Electric Power Systems Research 106 (2014) 188–202

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

An efficient automated topology processor for state estimation of power transmission networks

Mostafa Farrokhabadi [a], Luigi Vanfretti [a,b,*]

[a] KTH Royal Institute of Technology, School of Electrical Engineering, Electric Power Systems Department, Teknikringen 33, SE 100 44 Stockholm, Sweden
[b] Statnett SF, Research and Development, Nydalen Allé 33 0484 Oslo, Norway

ARTICLE INFO

Article history:
Received 23 April 2012
Received in revised form 13 May 2013
Accepted 22 August 2013
Available online 27 September 2013

Keywords:
Topology processor
State estimation
Real-time simulation
Phasor measurement unit

ABSTRACT

A robust network topology processor that can be utilized in both traditional and PMU-based state estimators is developed. Previous works in the field of topology processing are scrutinized and their drawbacks are identified. Building on top of the state of the art, an algorithm covering the limitations of available topology processing approaches and including new features is proposed. The presented algorithm was implemented in MATLAB and tested using two different power networks with detailed substation configurations (bus/breaker models) including a modified version of the IEEE Reliability Test System 1996. As the topology processor is intended to supply network topologies to a PMU-based Sate Estimator, the IEEE Reliability Test System 1996 is simulated in real-time using the eMegaSim Opal-RT real-time simulator which is part of "SmarTS Lab" at KTH Royal Institute of Technology. Testing is carried out through several test scenarios and computation times are calculated. It is shown that the computation times are adequate for supporting a PMU-only state estimator.

1. Topology processing

1.1. Synopsis

State estimation is a key energy management system (EMS) function that provides static estimates of the system states, i.e., bus voltages and angles, and line active and reactive power flows [1]. Topology processing is a key step in the state estimation process. It refers to the determination of the system topology through available measurements and connectivity data of switch status indicators and its link to a static database; this database holds relevant information about the parameters of such a topology.

At the first glance, the task seems to be straightforward: an open switch indicates that a line is disconnected. However, the problem is more complex in practice. Considering a real power system, there are numerous substations with different configurations, and various interconnections. On top of that, there are moments when an interconnected power system is split into separate islands due to a specific switching pattern. The task of a network topology processor is to correctly deal with all of these complexities.

The objective of this paper is to develop a robust topology processor capable of supporting both traditional SEs and fast PMU-only state estimators [2,3]. The algorithm is documented in detail and relevant testing has been performed using two different test systems with detailed substation configurations, including a fictitious network and a modified version of IEEE Reliability Test System 1996 [4]. The latter is simulated in real-time using the eMegaSim Opal-RT real-time simulator [5].

From the practical point of view, several steps are involved in a typical topology processor [6]. The first step is to accurately process input data. This data consists of pre-defined constant connectivity parameters and switch statuses. Traditionally, data is telemetered through analog transmitters using TCP/IP over the ICCP protocol. However, the arrival of PMUs made it possible to send the switches status digitally with a much higher rate compared to conventional telemetry methods. These digital data is included in the IEEE C37.118-2011 protocol [7]. It is shown in this paper that the performance of the proposed algorithm is adequate for its use with high rate PMU data as input.

The second step in topology processing is to analyze different substation configurations. In fact, due to various substation configurations, an open breaker does not necessarily disconnect a line from the substation. In addition, there might be moments that one single substation is split into two (or more) nodes. Similarly, there may be cases in which these previously split nodes merge back together to reform the original station.

* Corresponding author. Tel.: +46 87906625; fax: +46 87906510.
E-mail address: luigiv@kth.se (L. Vanfretti).

0378-7796/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.epsr.2013.08.014

The next major step carried out by a topology processing engine is to detect if islanding has occurred in a power network. Reversely, it should be able to detect if some separated islands have been unified. Identification of energized and de-energized islands is also a duty of a practical topology processor. In an energized island, there is at least one operating generator. This task is also of high importance, as the de-energized islands should not participate in the state estimation process. Therefore there is no need to formulate the equations related to them.

The openly available works on the implementation of the current topology processors pose several challenges for a practical implementation; this is mostly due to a lack of rigor in the description and documentation of these methods.

To have a proper background that motivates the development of a new algorithm, the most relevant previous works are discussed below. Please note that the works considered here are related to the term "Topology Processing" and not "Topology Estimation". The former is designated to the action of determining the topology of a power system from input data (breakers' status, and bus connectivity data). Topology estimation [13] refers to the action of estimating the topology of a power system. To carry out this, the topology of the system has already been determined a priory by a topology processor. However, this topology might have errors (due to bad input data). Thus, the process of topology estimation aims to detect these errors and mitigate them. The result is a sanitized topology of the network. This article does not propose algorithms for topology estimation.

