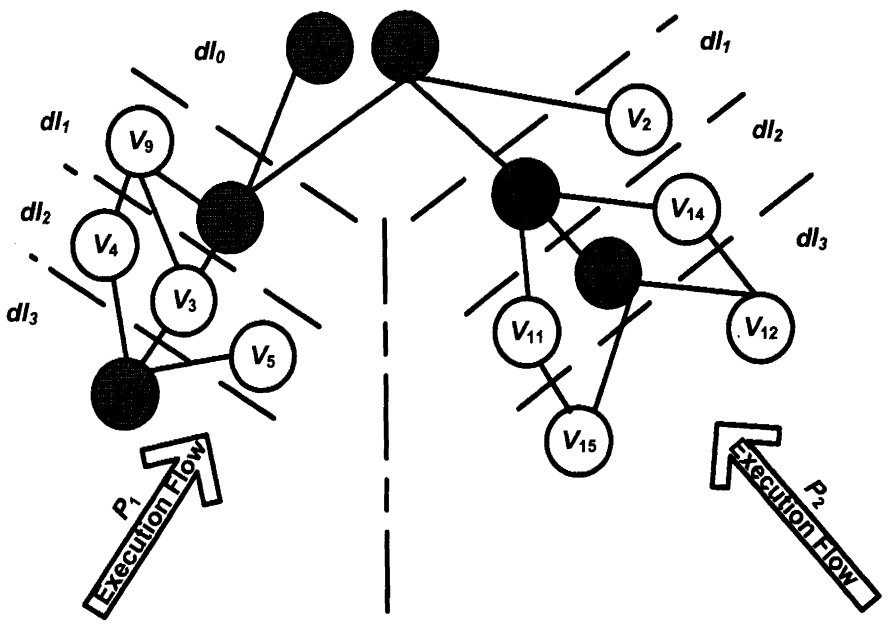


Figure 11: A Parallel Distance Level Construction of graph G

CHAPTER 5. EXPERIMENTAL RESULTS

We demonstrate the effectiveness of our proposed heuristics by comparing the performance results of each of the algorithms programmed in Matlab on the IEEE standard bus systems (14 bus, 30 bus, 57 bus, 118 bus, and 300 bus [14]) and a 2,383 bus system. The performance matrix was the minimum number of PMUs that a heuristic can place to ensure full observability. The results are classified by the bus size of the system in the subsequent sections.

Table 3 shows the placement results of the A* and the A* pruning methods. The A* heuristic finds the optimal solution, therefore we use the algorithm's results as a benchmark. Due to the quadratic run-time of the A* algorithm, some results for power systems larger than 30 buses were unobtainable; they are denoted by $\times$.

Table 3: A* Placement Results

Bus Systems	A*	Depth Method		Restricted List Method
		$d = 1$	$d = 2$	
14 bus	3	3	3	3
30 bus	7	7	7	7
57 bus	18	12	12	12
118 bus	29	34	$\times$	34
300 bus	112	110	$\times$	109
2,383 bus	$\times$	1040	$\times$	1032

5.1. 14 Bus Power System

The depth method for the A* heuristic performed the best on the 14 bus power system, finding the optimal solution of four PMUs in 0.0028 seconds as seen in Table 4. The 14 bus power system is a small test case that exemplifies the strength of the A* heuristic. Because of this property, the depth method for optimizing the A* heuristic further improves the heuristic's effectiveness. Table 5 shows that the Double Vertex heuristic executes faster than the Greedy heuristic. Figure 12 shows that the Greedy

heuristics found a solution of five PMUs. The Double Vertex heuristic performed worst with a solution of seven PMUs while the Single Vertex heuristic found the optimal solution of three PMUs. Due to the fact that the Greedy, Single and Double Vertex heuristics are random and non-deterministic, we ran 100,000 trials to instill confidence in the solutions of the heuristics.

Table 4: A* Run-Time Results In Seconds

Bus Systems	A*	Depth Method		Restricted List Method
		$d = 1$	$d = 2$	
14 bus	0.49	0.0028	0.0047	0.0043
30 bus	160.82	0.052	0.095	0.1198
57 bus	675.01	0.3673	0.4351	0.5417
118 bus	3051.88	2.0369	×	4.18
300 bus	1722.43	4.7814	×	105.36
2,383 bus	62244.52	1702.74	×	5033.52

Table 5: Non-Deterministic Heuristics Run-Time Results Per Trial In Seconds

Bus Systems	Heuristics		
	Greedy	Single Vertex	Double Vertex
14 bus	0.000372	0.000365	0.000287
30 bus	0.00082	0.00101	0.00072
57 bus	0.00163	0.00281	0.00189
118 bus	0.00656	0.01231	0.00779
300 bus	0.07086	0.15544	0.08839
2,383 bus	17.2198	99.7091	60.8226

The parallel A* heuristic, using one to ten processors, also found the optimal results of three PMUs as seen in Table 6. The run-time for two processors was 0.48 seconds, and 1.31 seconds for ten processors as seen in Table 7. The parallel A* communication-computation ratios, as shown in Figure 13, describe the ratio of communication intensity to computational intensity for different number of processors. It is observed from Figure 13 that the communication-computation ratio decreases as the number of processors increases.

