A COMPUTER PACKAGE FOR DETERMINING THE REAL-TIME NETWORK OBSERVABILITY

Mahmoud A.EL-Gammal, Nagui Sorial and Ahmed Abdel Razek
Department of Electrical Engineering, Faculty of Engineering
Alexandria University, Alexandria, Egypt.

ABSTRACT

The use of modern digital computer systems in power system control centers has been made it possible to track the changing conditions in the network with a real-time model in the computer. Since the availability of real-time measurements can change because of failures in the telemetering equipment, an observability check is usually made before the state estimator solution of the network is executed. In this paper, the software of a computer package for determining the topologically-based real-time network observability is developed. The treatment of unobservability conditions of power networks are also presented. The computer package programs are tested by using different real-time data of the IEEE 14-bus power system model.

Keywords: Computer-aided real-time modelling, Observability, Network topology, Computer center of power system.

1. INTRODUCTION

Observability analysis is a fundamental component of any real-time power system modelling and control [1]. Efficient implementation of observability is necessary in order to achieve the overall requirements for state estimation in modern control centers.

The following questions emerge naturally in conjunction with state estimation in system operation:

i- Are there enough real-time measurements to make state estimation possible?

ii- If not, which part or parts of the network whose states can still be estimated with the available measurements?

iii- How to select additional pseudo-measurements to be included in the measurement set to make state estimation possible?

The analysis which leads to an answer for these questions is called ***Observability test***. The analysis includes observability check and treatment of unobservability.

Mathematically, the question of observability is related to the rank of the Jacobian matrix of the nonlinear function $\mathbf{f}(\mathbf{x})$ that relates the measurement vector to the state vector [2]. The network is observable if the Jacobian matrix is of full rank. However, a floating point calculation of the rank of a large dimensional matrix is, at best, time consuming and, at worst, dependent upon the condition of the matrix under consideration. Furthermore after such calculations, no information is provided about the location of the network trouble. Therefore, the question of observability can be related to the network topology and thus motivates a topologically based observability algorithm. The algorithm provides a tool which can be used to resolve the following questions:

what parts of the network are observable ? ; and what measurement should be added to render the entire network observable ? [3].

Krumpholz, Clements, and Davis [4-6] published a graph-theoretic observability algorithm. The algorithm is based on a fundamental theorem which states that the necessary and sufficient condition for a network to be observable is that it contains at least one observable spanning tree. This paper describes the software package programs for determining network observability using telemetered measurements. The treatment of accidental unobservability due to unanticipated network topology changes or failures in the telecommunication systems are also presented. The

computer package programs has been tested by using the IEEE 14-bus power system.

2. SOFTWARE PACKAGE DEVELOPMENT

This section presents a complete description of the software for the topologically based observability and measurement placement determination of a real-time power system model using telemetered measurements. The method is based on the graph-theoretic observability algorithm of Clements et.al [4-6]

A software package for the observability algorithm, as indicated in the flowcharts of Figures (1a) & (1b), may be decomposed into two procedures : the first *builds the flow islands* and the second *processes the boundary injections.*