1.2. Literature review

1.2.1. Automatic power system network topology determination [8]

Introduced by Sasson *et al.*, this is the cornerstone work in the area and the common reference for the majority of proposed methods. Similarly, the method proposed in this article builds on top of the algorithm in Ref. [8]. However, we identified some of the drawbacks that limit its implementation and performance.

First, the algorithm is communicated in natural language form (written English); however, there are other dialects (such as flowcharts, influence diagrams) that are better suited for a translation into a software implementation. No flowcharts or mathematical descriptions are provided in Ref. [8]. This is challenging because for an implementation into actual computer code a more detailed functional specification and a step-by-step flowchart is needed. This flowchart must clearly describe in detail the steps needed to derive code for actual execution; it should be very similar to "data flow diagrams" which are used in computer programming (this is what we have attempted in our manuscript, as shown in the forthcoming sections).

There are other limitations in this algorithm that affect its performance. Whenever a breaker status change occurs, the algorithm needs to perform topology processing for all the substations with breaker statuses different from their initial values. A more sensible approach is to compare the breakers statuses with the ones of the most recent snapshot. This inability is due to the fact that the TP does not track the changes that have occurred in the previous cycle; this is a consequence from the lack of a number assignation procedure to mark splitting and merging nodes. As a result the TP has to redefine the whole topology of the system in every snapshot so to avoid numbering uncertainty. Even if the algorithm processes the topology and compares it to a pre-defined initial topology, it may significantly change the numbering of the nodes which were previously involved in splitting/merging incidents.

Another difficulty takes place during node merging/splitting. As the information regarding the original station numbers are not saved in the "configuration matrix" the algorithm may malfunction because this matrix is used in islanding analysis.

Additionally, the algorithm only considers changes within a substation. Switching changes that are not located within the substation configuration such as those placed on the lines are not accounted for. Furthermore, the rules of this TP are not able to account for ring-bus with series breakers configurations.

Finally, the TP is not able to detect if the islands are energized or de-energized. This is very important as the parts of the network which are not energized should not participate in state estimation or static analysis (power flow solution).

1.2.2. Real-time modeling of power networks [9]

Real-time modeling of power networks in [9] refers to a set of computer-based functions that aims to represent the current operation states of the power network. It comprises different functions such as topology processing, observability analysis, state estimation, bad data detection and external network modeling. In Ref. [9] all the required steps for real-time modeling of a power network are explained and general procedures for each one is described. The discussion on topology processing in this article is a short general definition of the topology-processing concept. The necessary steps for TP are only communicated in natural language form (written English) and no detailed dialects (such as flowcharts) are provided for computer software implementation.

1.2.3. A topology processor that tracks network modifications over time [10]

Ref. [10] provides an approach to save the outputs of the topology engine for each execution cycle; it uses this saved data to reduce the calculation burden of the state estimation process. Although the paper has a top-level descriptive flowchart, it does not provide a particular algorithm for the topology engine itself. The methodology used in the topology engine in Ref. [10] is based on the one proposed in Ref. [8]. Therefore, it is exposed to the same drawbacks as discussed in Section 1.2.1.

1.2.4. A new algorithm of topology analysis based on PMU information [11]

To the knowledge of the authors, this work is the only one that considers the usage of PMUs for topology processing. The basic idea of this work is to measure the current flows and compare them with a pre-defined threshold. However, necessary elements for topology processing, such as how to deal with different configurations, automatic numbering, and islanding checks, are not addressed by the algorithm proposed. Furthermore, the algorithm neglects the fact that when a change occurs in the network, this change may affect large portion of the network (i.e. this effect is not strictly localized). For example, if two lines are connected in series, the disconnection of one may result in the current flow through the other line to be low. This issue is investigated in detail in Ref. [14] and possible remedies are provided.

1.2.5. A new approach to initializing and updating the topology of an electrical network [12]

The method introduced in Ref. [12] uses graph theory. The power system network is transformed to a graph, and using a specific algorithm, this graph is transformed into a sparse matrix. A mathematical procedure makes use of this sparse graph to solve for the network topology.

The TP documented in Ref. [12] is more of an industrial report, with limited information on the dialectics for its implementation. Specifically, the methodology to solve the sparse matrix and determine the network topology is not elaborated. Furthermore, the explanations on how to transforms the power network to an equivalent graph is explained using an example, and no general

procedure is provided. Finally, the reported results are limited to the computation time required to determine the network topology.

2. Automated topology processing: proposed algorithm, basic rules, inputs, and outputs

For an actual implementation of the PMU-only state estimator approaches like the ones in Ref. [2] and [3], there is a need to develop a robust network topology processor. The discussion provided in the previous section reveals the need for a practical topology processor that is exhaustively described and documented so that it is possible to implement it in software code without ambiguity.