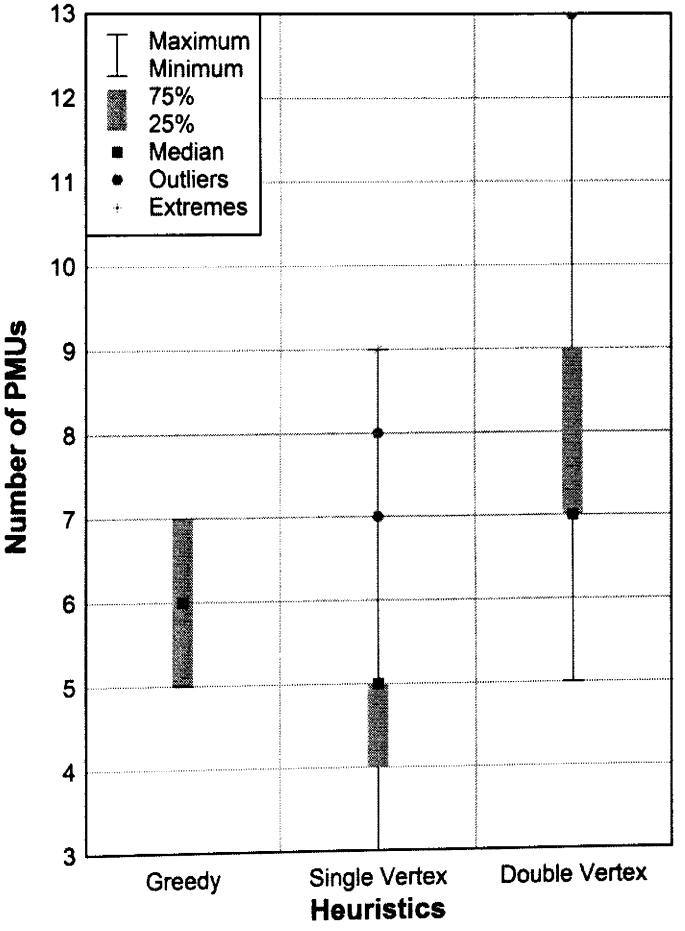


Figure 12: IEEE standard 14 bus results.

Table 6: Parallel A* Placement Results

Bus Systems	Number of Processors		
	Two	Five	Ten
14 bus	3	3	3
30 bus	7	7	7
57 bus	12	11	14
118 bus	29	32	36
300 bus	112	113	121
2383 bus	1920	1940	1955

Table 8 shows that *Seqla* also obtains the optimal PMU solutions using the various distance-level heuristics. The results of *Parla* executions which utilizes all the

Table 7: Parallel A* Run-Time Results In Seconds

Bus Systems	Number of Processors		
	Two	Five	Ten
14 bus	0.48	1.31	3.36
30 bus	168.33	191.01	43.3
57 bus	755.40	115.62	222.41
118 bus	3611.16	330.54	1292.82
300 bus	2017.71	5612.88	20541.12
2383 bus	11492.52	73250.50	451565.11

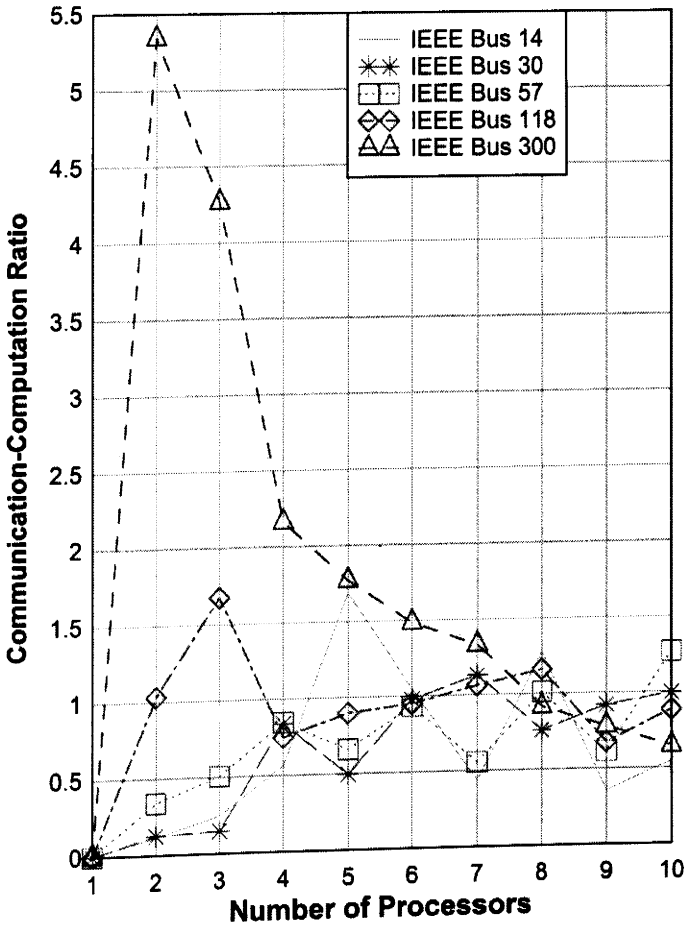


Figure 13: Parallel A* communication-computation ratios.

vertices as single root-vertices, presented in Table 9, shows the *Parla* was as effective as *Seqla* in finding the optimal PMU solutions. For *Seqla*, DLA heuristics *MAX-C*,

MAXSEQ-CPR, *MAX-R1L*, and *MAX-R2L* compared fairly well, as minimal variance is noted between the results of the different distance-level heuristics. The *Seqla* found PMU solutions in 0.1 seconds, as seen in Table 10.

