Optimization Using Method of Exhaustive Search, Greedy Search and MATLAB's solver

John Michael P. Corbeta*, Abdul Aziz G. Mabaning†
Department of Electrical Engineering, College of Engineering and Technology
Mindanao State University - Iligan Institute of Technology
Iligan City, Philippines
*johncorbeta1997@gmail.com, †mraaguevarra@gmail.com

***Abstract*—There are many fields in power systems that requires optimizations. Power system optimization ranges from continuous problems like economice dispatch to complex decision making of unit commitment. This paper discusses the pros and cons of the following optimization methods: exhaustive search, greedy search, and MATLAB/Octave solver. These methods will be implemented into a optimal placement problem for phasor measurement units (PMUs) in IEEE test sytems for power system estimation.**

***Index Terms*—Power system optimization, matlab, octave, exhaustive search, greedy algorithm**

I. Introduction

Monitoring the state of the power system is one of the important applications of power system optimization. This provides accurate understanding and data of the current state of power system operation. This requires measurement datas from measuring and communication devices located all throughout the power system. This devices, remote terminal units (RTUs) and PMUs provide measurement datas of their respective buses to the control center. PMUs are devices which uses synchronization signals from global positional system (GPS) satellites and provide positive sequence phasor voltages and currents measured at a given substation improving the performance of state estimators. This study refers to the pioneering work in optimal PMU placement (OPP) by Phadke et al. [1], [2] and its relaxed constraints to a simplified state estimator in [3]. This problem is formulated and solved using Octave's linear programming solver, exhaustive/brute search, and greedy algorithm.

GNU Octave is a software featuring high-level programming language primarily intended for numerical computation. Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. On the other hand, brute-force search or exhaustive search, also known as generate and test, is a very general problem-solving technique and algorithmic paradigm that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem's statement. While a brute-force search is simple to implement and will always find a solution if it exists, implementation costs are proportional to the number of candidate solutions, which in many practical problems tends to grow very quickly as the size of the problem increases [4]. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed. Greedy algorithm however, is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. Greedy heuristics are known to produce suboptimal results on many problems, [5] and so natural questions are:

- For which problems do greedy algorithms perform optimally?
- For which problems do greedy algorithms guarantee an approximately optimal solution?
- For which problems are the greedy algorithm guaranteed not to produce an optimal solution?

A large body of literature exists answering these questions for general classes of problems, such as matroids [6], as well as for specific problems [7], such as set cover.

This paper discusses the effectiveness of these methods in solving an optimial PMU place ment problem. An IEEE 7-bus system will be used in analyzing the constraints of the optimization problem. The methods will then be implemented into larger IEEE test systems.

II. Problem Formulation

A PMU placed at a given bus is capable of measuring the voltage phasor of the bus as well as the phasor currents for all lines incident to that bus. Thus, the entire system can be made observable by placing PMUs at strategic buses in the system. The objective of the OPP problem is to accomplish this task by using a minimum number of PMUs.

For an n-bus system, assuming an equal cost of all PMU installation, the cost of PMU placement would be irrelevant for the optimization problem. Thus only the number of PMUs is considered, the PMU placement problem can be formulated as follows:

$$\min \;\; c(\mathbf{x}) = \sum_{i=1}^{n} x_i \tag{1}$$

$$\text{s.t.} \;\; \mathbf{f}(\mathbf{x}) \geq \hat{1} \tag{2}$$

where $\mathbf{x}$ is a binary decision variable vector, whose entries are defined as:

$$x_i = \begin{cases} 1 & \text{is a PMU is installed in bus } i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$\mathbf{f(x)}$ is a vector function whose entries are non-zero if the corresponding bus voltage is solvable using the given measurement set and zero otherwise while $\hat{1}$ is a vector whose entries are all ones. Consider the 7-bus system and its measurement configuration shown in Figure. First, form the

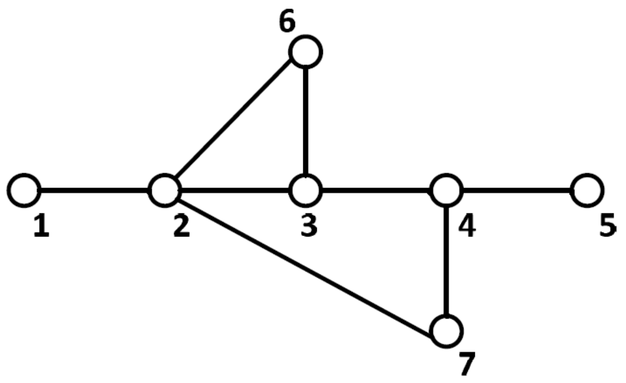

Fig. 1. IEEE 7-bus system

binary connectivity matrix A. The entries of A are defined as follows:

$$A_{j,k} = \begin{cases} 1 & \text{if } Y_{j,k} \neq 0 \\ 0 & \text{if otherwise} \end{cases} \tag{4}$$