There are two major requirements that the proposed topology processor must meet:

- It should be applicable to state estimators focusing on the HV backbone of the network, i.e. transmission networks [1–3].
- It should be applicable to (balanced) 3 phase switch status changes.

Several new functionalities are proposed to increase the algorithm robustness and efficiency:

- It performs topology processing just for those parts of the system that have suffered changes compared to the previous execution cycle, and not the initial pre-defined statuses.
- It deals both with splitting and merging of nodes efficiently.
- The algorithm is described using dialectics to facilitate its implementation in computer software.
- All the necessary parts of a practical topology processor [1] are included.
- The algorithm automatically assigns numbers to any recently formed island or splitting/merging nodes in an efficient way so that the changes in each part of the network can be quickly tracked.

Rigorous tests are provided in Section 4 as evidence to support these statements.

2.1. Basic rules

The algorithm is built on top of the TP proposed in Ref. [8]; hence it shares some common rules. In addition, new rules are introduced to support new functionalities and address identified limitations, as listed below:

(1) Every substation should be assigned a number from 1 onwards.
(2) Every switch should be assigned a number from 1 onwards.
(3) Every line should be assigned a number from 1 to n. Then, the bus bars in the system are assigned numbers from $n+1$ onwards.
(4) A switch status is either 1 or 0.1 represents the closed position, while 0 means that the switch is open.
(5) A switch can not be connected to more than two circuits, i.e. lines or buses. In the case that there is a switch connected to more than two circuits, phantom switches should be introduced (which are always closed).
(6) Each switch should be listed with one of the following four types:
   - Type 1: Switches which are located on a line directly connected to a generator.
   - Type 2: Switches which are located on a line connecting two different substations.
   - Type 3: Switches which are located within a substation.
   - Type 4: Switches which are located on a shunt element, i.e. loads or capacitor banks.

2.2. Inputs

The proposed algorithm in this method reads the input data and forms two matrices described below.

2.2.1. Switch table matrix

This matrix contains information regarding switches and their corresponding circuits. If there are N switches in the power network, the "switch table" matrix would be an $N \times 8$ matrix. Each column contains specific information, as follows:

- Column 1: switch numbers – the first column contains the numbers assigned to the switches from 1 to N.
- Column 2: near substation number – by near substation number we refer to the closest substation to the switch. For Type 1, 3 and 4 switches the number corresponds to that of the substation to which they belong to. For Type 2 switches, which are located on lines connecting two different substations, the nearest station's number is entered.
- Column 3: far substation number – for Type 1, 3 and 4 switches the number corresponds to that of the substation to which they belong to. For Type 2 switches the furthest substation's number is entered.
- Column 4: switch type.
- Column 5: switch status – either 1 or 0.1 represents that the switch is closed, while 0 indicates that the switch is open.
- Column 6: circuit analysis – based on the type of the switch, a switch is either placed on a single circuit (1, 2, 4) or is connected to two different circuits (3). For Type 2 switches, the entry of Columns 6 is the corresponding line number and the entry of Column 7 is 0. For Type 3 switches, the corresponding circuits' numbers are entered in Columns 6 and 7. Please note that if any circuit is connected to more than one switch, then its number should be entered as negative. Finally, for Type 1 and 4 switches the entries for both columns are 0 (Note that the order of entries in rows 6 and 7 is not important for Type 3 switches).
- Column 7: refer to Column 6.
- Column 8: original substation number – assume that all the switches are closed so there is no split nodes in the network. The original substation number is the number assigned to substation to which the switch belongs.

2.2.2. Configuration matrix

The configuration matrix conveys information regarding different circuits in the power network. To maintain consistency with Ref. [8], a circuit here is defined as either a single bus or a single line. The TP will analyze this matrix to find the connectivity configuration of different circuits in the network. If there are M circuits in the network, i.e. M different lines and buses, the configuration matrix would be an $M \times 11$ matrix. Each column contains specific information regarding the circuit corresponding to that row. The type of the information and how the columns are filled is as follows:

- Column 1: The first column contains circuit numbers from 1 to M.
- Column 2, 3, 4, 5, 6, 7, 8: The content of these columns depends on the type of the circuit, and its measurement configuration. In fact, a circuit can be either a line or bus bar, and may have different measurement configurations. Additionally, if a circuit is a line, it connects two different substations, while if it is a bus bar it only belongs to one single substation. Hence, a circuit can be classified in different "cases" as shown in Fig. 1.