Table 8: Sequential DLA IEEE standard 14 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	3	3
MAXSEQ-CPR	3	4
MAX-R1L	3	4
MAX-R2L	3	3

Table 9: Parallel DLA IEEE standard 14 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	3	4
	4	3	5
	6	3	4
	8	3	5
	10	3	3
MAXSEQ-CPR	2	3	4
	4	3	5
	6	3	4
	8	3	5
	10	3	4
MAX-R1L	2	3	4
	4	3	5
	6	3	4
	8	3	5
	10	3	3
MAX-R2L	2	3	4
	4	3	5
	6	3	4
	8	3	5
	10	3	4

The parallel DLA found the optimal solution of three PMUs (Table 9) for the

Table 10: DLA Run-Time Results In Seconds

Bus Systems	DLA	
	Sequential	Parallel
14 bus	0.1	0.01
30 bus	0.68	0.51
57 bus	3.04	2.64
118 bus	18.23	13.46
300 bus	219.05	127.58
2383 bus	105.56	66.13

IEEE 14 bus system in 0.01 seconds as shown in Table 10. However, compared to *Seqla*'s execution results of the IEEE bus 14 system, a wider margin of error is observed for the *Parla*'s executions, as shown in Table 9. Also, for the *Parla* executions, the error margin was roughly equivalent for all DLA heuristics.

5.2. 30 Bus Power System

The depth method for the A* heuristic performed the best on the 30 bus power system as well, finding the optimal solution of seven PMUs in 0.052 seconds as seen in Table 4. The A* heuristic found the optimal solution in 4,922 seconds. The 30 bus system exposes the disadvantage of the A* heuristic and the advantage of the two pruning methods for the A* heuristic. The two pruning methods for the A* heuristic reduced the execution time by a factor of 40,000 or greater, and still found the optimal solution. The results of the non-deterministic heuristics on the 30 bus case are shown in Figure 14. The only heuristic unable to reach the optimal solution was the Greedy heuristic. Hence, the Greedy heuristic performed the worst.

The parallel A* heuristic also found the optimal results of seven PMUs as seen in Table 6. The run-time for two processors was 168.33 seconds, and 43.3 seconds for ten processors as seen in Table 7.

The sequential DLA found the optimal solution of seven PMUs for the 30 bus system while maintaining a low margin of error, as seen in Table 11. DLA heuristic

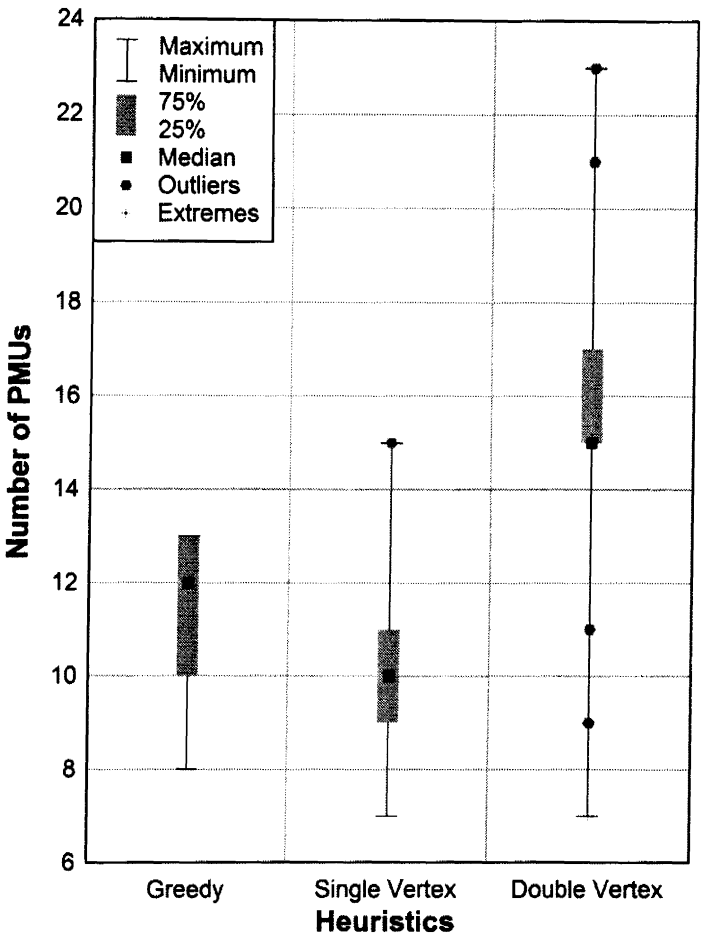


Figure 14: IEEE standard 30 bus results.

MAX-R1L had the highest margin of error, assigning nine PMUs. As seen in Table 10, *Seqla* reported a run-time of 0.68 seconds.

Table 11: Sequential DLA IEEE standard 30 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	7	8
MAXSEQ-CPR	7	8
MAX-R1L	7	9
MAX-R2L	7	8

The parallel DLA also found the optimal solution of seven PMUs for the 30 bus

system. However, compared to *Parla*'s execution results of the IEEE bus 14 system, a wider margin of error is observed, as shown in Table 12. The error margin was greatest for executions utilizing four, six, eight processors and least for executions utilizing two and ten processors. DLA heuristic *MAX-R1L* performed best in assigning the optimal number of PMUs, irrespective of the number of processors. Table 10 shows that *Parla* found the PMU solutions 0.17 seconds faster than *Seqla*.