Matrix A can be directly obtained from the bus admittance matrix by transforming its entries into binary form. Building the A matrix for the 7-bus system of Figure yields:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The constraints for this case can be formed as the product of matrix $\mathbf{A}$ and solution vector $\mathbf{x}$ in (6).

$$\mathbf{f(x)} = \mathbf{A} \cdot \mathbf{x} \geq \hat{1} \tag{6}$$

A. Exhaustive/Brute Search

In this subsection, the framework for exhaustive/brute search is designed. Exhaustive search has the computation time in solving optimization problem. It collects all possible solution in finding the optimal objective value. For an OPP problem of an n bus system, with each x_i having the two possible value as described in (3), the number of possible solution is as follows:

$$m = 2^n - 1 \tag{7}$$

Each solution k is then evaluated to the current optimal solution $\hat{\mathbf{x}}$ where:

$$c(\mathbf{x}^{(k)}) < c(\hat{\mathbf{x}}) \rightarrow \mathbf{x}^{(k)} \text{ is optimal} \tag{8}$$

$$c(\mathbf{x}^{(k)}) = c(\hat{\mathbf{x}}) \rightarrow \mathbf{x}^{(k)} \text{ is cooptimal} \tag{9}$$

$$c(\mathbf{x}^{(k)}) > c(\hat{\mathbf{x}}) \rightarrow \mathbf{x}^{(k)} \text{ is not optimal} \tag{10}$$

In (8-10), brute search algorithm for OPP problem can obtain more than one solution, cooptimal solutions. The algorithm can be modified to record all the possible solution and respected value by eliminating the condition of optimality and sorting the record of solutions. Looking for the next optimal solution from the previously found solution adds complexity to the algorithm. In order to do so , we must add the solution to the constraints as shown in (11) and (12). The algorithm for the brute search algorithm is depicted in Fig. 2.

$$\mathbf{A}_{new} = \begin{bmatrix} \mathbf{A}_{old} \\ 1 - \hat{\mathbf{x}} \end{bmatrix} \tag{11}$$

$$\mathbf{f(x)}_{new} = \mathbf{A}_{new} \cdot \mathbf{x} \geq \hat{1} \tag{12}$$

The importance in obtaining cooptimal and sub-optimal solution is to have viable options in case we are restricted in using the global optimum for some reasons like geological or economical.

B. Greedy Algorithm

The greedy algorithm is a multi-stage optimization that involves decision making of selecting the optimum value of each variable to obtain the overall optimum of the system. Considering Fig. 1, in order to obtain the minimum number of PMUs to be installed, each PMU must observe as much buses as it can. Thus the single stage process of the greedy method is to select the PMU that can observe the maximum buses. Now the buses to be observed are the constraints of the OPP, therefore each stage maximizes the constraints covered by $\hat{x_i} = 1$. We must first initialize our solution as $\hat{\mathbf{x}} = \hat{0}$. Looking closely at (5), the columns represent each sample variable $\hat{x_i}$. For the first stage, the sum of the elements of the column is the number of constraints covered by each variable given by (13).

$$\text{sum}(\mathbf{A}, \downarrow) = \begin{bmatrix} 2 & 5 & 4 & 4 & 2 & 3 & 3 \end{bmatrix} \tag{13}$$

From (13), 5 at column 2 is the maximum value, thus $\hat{x_2} = 1$. For stage 2, we repeat stage 1 but exclude $\hat{x_2}$ from the process. This time we need to create a filter vector to multiply to the rows $\mathbf{A}$ matrix. This filter vector $\mathbf{g}$ can be derived as follows:

$$\mathbf{g} = 1 - \hat{\mathbf{x}} \tag{14}$$

Filter vector from (14) guarantees that the sum for column 2 is zero. The sum for stage 2 is as follows:

$$\text{sum}(\mathbf{A}, \downarrow) = \begin{bmatrix} 0 & 0 & 1 & 2 & 2 & 0 & 1 \end{bmatrix} \tag{15}$$

In this case either $\hat{x}_4$ of $\hat{x}_5$ is selected. This process is repeated until the all m constraints are covered. Searching for other solutions is also possible with (11). This algorithm is depicted in Fig. 3.