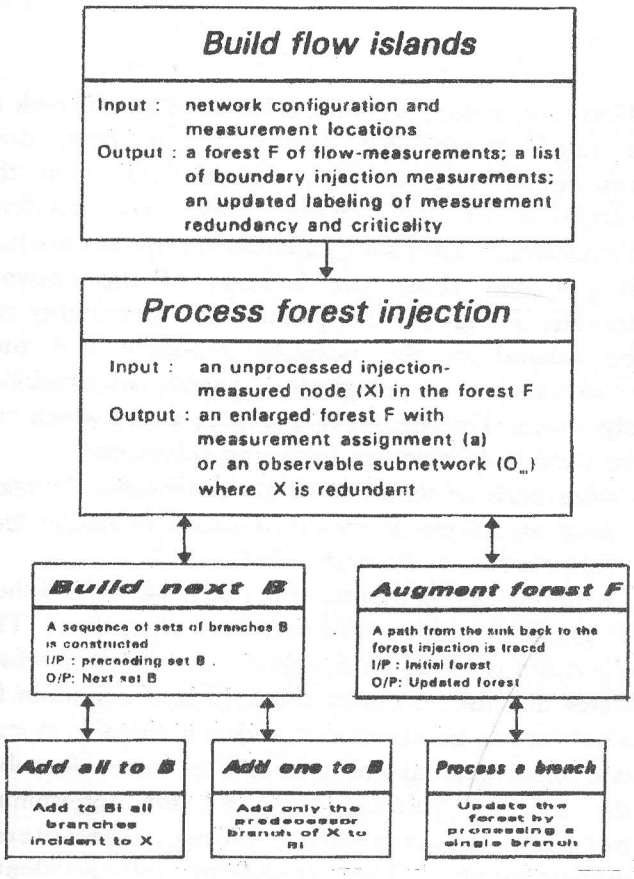


Figure 1a. Steps of the topologically based observability algorithm.

2.1 Step 1: Building the flow islands

Objective : Process must-use flow measurements.
Input : A network graph set of must-use measurements.
Output : A forest F of flow-measurements, a list of boundary injection measurements, and an updated labeling of measurement redundancy and criticality

Program structure : The program produces a forest F of flow-measured branches with the implicit measurement assignment of each flow-measured branch to itself. It also constructs a list of boundary injections for the next step of the processing. Build Flow Islands accomplishes these tasks by first performing a breadth search of the flow-measured network. An auxiliary data structure, a queue, is used for this search. (A queue is a first-in, last-out list. Nodes are added to the bottom of the queue and removed from the top). Flowchart of Figure (2) illustrates the program structure.

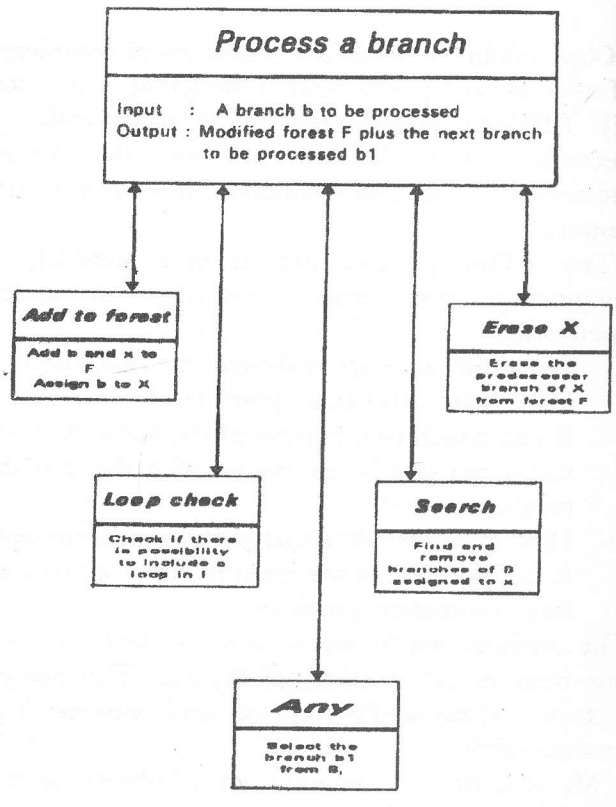


Figure 1b. Elementary subroutines of processing a branch.