Table 12: Parallel DLA IEEE standard 30 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	7	9
	4	7	11
	6	10	13
	8	8	10
	10	7	9
MAXSEQ-CPR	2	7	8
	4	8	11
	6	7	10
	8	8	10
	10	7	9
MAX-R1L	2	7	9
	4	8	11
	6	7	10
	8	8	10
	10	7	9
MAX-R2L	2	7	8
	4	7	11
	6	7	10
	8	8	10
	10	7	9

5.3. 57 Bus Power System

The best solution found by A* heuristics for the 57 bus system was 12 PMUs, as seen in Table 3. The depth method ($d = 1$) for the A* heuristic performed the best, reaching the optimal solution in 0.3673 seconds as seen in Table 4. The A* heuristic,

in 675.01 seconds, also found a near-optimal number of 12 PMUs. Note that as the bus system sizes increase, the Double Vertex heuristic starts to perform worse than most of the other heuristics presented in this paper. Figure 15 reports the results of Greedy, Single and Double Vertex algorithm executions on the 57 Bus Power System.

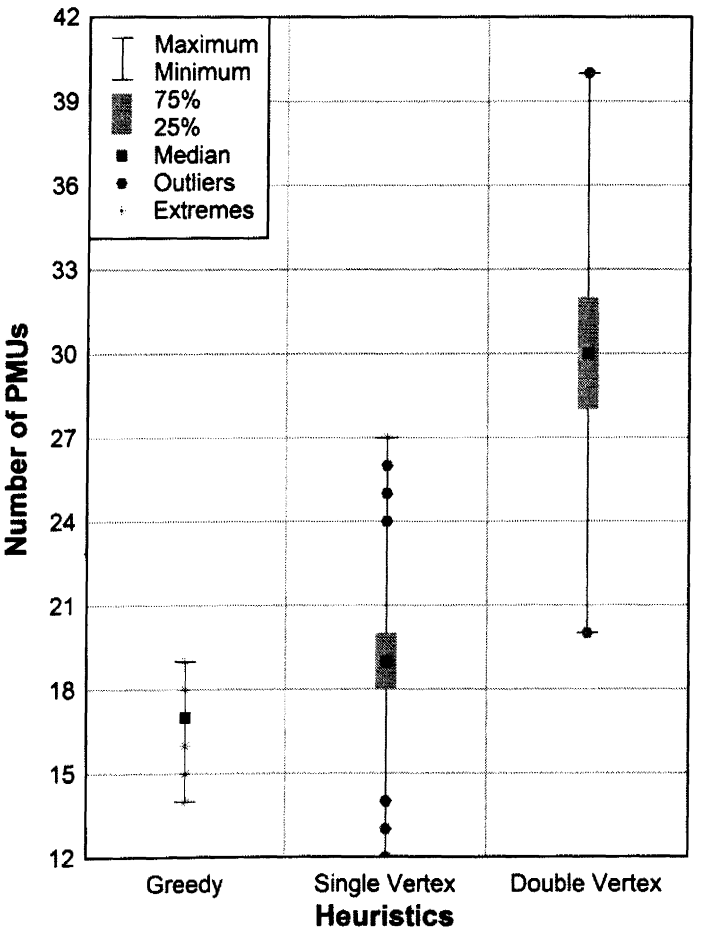


Figure 15: IEEE standard 57 bus results.

The parallel A* heuristic found an optimal solution of 12 PMUs as seen in Table 6. The run-time for two processors was 755.40 seconds, and 222.21 seconds for ten processors as seen in Table 7. It is observed from Figure 13 that, as compared to results from parallel A* executions of IEEE 14, 30 bus system, the parallel A* communication-computation ratios for the IEEE 57 bus system was better on

executions using two or three processors.

The sequential DLA found an optimal solution of eleven PMUs for the 57 bus system while maintaining a low margin of error, as seen in Table 13. *Seqla* recorded a run-time of 3.04 seconds and DLA heuristic *MAX-C* had the lowest margin of error, assigning a maximum of 13 PMUs during *Seqla*'s execution.

Table 13: Sequential DLA IEEE standard 57 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	11	13
MAXSEQ-CPR	11	14
MAX-R1L	11	16
MAX-R2L	11	14

The *Parla* found an optimal solution of ten PMUs for the 57 bus system, as shown in Table 14. The error margin was roughly equivalent for executions utilizing two, four, six, eight or ten processors. DLA heuristic *MAX-R1L* performed best in assigning the minimal optimal number of PMUs, on four processors during *Parla*'s execution. Of all algorithms presented in this paper, only *Parla* found the absolute minimum number of PMUs for the IEEE 57 bus system.