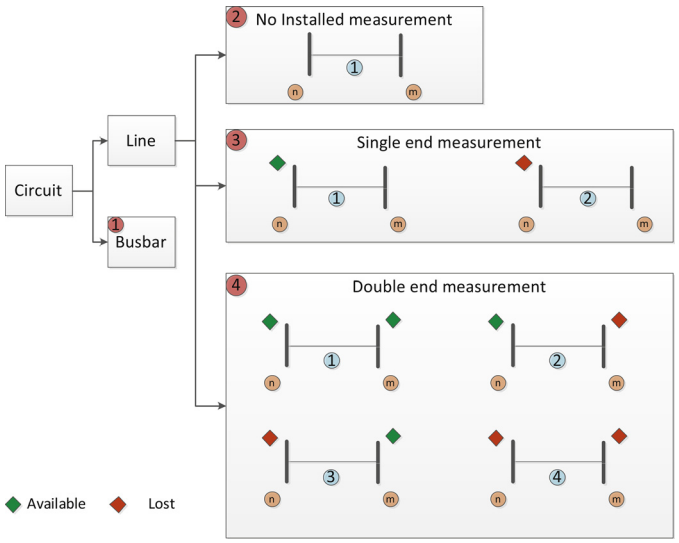

Fig. 1. Circuit measurement configurations.

Columns 2–8 carry information regarding the substations to which the circuits belong to, and the availability of their measurements. The content of columns 2–4 is defined by the stations numbers to which the circuits belong to. Then columns 5–8 carry information regarding the measurement availability.

There are four major cases as shown in Fig. 1. Case 1 is when a circuit is a bus bar. Case 2 is when a circuit is a line but with no measurement installed at neither ends of the line. Case 3 is a line with a single measurement installed at one end. Depending on measurement availability, case 3 is divided into two minor cases as shown in Fig. 1. Finally, case 4 is a line with measurements installed at both ends. Again, depending on measurement availability case 4 is divided in 4 sub-cases as shown in Fig. 1. Columns 2, 3, 4, 5, 6, 7 and 8 of the configuration matrix are filled according to these cases. The procedure is as follows:

- For case number 1, i.e. when the circuit is a bus, the number of the bus' station is entered in the second column, and in columns 3, 4, 5, 6, and 8 the entries are 0.
- For case number 2, i.e. when the circuit is a line between stations n and m, regardless of the order, n and m are the entries of columns 2 and 3. All the columns 4, 5, 6, 7 and 8 entries are 0.
- For case 3.1, the entry of Columns 2 is n, the entry of Columns 3 is m, the entry of Columns 4 is 0, the entry of Column 5 is 1, the entry of Column 6 is 0, the entry of Column 7 is 1, and the entry of Column 8 is 0.
- For case 3.2, the entry of Column 2 is n, the entry of Column 3 is m, the entries of all the columns 4–8 are zero.
- For case 4.1, the entry of Column 2 is n, the entry of Column 3 is m, the entry of Column 4 is n, and the entries of all the columns 5–8 are 1.
- For case 4.2, the entry of Column 2 is n, the entry of Column 3 is m, the entry of Column 4 is n, the entry of Column 5 is 1, the entry of Column 6 is 0, the entry of Column 7 is 1, and the entry of Column 8 is 0.
- For case 4.3, the entry of Column 2 is n, the entry of Column 3 is m, the entry of Column 4 is n, the entry of Column 5 is 0, the entry of Column 6 is 1, the entry of Column 7 is 1 and the entry of Column 8 is 0.
- For case 4.4, the entry of Column 2 is n, the entry of Column 3 is m, the entry of Column 4 is n, and the entries of all the columns 5–8 are zero.

Table 1
Substation number matrix.

Original substation numbers	Derived station no.	Derived station no.
1	5	6
2	0	0
3	7	8
4	0	0

- Column 9: This column contains the number assigned to the island to which the substation of the second column belongs to. In the case that there is only one island in the system, Column 9 entry's is 1.
- Column 10: This column contains the number assigned to the island to which the substation of the third column belongs. In the case that there is only one island in the system, Column 9 entry's is 1.
- Column 11: There are occasions when an original station is split into several substations (from the bus/breaker point of view). If the substation number entered in Column 2 is a split station, the number of the original station to which this split station belongs is entered in Column 11.
- Column 12: Similar to the previous column, if the substation number entered in Column 3 is a split station, the number of the original station to which this split station belongs to is entered in Column 11.

An example of this table is included for the test scenario in Section 4.1.

2.3. Outputs

The outputs of this algorithm are also in matrix-form as described below.

2.3.1. Updated switch table and configuration matrices

Both the switch table and configuration matrices are updated and ready to be used as an input for the next TP execution cycle. If any new node has been split or any new island has been formed, their corresponding numbers are replaced with the previous number. If any circuit is open or disconnected a minus sign is introduced before its number in the configuration table.

2.3.2. Sub matrix

This matrix reports the substation(s) in which any switching has occurred.