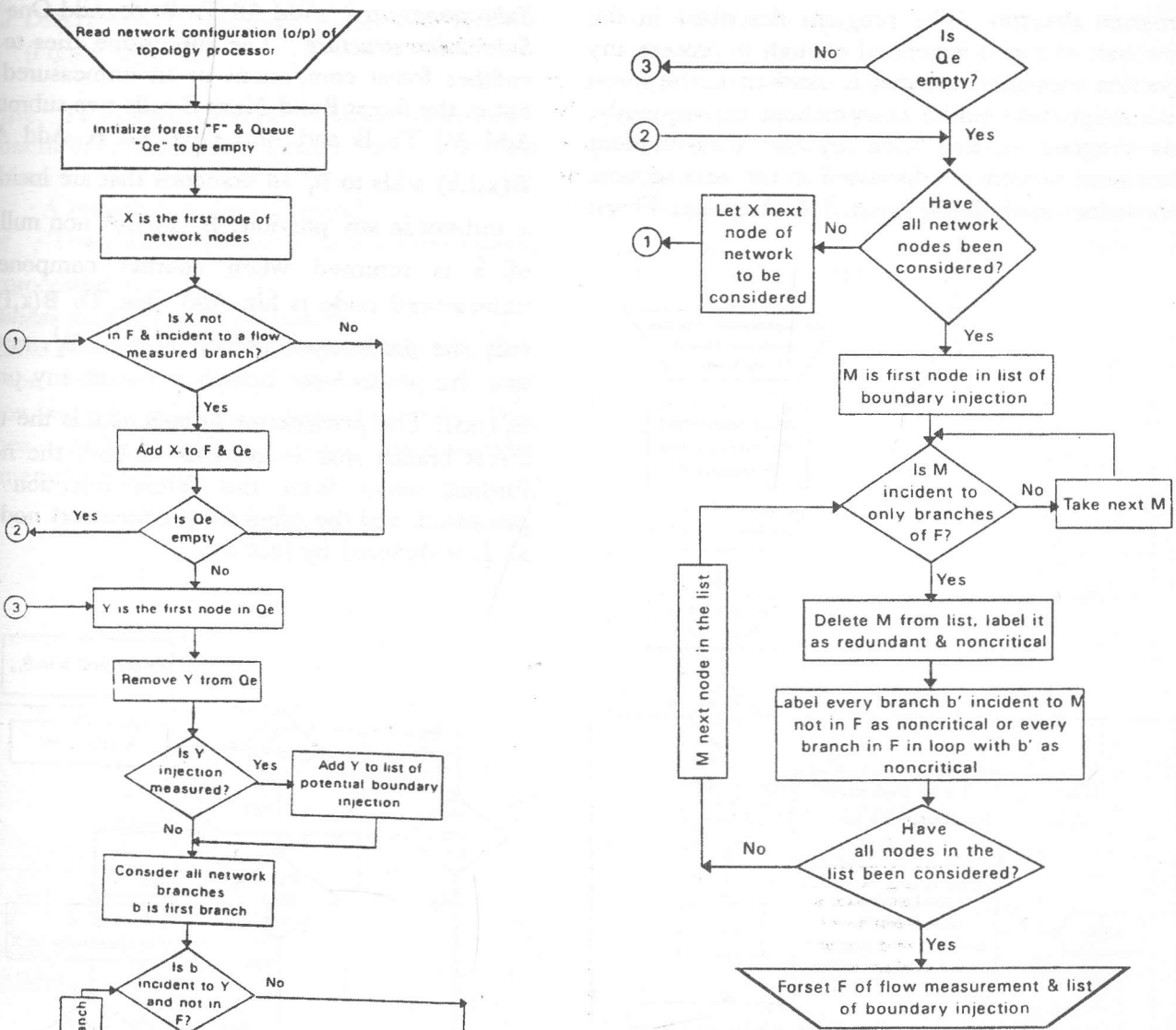


Figure 2. Procedures for building flow islands.

2.2 *Step 2: Processing the forest injections*

Objective : Process the must-use boundary injections.

Input : An injection measurement at a node x of the forest F under construction. The node x has not yet been processed, so x is not assigned to any branch of F.

Output : An enlarged forest F that contains a branch b assigned to the node x or an observable subnetwork $O_m = (G(O_m), M(O_m))$ such that $M(O_m)$ is a minimal dependent set of measurements containing x.

Program structure: The program described in the flowchart of Fig. 3 is general enough to process any injection measurement that is incident to the forest with assignment under construction; consequently, this program is also used by the measurement placement procedures discussed in the next section.
Subroutines used: Build Next B & Augment Forest F.