5.4. 118 Bus Power System

The best solution reported by A* heuristics for the 118 bus system was 29 PMUs, as seen in Table 3. The depth method performed fastest, finding a solution in 2.0369 seconds as seen in Table 4. The 118 Bus case starts to demonstrate the effects of the heuristics on a larger-scaled system. The A* depth method $d = 2$ is not able to find the best solution but with $d = 1$. The A* heuristic was unable to return a solution as seen in Table 3. As compared to the other heuristics presented in this paper, the Double Vertex heuristic showed the largest differential from the best

Table 14: Parallel DLA IEEE standard 57 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	11	13
	4	11	15
	6	11	15
	8	12	17
	10	15	19
MAXSEQ-CPR	2	11	15
	4	12	16
	6	12	16
	8	11	17
	10	15	20
MAX-R1L	2	11	15
	4	10	20
	6	13	16
	8	13	17
	10	15	20
MAX-R2L	2	11	13
	4	12	16
	6	11	15
	8	12	17
	10	15	20

solution found in the previous bus systems. Figure 16 reports the results of Greedy, Single and Double Vertex algorithm executions on the 118 Bus Power System.

The parallel A* heuristic found a near-optimal solution of 29 PMUs as seen in Table 6. The run-time for five processors was 330.54 seconds, and 1292.82 seconds for ten processors as seen in Table 7. It is observed from Figure 13 that, as compared to results from parallel A* executions of IEEE 14, 30, 57 bus system, the parallel A* communication-computation ratios for the IEEE 118 bus system were better for executions utilizing more processors.

The sequential DLA found an optimal solution of 29 PMUs for the 118 bus system while maintaining a low margin of error, as seen in Table 15. DLA heuristics *MAX-C* and *MAX-R2L* had the lowest margins of error, assigning a minimal of 29

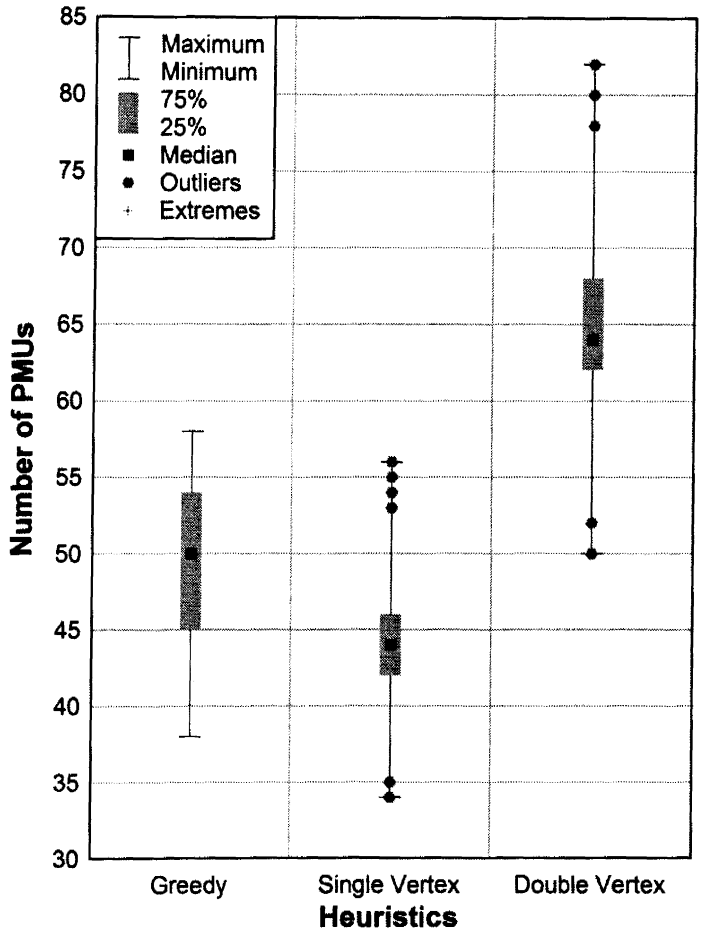


Figure 16: IEEE standard 118 bus results.

PMUs. *Seqla* had a run-time of 18.23 seconds, as shown in Table 10.

Table 15: Sequential DLA IEEE standard 118 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	29	31
MAXSEQ-CPR	30	34
MAX-R1L	30	37
MAX-R2L	29	32

Parla found an optimal solution of 28 PMUs for the 118 bus system, as shown in Table 16. For executions utilizing the different number of processors the error margin

was roughly equivalent while DLA heuristic *MAX-C* performed best in assigning the minimal number of PMUs. A run-time of 13.46 seconds was reported for *Parla*'s execution in Table 10. We see that *Parla* and *Seqla* emerge as the best of the all the algorithms presented, with respect to solution quality.

Table 16: Parallel DLA IEEE standard 118 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	30	32
	4	28	35
	6	30	34
	8	28	34
	10	30	39
MAXSEQ-CPR	2	31	33
	4	31	36
	6	31	37
	8	28	36
	10	33	39
MAX-R1L	2	32	34
	4	31	36
	6	31	36
	8	30	37
	10	34	39
MAX-R2L	2	30	33
	4	28	35
	6	30	34
	8	29	36
	10	34	40

5.5. 300 Bus Power System

The best solution found by the A* heuristics for the 300 bus system was 109 PMUs. The Single Vertex heuristic reached a solution of 107 PMUs. The time spent per trial seen in Table 5 for the non-deterministic heuristics may suggest that these heuristics out-perform the pruning methods for the A* heuristic. However, when multiple trials are taken into account, the A* pruning methods were able to return a

result over 150 times faster. The results of the A* depth and restricted list methods seen in Table 5 were 110 and 109 PMUs, respectively. Figure 17 shows that the Double Vertex heuristic performed the worst among the heuristics that completed the trial. The best solution of the Double Vertex Heuristic placed 40 more PMUs compared to the best solution identified by the Single Vertex heuristic.