2.3.3. Substation number matrix

The substation number matrix holds information regarding splitting/merging nodes. In fact, if there are L original substations in the system, i.e. when all the switches are closed there are L substations, this matrix is a $L \times K$ matrix. The first column is the number assigned to the original substations, and the other columns contain numbers assigned to nodes that are derived from this original substation. For example, if there are 4 stations in a system, and stations 1 and 3 each are split to 3 separate nodes, the corresponding substation number matrix is as shown in Table 1:

2.3.4. List matrix

For each substation in which switching has occurred, a list matrix is created. The number of rows is equal to the number of "separate closed paths" within a substation. In the columns the circuits' numbers belonging to each closed path are entered. A closed path can contain one single component (which means that the

Table 2
Heart matrix.

3	3	2	4	1	4	5	8	10	14	16	13	0	0
7	2	4	6	20	23	27	28	30	31	34	36	40	41

circuit is disconnected) or consists of several circuits. In the case that a substation has two or more closed paths, each consisting of two or more circuits, the substation has been split into different nodes.

2.3.5. Heart matrix

As the name suggest, this is a key output matrix. It contains most of the information necessary to interpret the processed network topology in a compact form. To better understand the matrix schema an example is provided. Please note that the dimension of the heart matrix is dependent of the nature of the changes that have occurred in the system. In a system with 10 substations, suppose that two substations have had changes in their switch statuses, let's say substation 3 and substation 7. Now assume that there are three closed paths within substation number 3, while there are 2 separate closed paths in number 7. The circuit numbers that belong to each path in substation 3 are (4, 5), (8, 10, 14, 16), and (13); also the circuit numbers that belong to each path in substation 7 are (20, 23, 27, 28), and (30, 31, 34, 36, 40, 41). The corresponding Heart matrix for this cycle of changes is as shown in Table 2:

As it can be observed from the simple example above, by analyzing this matrix, the number of newly formed nodes, their corresponding circuits and the circuits that are disconnected due to changes within a substation can be identified and reported.

2.3.6. Island matrix

This matrix includes three types of information. First, the row numbers show the number of separate islands in the system. The numbers assigned to each island are entered in the first column. The next column presents the number assigned to substations that belong to the corresponding island of the same row. Finally, the last column shows whether the islands are energized or not.

Fig. 2 shows the input and output matrices and their relation in general.

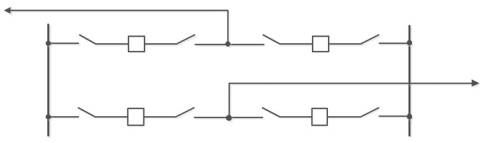

Fig. 4. Switch-disconnector model.

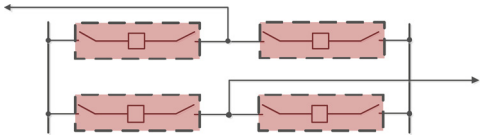

Fig. 5. Switch-disconnector series.

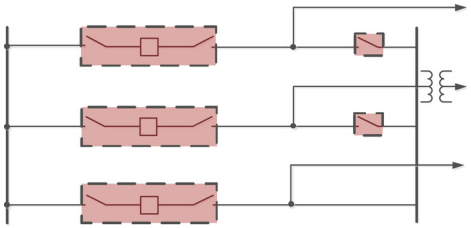

Fig. 6. Switch-disconnector series.

3. Algorithm execution procedure

In this section, the presented algorithm is explained in detail. The algorithm can be divided in six different parts as shown in Fig. 3:

3.1. Read the input matrices

This section is in charge of reading inputs, i.e. switches' statuses and network connectivity data. In previous works disconnectors were neglected in the tested models. However, the presence of disconnectors have a considerable effect on the outcome of the topology processor.

Usually disconnectors are located at both sides of a breaker, and are used for its maintenance. In that case there will be a series of switch-disconnectors, as shown in Fig. 4:

Each series of switch-disconnectors can be considered as one single unified switch as shown in Fig. 5:

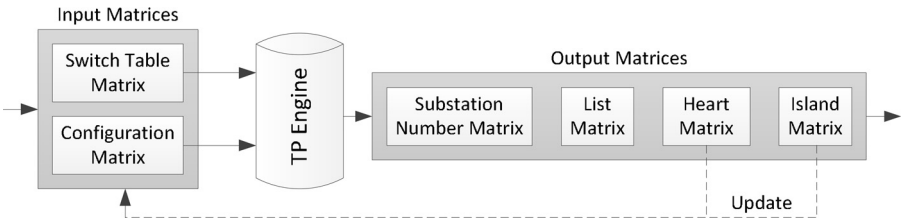

Fig. 2. Input and output matrices.