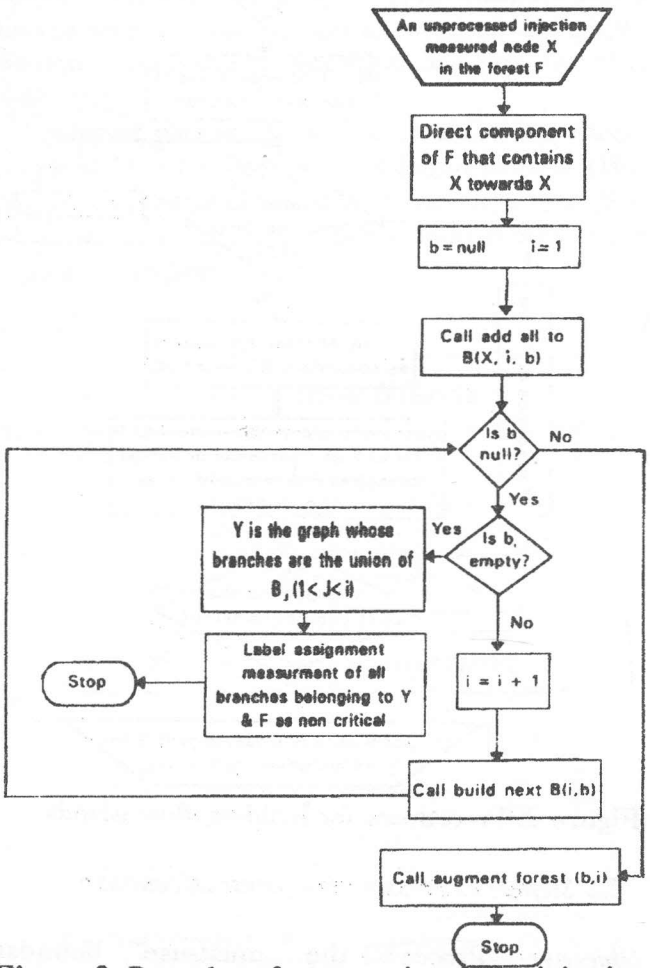


Figure 3. Procedure for processing forest injection.

2.2.1 Description of build next B subroutine (Fig.4):

Objective: Construct the sequence of sets of branches $B_i^1 (i \geq 1)$

Input : The preceding set B_{i-1}^1

Output : The next set B_i^1. (The first set B_i^1 consists of all branches incident to the forest injection being processed).

Subroutines used : Add All To B & Add One To B.
Subroutine structure : The subroutine tries to reach another forest component or an unmeasured node not in the forest. Build Next B calls two subroutines, Add All To B and Add One To B. Add All To B(x,i,b) adds to B_i^1 all branches that are incident to x and not in any previous $B_j^1 (j \leq i)$. A non null value of b is returned when another component or unmeasured node is hit. Add One To B(x,i) adds only the predecessor branch of x to B_i^1, assuming that the predecessor branch is not in any previous $B_j^1 (j \leq i)$. The *predecessor branch* of x is the unique forest branch that is incident to both the node x, furthest away from the forest injection being processed, and the other (or predecessor) node of x, x^-. It is denoted by $[x,x^-]$.