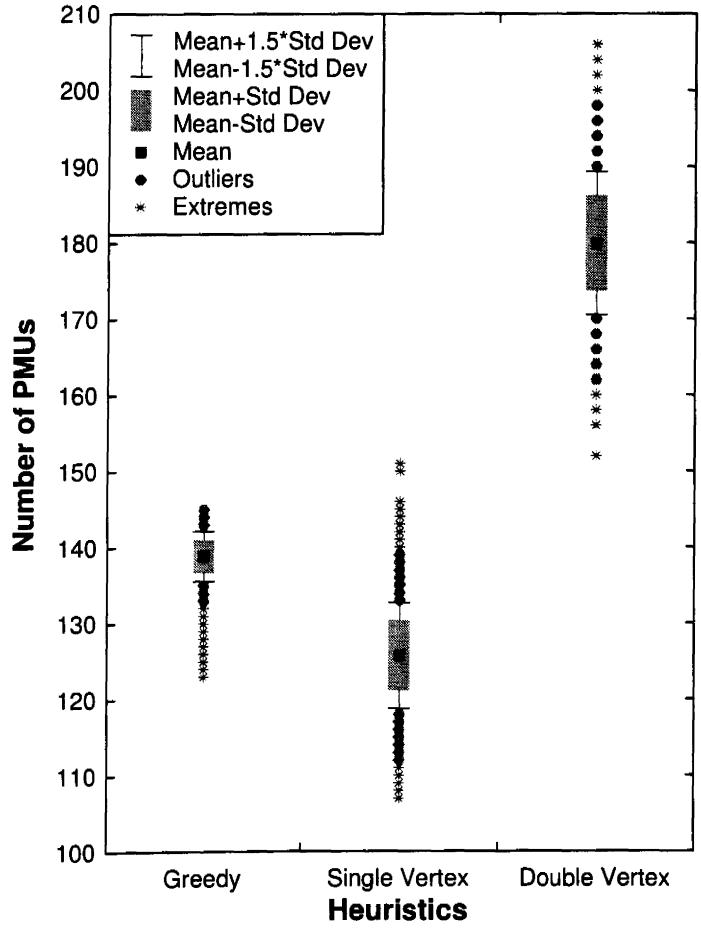


Figure 17: IEEE standard 300 bus results.

The parallel A* heuristic found an optimal solution of 112 PMUs as seen in Table 6. The run-time for five processors was 5612.88 seconds (1.588 hours), and 20541.12 seconds (5.705 hours) for ten processors as seen in Table 7. It is observed from Figure 13 that the parallel A* communication-computation ratio for the IEEE

300 bus system is greater for executions utilizing fewer processors.

The sequential DLA found an optimal solution of 88 PMUs for the 300 bus system while maintaining a relatively low margin of error, as seen in Table 17. DLA heuristics *MAX-C* and *MAX-R2L* had the lowest margins of error, assigning a minimal of 88 PMUs. *Seqla* had a run-time of 219.05 seconds, as shown in Table 10.

Table 17: Sequential DLA IEEE standard 300 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	88	91
MAXSEQ-CPR	90	94
MAX-R1L	91	102
MAX-R2L	88	93

Parla found an optimal solution of 89 PMUs for the 300 bus system, as shown in Table 18. For *Parla* executions utilizing two, four, six, eight, or ten processors, the error margin was roughly equivalent while DLA heuristic *MAX-C* performed best in assigning the minimal number of PMUs. A run-time of 127.58 seconds was reported for *Parla*'s execution in Table 10. Again, we see that *Parla* and *Seqla* perform better, amongst all the algorithms presented, with respect to solution quality.

5.6. 2,383 Bus Power System

Figure 18 shows that the Single Vertex heuristic was able to reach a solution of 1025 PMUs. This solution is 21 PMUs fewer than the best A* optimization of 1046 PMUs, as observed in Table 3. Because the 2,383 bus power system is a large-scale system, we were only able to run 1,000 trials each for the Greedy, Single Vertex, and Double Vertex heuristics. The previous trials of the 14, 30, 57, 118, and 300 bus systems were all executed 100,000 times. We hypothesize that if more trials were run on the 2,383 bus system, better solutions may have been found.

Table 18: Parallel DLA IEEE standard 300 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	89	89
	4	91	94
	6	91	95
	8	94	95
	10	92	97
MAXSEQ-CPR	2	91	92
	4	96	99
	6	96	97
	8	95	100
	10	93	98
MAX-R1L	2	98	102
	4	99	103
	6	96	103
	8	95	101
	10	93	106
MAX-R2L	2	89	90
	4	94	95
	6	92	97
	8	92	97
	10	92	100

The sequential DLA found an optimal solution of 756 PMUs for the 2383 bus system as seen in Table 19. DLA heuristics *MAX-C* and *MAXSEQ-CPR* had the lowest margins of error, assigning a minimal of 756 PMUs. *Seqla* had a run-time of 105.56 seconds, as shown in Table 10.