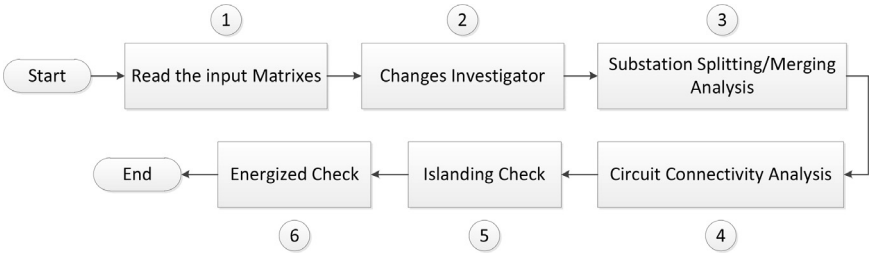

Fig. 3. Algorithm flowchart.

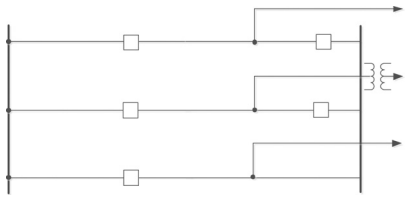

Fig. 7. Unified switch model.

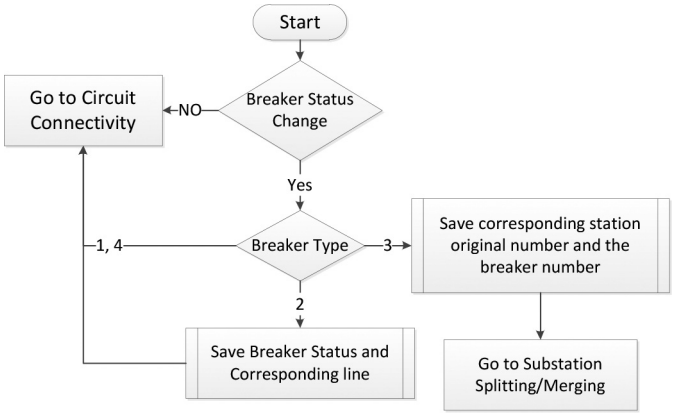

Fig. 8. Changes investigator flowchart.

The status of a unified switch is then determined using the status of all the switches and disconnectors that belong to it. All the series switches, either breakers or disconnectors, are considered as one single unified switch as shown in Fig. 6. There may be single disconnectors in the system; they are treated as a unified switch.

After the unified switches are formed, each one can be treated as a single switch when determining the system's topology (Fig. 7).

Reading the input data, the topology engine first searches for those switches and disconnectors that have experienced a change in their status. Afterwards, their respective unified switch status is re-checked. Please note that the numbers of the switch table

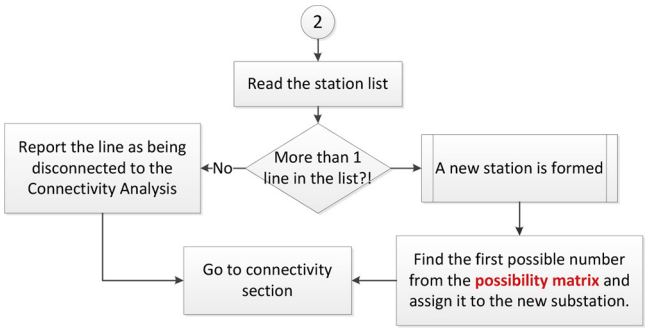

Fig. 10. Substation splitting/merging.

matrix are assigned to unified switches, and not to single breakers or disconnectors.

3.2. Changes investigator

As shown in Fig. 8, the topology engine compares the unified switch statuses of the current snapshot with its previous one and identifies the switches with changes in their status. In the case that the switch type is 3, it saves both the switches' number and their corresponding original substation number. Switches Type 1 or 4 plays no role in circuit connections, so there is no need to save their changes. For switches Type 2, the switch number and its corresponding line number should be also saved.

3.3. Substation splitting/merging analysis

If there is any Type 3 switch that has had a change in its status, the program then starts scrutinizing the changes within the substation. Otherwise it simply skips this step. This step investigates the substations that have experienced switching, one by one. For example, assume that substation 1 has previously split into three nodes 1, 8, and 10. Now in the current execution cycle, the topology engine finds that there are some changes within node 8. The TP determines the topology for the *original* substation 1 again.