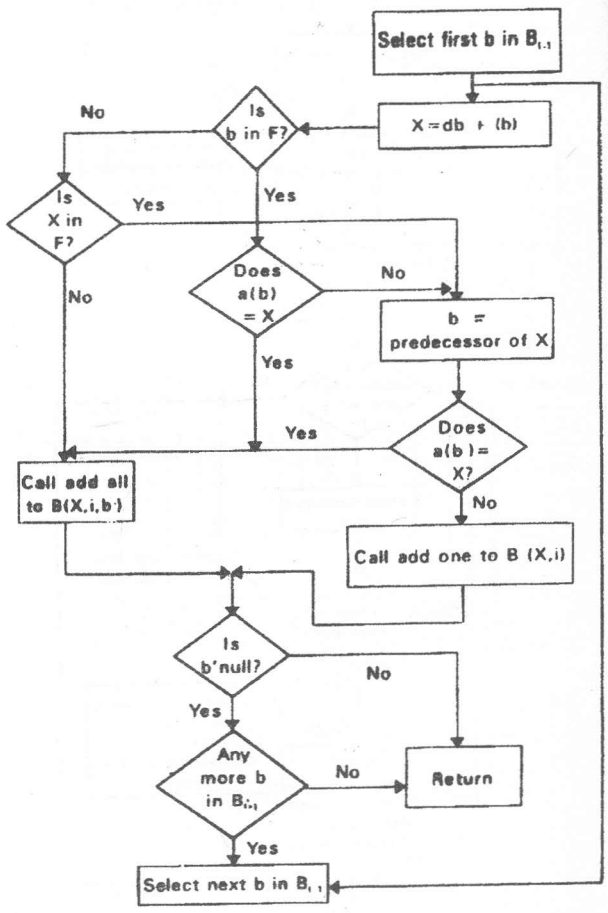


Figure 4. Procedure for building B_I from B_{I-1}.

2.2.2 *Description of augment a forest subroutine (Figure (5,6)) :*

Objective: To update the forest when another forest component or an unmeasured node not in the forest is reached.
Input : A branch b to be processed;
Output : The modified forest plus the next branch to be processed b́
Subroutines used: Process A Branch.
Subroutine structure : This subroutine traces a path from the objective reached back to the forest injection being processed, adding and deleting forest branches and modifying the measurement assignment as necessary.

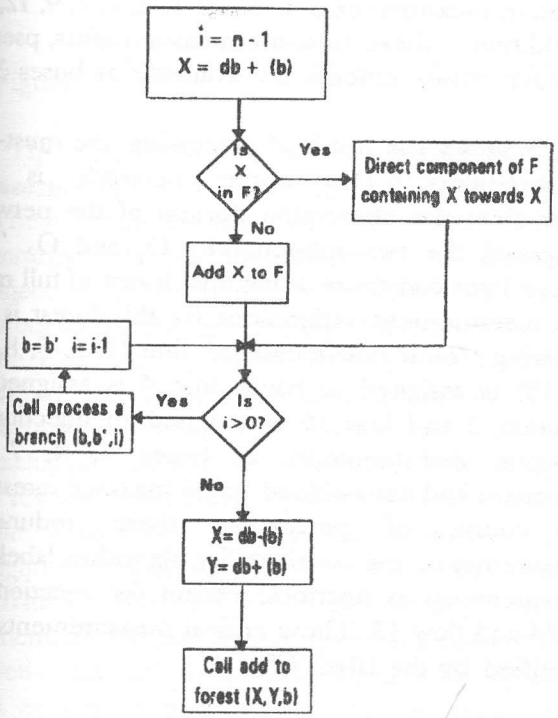


Figure 5. Procedure augment forest.

3. SAMPLE SYSTEM

The developed software computer package programs are tested on the IEEE 14-bus sample of power system. Two sets of measurement data are considered:
Case study 1 : Shows an observable system.
Case study 2 : Shows the treatment of unobservability conditions.

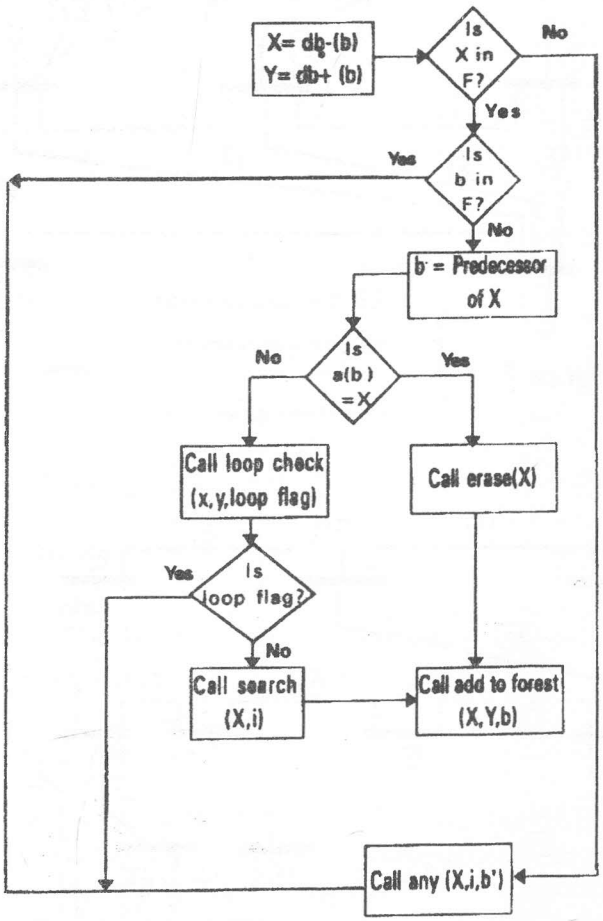


Figure 6. Procedure for updating the forest by processing a single branch.