Table 19: Sequential DLA 2383 bus results

DLA Heuristic	Number of PMUs	
	Minimum	Maximum
MAX-C	756	767
MAXSEQ-CPR	786	804
MAX-R1L	917	1000
MAX-R2L	775	825

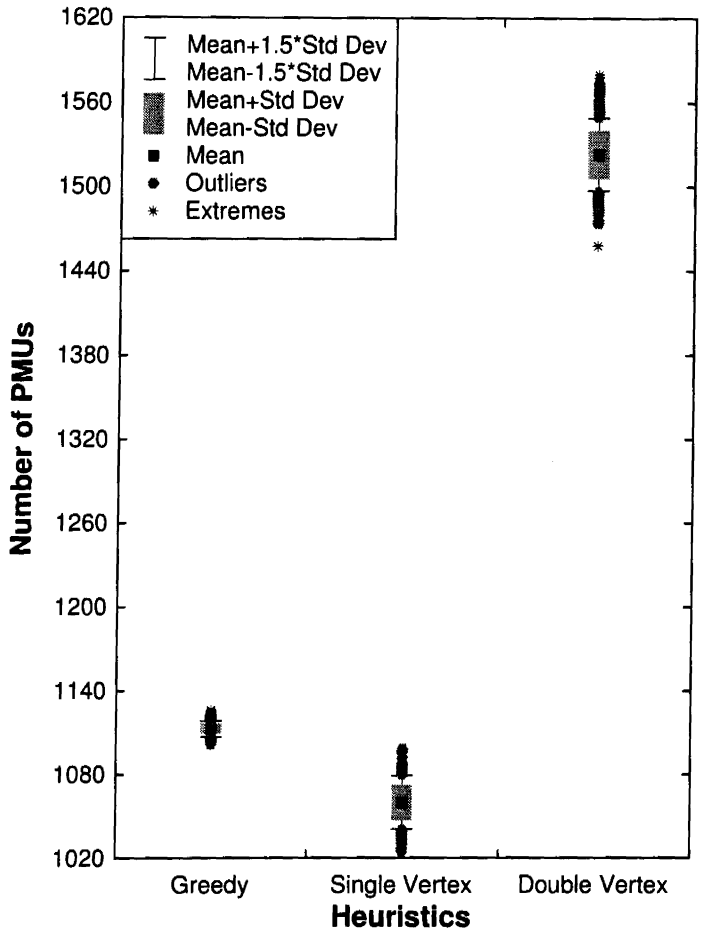


Figure 18: 2383 bus results.

Parla found an optimal solution of 740 PMUs for the 2383 bus system, as shown in Table 20. DLA heuristic *MAX-C* performed best in assigning the minimal number of PMUs, when utilizing eight processors. A run-time of 66.13 seconds was reported for *Parla*'s execution in Table 10. Again, we see that *Parla* and *Seqla* perform better, amongst all the algorithms presented, with respect to solution quality.

Table 21 summarizes our proposed algorithms with other published algorithms. As can be clearly seen, the A*, *Seqla*, *Parla* algorithms are as optimal as other algorithms. Run-times could not be compared since not all published algorithm run-times were available. Nevertheless, we are convinced — following our simulation

Table 20: Parallel DLA 2383 bus results

DLA Heuristic	Number of Processors	Number of PMUs	
		Minimum	Maximum
MAX-C	2	769	786
	4	781	814
	6	783	804
	8	740	816
	10	756	864
MAXSEQ-CPR	2	812	856
	4	806	847
	6	820	877
	8	777	877
	10	803	901
MAX-R1L	2	881	956
	4	846	919
	6	844	971
	8	757	923
	10	819	902
MAX-R2L	2	791	824
	4	803	828
	6	810	861
	8	766	871
	10	796	875

results — that our newly proposed distance level algorithms are comparable, if not more optimal in execution times.

Table 21: Comparison with other PMU placement methods

IEEE Test System	Integer Programming [2]	Simulated Annealing [5]	Immunity Genetic Algorithm [16]	A Proposed Model [27]	Proposed Algorithms		
					A*	*Seqla*	*Parla*
14-Bus	3	3	3	3	3	3	3
30-Bus	-	7	7	7	7	7	7
57-Bus	12	11	11	11	12	11	10
118-Bus	29	-	28	28	29	29	28

To conclude our simulation findings, we suggest that if the given bus system is

small, then the A* heuristic with the pruning methods presented would be the ideal for solving the problem. On the other hand, for large bus systems of 118 buses or greater, we recommend the utilizing sequential distance level algorithm (*Seqla*) or the parallel distance level algorithm (*Parla*) with the *MAX-C* DLA heuristic. For Parallel A* executions, the communication-computation ratio should be limited to 0.6. Communication-computation ratios greater than 0.6 mean the utilized processors will process less computational tasks, and more communication tasks than desired.

CHAPTER 6. CONCLUSION

In this paper, we tackled the PMU placement problem. We proposed heuristics varying from a Greedy heuristic to an informed search A* heuristic. The DLA, a novel family of algorithms was presented. Parallel algorithms, parallel A* and parallel DLA were introduced as multi-processor algorithms useful in solving the PMU placement problem. Experimental results for all algorithms introduced in this paper were presented. Based on the results presented, we concluded that for small-scale power systems the A* heuristic with the proposed pruning methods would be the ideal choice for solving the PMU placement problem. For large-scale power systems, the novel sequential or parallel distance-level algorithms presented were most practical.

REFERENCES

[1] N. H. Abbasy and H. M. Ismail, "A Unified Approach for the Optimal PMU Location for Power System State Estimation," *IEEE Transactions on Power Systems*, Vol. 24, No. 2, pp. 806–813, May 2009.