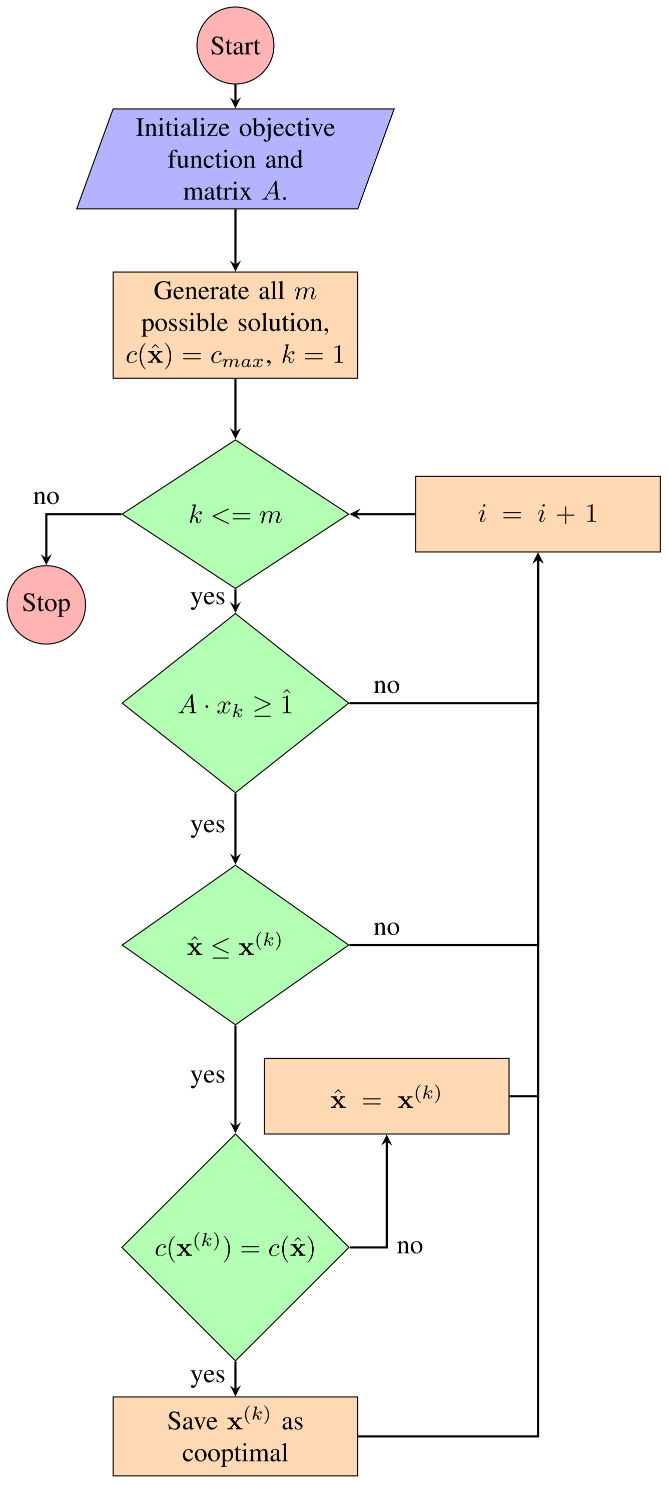
Start
Initialize objective function and matrix A.
Generate all m possible solution, $c(\hat{\mathbf{x}}) = c_{max}$, $k = 1$
$k <= m$
no
Stop
yes
$i = i + 1$
$A \cdot x_k \geq \hat{1}$
no
yes
$\hat{\mathbf{x}} \leq \mathbf{x}^{(k)}$
no
yes
$\hat{\mathbf{x}} = \mathbf{x}^{(k)}$
$c(\mathbf{x}^{(k)}) = c(\hat{\mathbf{x}})$
no
yes
Save $\mathbf{x}^{(k)}$ as cooptimal