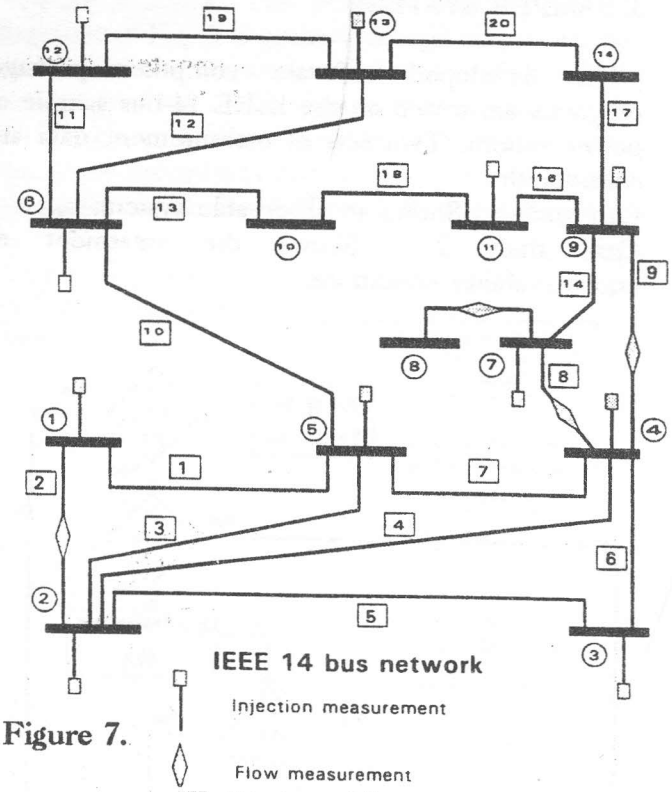


Figure 7.

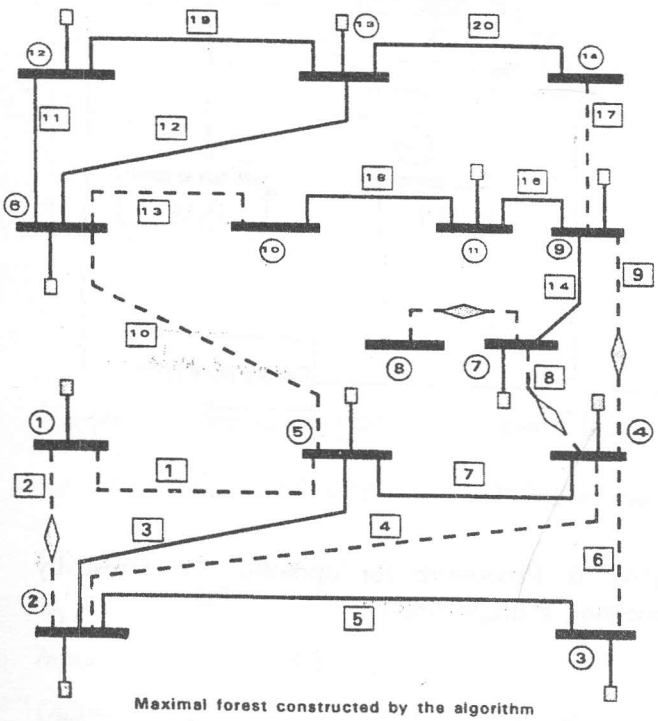


Figure 8.