[2] A. Abur and B. Xu, "Observability Analysis and Measurement Placement for Systems with PMUs," *IEEE Power Systems Conference and Exposition (PES '04)*, Oct. 2004, pp. 943–946.

[3] A. Ali, *Optimal Placement of Phasor Measurement Units*, Technical Report PSERC–05-58, Electrical and Computer Engineering Department, Texas A and M University, Oct. 2005, 13 pp.

[4] J. R. Altman, *A Practical Comprehensive Approach to PMU Placement for Full Observability*, Master's Thesis, Electrical and Computer Engineering Department, Virginia Polytechnic Institute and State University, Jan. 2007, 64 pp.

[5] T. L. Baldwin, L. Mili, M. B. Boisen, and R. Adapa, "Power System Observability With Minimal Phasor Measurement Placement," *IEEE Transactions on Power Systems*, Vol. 8, No. 2, pp. 707–715, May 1993.

[6] S. Chakrabarti and E. Kyriakides, " Optimal Placement of Phasor Measurement Units for Power System Observability," *IEEE Transactions on Power Systems*, Vol. 23, No. 3, pp. 1433–1440, Aug. 2008.

[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, NY, 2001.

[8] M. Kafil and I. Ahmad, *Optimal Task Assignment in Heterogeneous Distributed Computing Systems*, IEEE Concurrency, July-Sept. 1998, pp. 42-51.

[9] R. M. Karp, "On the complexity of Combinatorial Problems," Journal of Networks, Vol. 5, No. 1, pp. 45–68, 1975.

[10] S. U. Khan and I. Ahmad, "Comparison and Analysis of Ten Static Heuristics-based Internet Data Replication Techniques," *Journal of Parallel and Distributed Computing*, vol. 68, no. 2, pp. 113-136, 2008.

[11] R. F. Nuqui, *State Estimation and Voltage Security Monitoring Using Synchronized Phasor Measurements*, Ph.D. Dissertation, Electrical and Computer Engineering Department, Virginia Polytechnic Institute and State University, Jan. 2001, 213 pp.

[12] S. Sahni and T. Gonzales, "P-complete Approximation Problems," *Journal of the Association of Computing Machines*, Vol. 23, No. 3, pp. 555–565, 1976.

[13] V. V. Vazirani, *Approximation Algorithms*, Springer, Berlin, 2001.

[14] Power Systems Test Case Archive - UWEE, University of Washington. Available at: http://www.ee.washington.edu/research/pstca/.

[15] R. Zivanovic and C. Cairns, "Implementation of PMU technology in state estimation: An overview" in *Proc. IEEE AFRICON,* vol. 2, pp. 1006Â–1011, Sep. 1996.

[16] F. Aminifar, C. Lucas, A. Khodaei, M. Fotuhi-Firuzabad, "Optimal placement of phasor measurement units using immunity genetic algorithm," *IEEE Transactions on Power Delivery*, vol. 24, No. 3, pp.1014–1020, July. 2009.

[17] B. Milosevic and M. Begovic, "Nondominated sorting genetic algorithm for optimal phasor measurement placement," *IEEE Trans. Power Syst.*, vol. 18, no. 1, pp. 69–75, Feb. 2003.

[18] B. Gou, "Generalized integer linear programming formulation for optimal PMU placement," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 1099–1104, Aug. 2008.

[19] R. F. Nuqui and A. G. Phadke, "Phasor measurement unit placement techniques for complete and incomplete observability," *IEEE Transactions on Power Delivery*, vol. 20, No. 4, pp.2381–2388, Oct. 2005.

[20] B. Gou, "Optimal placement of PMUs by integer linear programming," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 1525–1526, Aug. 2008.

[21] C. Rakpenthai, S. Premrudeepreechacharn, Sermsak Uatrongjit, and N. R. Watson, "An optimal PMU placement method against measurement loss and branch outage," *IEEE Transactions on Power Delivery*, vol. 22, No. 4, pp.101–107, Jan. 2007.

[22] J. Chen and A. Abur, "Placement of PMUs to enable bad data detection in state estimation," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1608–1615, Nov. 2006.

[23] D. N. Kosterev, J. Esztergalyos, and C. A. Stigers, "Feasibility study of using synchronized phasor measurements for generator dropping controls in the colstrip system," *IEEE Trans. Power Syst.*, vol. 13, no. 3, pp. 755–761, Aug. 1998.

[24] Z. Zhong, C. Xu, B. J. Billian, L. Zhang, S. J. S. Tsai, R. W. Conners, V. A. Centeno, A. G. Phadke, and Y. Liu, "Power system frequency monitoring network (FNET) implementation," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1914–1921, Nov. 2005.

[25] G. T. Heydt, C. C. Liu, A. G. Phadke, and V. Vital,"Solutions for the crisis in electric power supply," *IEEE Comput. Appl. Power Mag.*, vol. 14, no. 3, pp. 22–30, Jul. 2001.

[26] G. Chartrand and L. Lesniak "Graphs Diagraphs," Chapman Hall/CRC, 2005.

[27] F. Amnifar, A. Khodaei, M. Fotuhi-Firuzabad, and M. Shahidehpour, "Contingency-Constrained PMU Placement in Power Networks," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 516–523, Feb. 2010.