Fig. 2. Brute force algorithm for optimal PMU placement

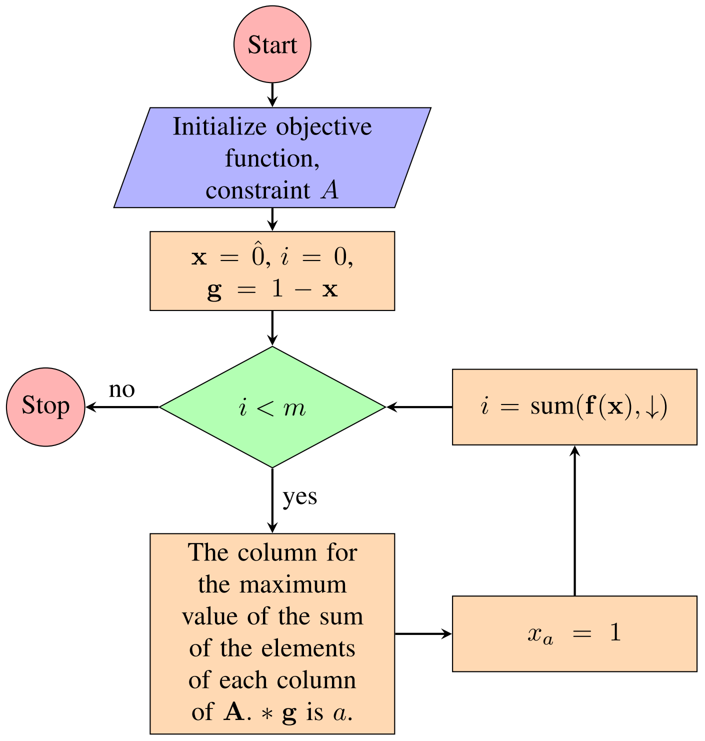
Start
Initialize objective function, constraint A
$\mathbf{x} = \hat{0}$, $i = 0$, $\mathbf{g} = 1 - \mathbf{x}$
$i < m$
no
Stop
$i = \text{sum}(\mathbf{f}(\mathbf{x}), \downarrow)$
yes
The column for the maximum value of the sum of the elements of each column of $\mathbf{A}. * \mathbf{g}$ is a.
$x_a = 1$

Fig. 3. Greedy algorithm for optimal PMU placement

C. *Octave glpk Function*

The function glpk is used to solve Linear Programming (LP) problems of the type min $\mathbf{c}^T\mathbf{x}$ subject to the linear constraints $\mathbf{Ax} = \mathbf{b}$ where $x \geq 0$ and its variations, namely, maximize instead of minimize, and inequality constraints or mixed constraints instead of equality constraints.

III. Simulation Results

The following method in section 2 is tested to IEEE 7, 14, 57 and 118 bus test systems using Octave GNU with the hardware in Table I with the following resuts.

TABLE I
Hardware Specification

CPU	LEVEL 2 CACHE	SYSTEM MEMORY
AMD RyzenTM 3 2200U	5.0 MB	4 GB

Test results are of the 4 IEEE bus systems is in Table II, III, IV and V. It is observed that all methods achieves global optimum for IEEE 7-bus system with the greedy algorithm fastest while exhaustive search comparably slowest. For the 14-bus however, the fastest greedy algorithm can only obtain sub-optimal value very close to the global optimum obtained by the rest. The computational burden of exhaustive search is starting to show a huge difference in speed. For the 57-bus system, exhaustive search seems to be too much of an effort for the hardware while greedy algorithm continues to be the fastest and with a very convincing sub-optimal solution.

Finally for the 118-bus system, glpk has finally exceeded the greedy algorithm in speed and more effective solution.

TABLE II
IEEE 7-BUS SIMULATION

Method	# of PMU	Time
Octave glpk	2	0.00165606 s
Exhaustive	2	0.020926 s
Greedy	2	0.000154018 s

TABLE III
IEEE 14-BUS SIMULATION

Method	# of PMU	Time
Octave glpk	4	0.00138807 s
Exhaustive	4	2.58746 s
Greedy	5	0.000287056 s

TABLE IV
IEEE 57-BUS SIMULATION

Method	# of PMU	Time
Octave glpk	16	0.00223398 s
Exhaustive	N/A	OUT OF MEMORY
Greedy	18	0.00127387 s

TABLE V
IEEE 118-BUS SIMULATION

Method	# of PMU	Time
Octave glpk	32	0.00271201 s
Exhaustive	N/A	OUT OF MEMORY
Greedy	36	0.00635505 s

Table VI shows the effectiveness of the three algorithms in excluding previously found solution. The glpk function seems to be restricted to the previously found solution while greedy algorithm has the possibility of finding more optimal solution.