3.1 Case study 1:

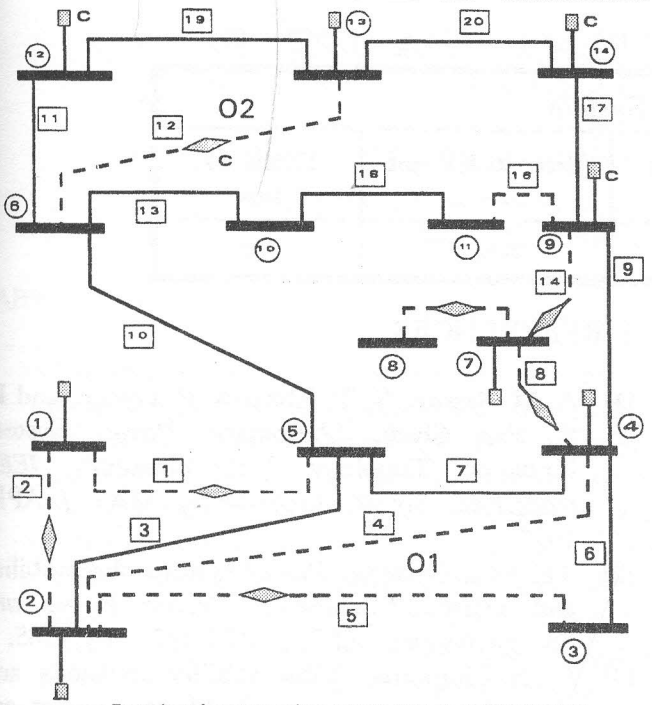
The algorithm is applied to determine the observability of the IEEE 14-bus test network with must-use measurement set indicated in Fig.7

Figure (8) gives the maximal forest constructed by the algorithm (tree T); T is connected and contains all unmeasured nodes. Since there exists a measurement assignment for T, T is a minimal spanning tree of full rank [4-6]. Therefore, the network is *observable.*

3.2 *Case study 2:*

Consider the IEEE 14-bus system with 7 flow measurements on lines 1, 2, 5, 8, 12, 14, 15 and 7 injection measurements at buses 1, 2, 4, 7, 9, 12, 14. In addition to these must-use measurements, pseudo injection measurements are available at buses 3, 5, 10.

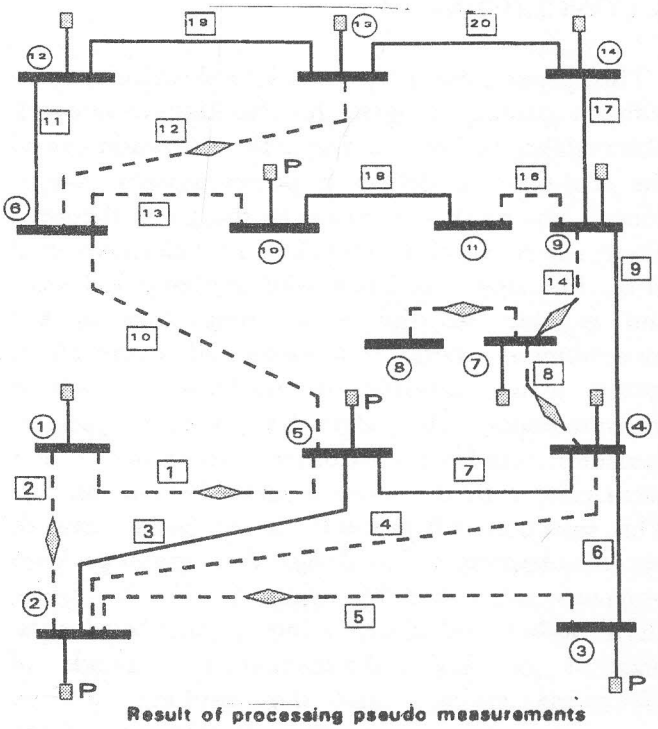
Fig.9 shows the result of processing the must-use measurements. The entire network is not observable; the observable portion of the network comprises the two subnetworks O_1 and O_2. The broken lines constitute a maximal forest of full rank. The measurement assignment for this forest is the following : each flow-measured line (1, 2, 5, 8, 12, 14, 15) is assigned to itself; line 4 is assigned to injection 2 and line 16 is assigned to injection 9. Injection measurements at buses 1, 4, 7 are redundant and not assigned to the maximal forest. In the course of processing these redundant measurements, the observability algorithm labels all measurements as uncritical except for injections 9, 12, 14 and flow 12. These critical measurements are identified by the label "C".