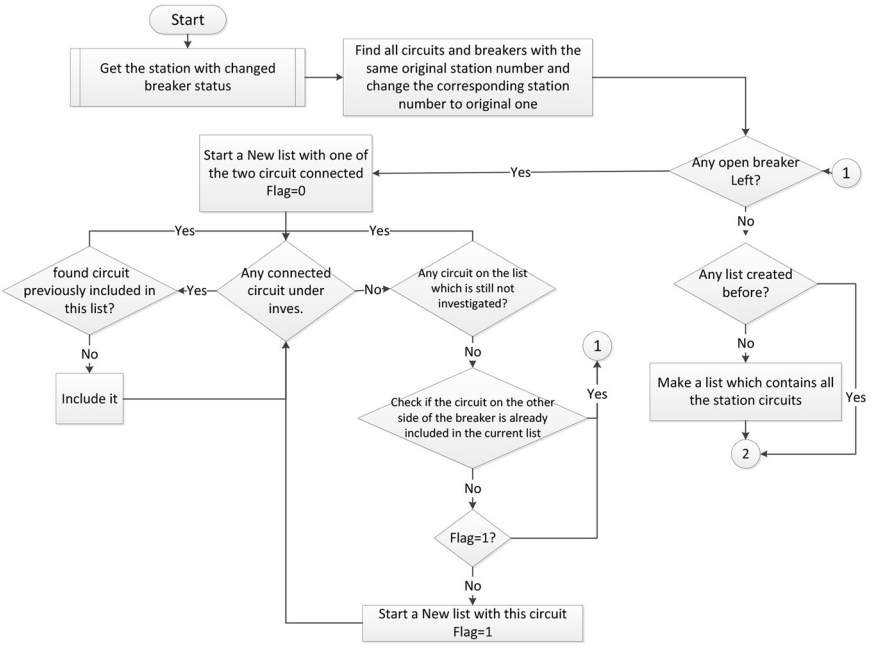

Fig. 9. Substation splitting/merging.

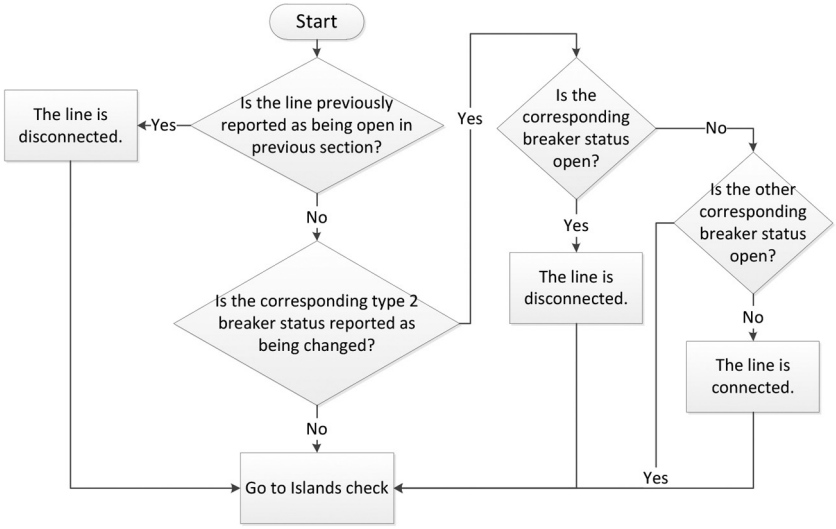

Fig. 11. Circuit connectivity analysis.

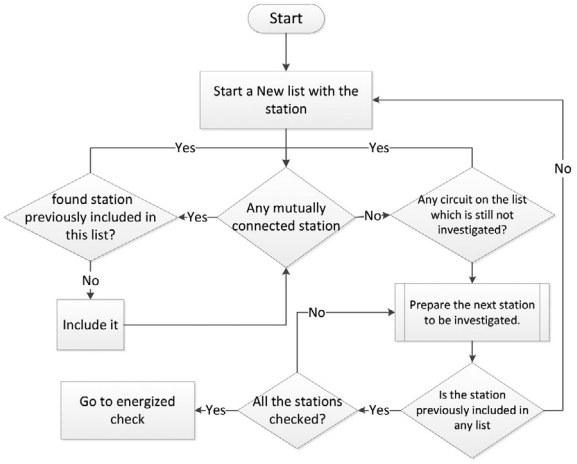

Fig. 12. Islanding check.

This makes the topology processor able to efficiently deal with any splitting/merging nodes.

To re-determine the topology, the topology engine searches for any open switch within the station. If there are no open switches, it makes one list for the station that consists of its corresponding circuits. However, if it finds any open switches, it saves the switch number. Then it starts a list with one of the two circuits connected to this open switch. By looking through the switch table matrix, it finds all the other circuits that are connected through the closed switches to the circuit previously added in the list. It saves all the switch numbers that are located through the closed path in a separate matrix. It performs the same procedure for all the added circuits to the list one by one, until no any further circuits can be added. This procedure is carried out for all the open switches, until all the closed paths are found (Fig. 9).

The output of this section is the List Matrix. The List Matrix is analyzed to see how many components each closed path contains, and what those components' numbers are. The output of this analysis is the Heart Matrix.