IV. CONCLUSION

Exhaustive or brute search provides the best solution but its simple approach draws the most computational time and resources. Greedy algorithm covers the cons of the brute search in return for a less optimal solution as the size of the problem increases. Integer linear programming seems to be the most efficient method in obtaining optimal solution with few computational resources. However, problems with more constraints than the input size of the problem, such as finding the next optimal solution excluding the previously obtained solution, glpk function seems to be restricted to non-integer solutions having the same optimal value. While, greedy algorithm however seems to be able to find better solutions than it previously found. Thus the paper effectively compares and sort out the advantages of the three methods.

There is a variety of integer linear programming. Further research could study each unique variety of integer linear programming and its advantages to one another. Also, glpk is an integer linear programming while MATLAB fmincon is a gradient optimization involving continuous variable and hessian matrices.

TABLE VI
ALTERNATIVE SOLUTIONS FOUND EXCLUDING THE OBTAINED OPTIMAL SOLUTION.

Method	IEEE-7	IEEE-14	IEEE-57	IEEE-118
Octave glpk	2	4	16	32
Exhaustive	2-7	4-14	16-57	32-118
Greedy	2-7	4-14	18...	35...

REFERENCES

[1] A. G. Phadke, J. S. Thorp, and K. J. Karimi, "State estimation with phasor measurements," vol. PER-6, no. 2, pp. 48–48, feb 1986.

[2] A. Phadke, "Synchronized phasor measurements in power systems," vol. 6, no. 2, pp. 10–15, apr 1993.

[3] T. Baldwin, L. Mili, M. Boisen, and R. Adapa, "Power system observability with minimal phasor measurement placement," vol. 8, no. 2, pp. 707–715, may 1993.

[4] C. G. Emily Fox. Complexity of brute force search. [Online]. Available: https://www.coursera.org/lecture/ml-clustering-and-retrieval/complexity-of-brute-force-search-5R6q3

[5] U. Feige, "A threshold of ln n for approximating set cover," vol. 45, no. 4, pp. 634–652, jul 1998.

[6] H. Nishimura and S. Kuroda, Eds., *A Lost Mathematician, Takeo Nakasawa*. Birkhäuser Basel, 2009.

[7] V. Chvatal, "A greedy heuristic for the set-covering problem," vol. 4, no. 3, pp. 233–235, aug 1979.

$$\text{OR} \tag{16}$$

$$\sum_{i=1}^{n} x_i \leq ny \tag{17}$$

$$\sum_{i=1}^{n} x_i \geq y \tag{18}$$

$$\text{AND} \tag{19}$$

$$\sum_{i=1}^{n} x_i \geq ny \tag{20}$$

$$\sum_{i=1}^{n} x_i - (n-1) \leq y \tag{21}$$

$$\text{XOR} \tag{22}$$
$$a = [0,1]\text{binary} \tag{23}$$
$$b = [0,1]\text{binary} \tag{24}$$
$$\sum_{i=1}^{n} x_i \leq na \tag{25}$$
$$\sum_{i=1}^{n} x_i \geq a \tag{26}$$
$$\sum_{i=1}^{n} x_i \geq nb \tag{27}$$
$$\sum_{i=1}^{n} x_i - (n-1) \leq b \tag{28}$$
$$a - b = y \tag{29}$$

$$\text{XAND} \tag{30}$$
$$a = [0,1]\text{binary} \tag{31}$$
$$b = [0,1]\text{binary} \tag{32}$$
$$\sum_{i=1}^{n} x_i \leq na \tag{33}$$
$$\sum_{i=1}^{n} x_i \geq a \tag{34}$$
$$\sum_{i=1}^{n} x_i \geq nb \tag{35}$$
$$\sum_{i=1}^{n} x_i - (n-1) \leq b \tag{36}$$
$$1 - (a - b) = y \tag{37}$$

$$y = (\overline{A} \cdot B) + (A \cdot \overline{B}) \tag{38}$$
$$y = (A + B) \cdot (\overline{A} + \overline{B}) \tag{39}$$
$$y = (A + B) \cdot \overline{(A \cdot B)} \tag{40}$$

$$y = (A \cdot B) + (\overline{A} \cdot \overline{B}) \tag{41}$$
$$y = (A \cdot B) + \overline{(A + B)} \tag{42}$$