Result of processing must-use measurements

Figure 9.

The next step is to process the ***pseudo injection measurement*** at buses 3, 5, 10 (labeled with a "P"). These measurements are to be used only if they contribute to the observability of the network. The injection at bus 3 lies well within the observable subnetwork O_1 and, consequently, does not increase the observability of the network. (Injection 3 lies within the noncritical subnetwork of X_m; such pseudo measurements may be automatically excluded from consideration). Processing of injection 3 results in injection 3 being labeled as redundant; the observable subnetwork identified by the algorithm consists of line 1, 2, 3, 4, 5, 6. None of the existing critical must-use measurements are made uncritical by injection 3. Since injection 3 lies in a well-measured, noncritical, observable region, there is no reason to include it in the state estimate; injection 3 is therefore deleted. The pseudo injections at buses 5, 10, however, do contribute to observability. Their processing adds branches 10 (assigned to injection 5) and 13 (assigned to injection 10) to the maximal forest, making the maximal forest a minimal spanning tree and the entire network observable. (The lines of the minimal spanning tree of full rank are shown broken in Fig.10). Pseudo injections 5 and 10, therefore, are added to the measurement set and are critical.



Result of processing pseudo measurements

Figure 10.

3.3 Case study 3 :

Several tests were carried out using the IEEE 30 bus, Alexandria city 66 KV subtransmission network (45 bus and 47 lines) and IEEE 57 bus systems to show the effectiveness of the developed software computer package programs. The programs were executed on 80486 DX2, 66 MHz,IBM compatible PC-computer and the CPU times in seconds are given in Table 1.

Table 1.

	Test System			
	IEEE 14 bus	IEEE 30 bus	Alex. 66 KV net.	IEEE 57 bus
CPU sec	0.055	0.33	0.824	3.4

4. CONCLUSION

This paper presents a complete description of a software package program for checking-up network observability before entering to the state estimator of the real-time model in a power system control center. The method is based on the graph-theoretic observability algorithm developed by clements et.al [4-6]. The algorithm has a solid mathematical basis and is able to find a spanning tree in the measurement whenever it exists. When the entire system is unobservable on the basis of available measurements, the algorithm selects pseudo-measurements from a list ordered according to their accuracies, with the more accurate pseudo on top. This selection will proceed till the entire network become observable. The designed computer package programs are tested by using the IEEE 14-bus, IEEE 30-bus, and IEEE 57-bus systems besides the practical 66 KV subtransmission network of Alexandria region and the resulting reduced computation time verify the effectiveness of the developed computer package for the use in observability test in real-time modelling procedures of power systems in EM and SCADA control centers.

5. REFERENCES

[1] A. M. Sasson, S. T. Ehrman, P. Lynch, and L. S. Van Slyck, "Automatic Power System Network Topology Determination", *IEEE trans.PAS*, vol.92, pp.610-618, MAY / APR. 1973.

[2] Th. Van. Custem, "Power system observability and related functions", *Electric Power and Energy Systems*, vol 7, pp.175-182, July 1985.

[3] K. A Clements, "Observability methods and optimal meter placement", *Electric Power and Energy Systems*, vol.12, pp.88-93, April 1990.

[4] G.R. Krompholz, K.A. Clements, and P. W. Davis, "Power system observability: A practical algorithm using network topology," *IEEE Trans.PAS*, vol.99, pp. 1534-1542, July/Aug 1980.

[5] G.R. Krompholz, K.A. Clements, and P. W. Davis,"Power system state estimation with measurement deficiency: an algorithm that determines the maximal observable subnetwork," *IEEE Trans. PAS*, vol.101, pp. 3044-3052, Sept 1982.

[6] G.R. Krompholz, K.A. Clements, and P. W. Davis,"Power system state estimation with measurement deficiency: an observability / measurement placement algorithm" *IEEE Trans. PAS*, vol.102, pp.2012-2020, July 1983.