The next step is to assign numbers to newly formed nodes as shown in Fig. 10. For each substation, if there are two or more lists which contain more than one circuit it means that the original substation has been split. To assign numbers to these newly formed nodes, the first step is to build a vector of unoccupied numbers. This vector is named *possibility vector*. The possibility vector is a vector from 1 to *m*, i.e. a number which is greater than the total number of original stations by a safe margin (usually 5 times greater). After the substation with changes has been recognized, from the *substation number matrix* of the previous cycle, the numbers assigned to previously split nodes are replaced by 0 in the possibility vector (which means this number is available to be assigned in further steps).

After the possibility vector is formed, the newly formed nodes are assigned a number from the first possible number onwards. By number assignation, we refer to two different tasks. The first task is to find all the circuits and switches belonging to the new node, and replace their previous node with the new one in both the

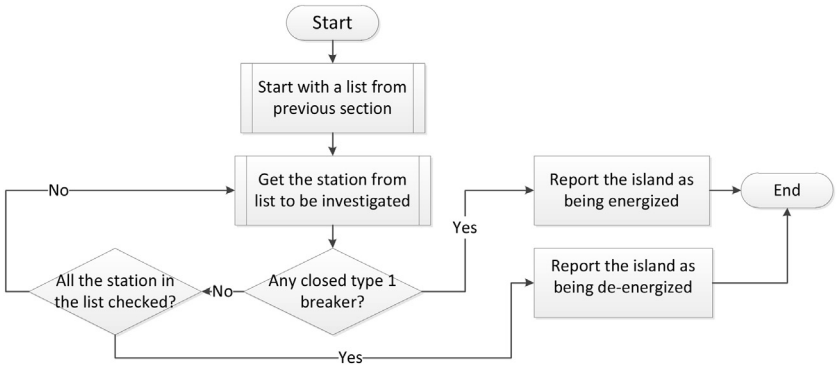

Fig. 13. Energized check.

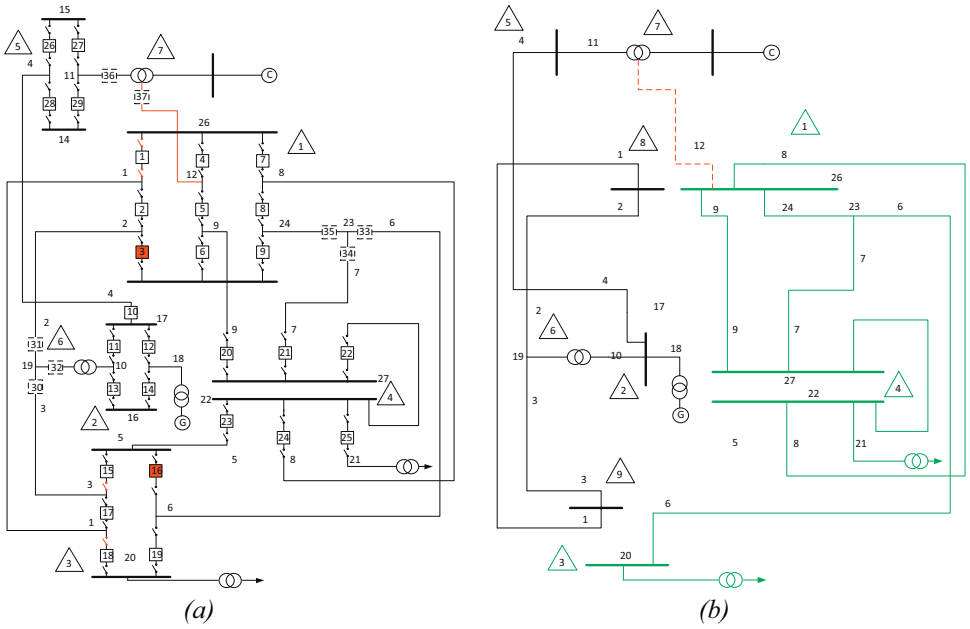

Fig. 14. First scenario, (a) bus-section/switching-device model; (b) bus/branch model. Elements in red indicate open breakers/disconnectors.

configuration matrix and *switch table matrix*. This is done by searching the original station column and finding the corresponding element. The second task is to build the *substation number matrix* to be used for the next cycle.

Please note that before any new entry is added to the list, a redundancy check should be performed to block redundant circuit numbers to enter in the same list. The procedure checks if the switch under investigation has been previously entered in the used switches list, or if the circuits that are going to be added previously exist in the list.

The next step is to build the substation number matrix. This is done by a simple search in the *switch table matrix*; it finds out the node number entered for Type 3 switches of the original substations which have suffered changes.

3.4. Circuit connectivity analysis

This analysis is performed in two different steps. One step is performed simultaneously during the splitting merging analysis as shown in Fig. 9. The splitting/merging analysis scrutinizes the *Heart Matrix* to check the lists; in the case that there is just one circuit in the list under investigation, it simply reports that circuit as being disconnected.

The other part is dedicated to Type 2 switches as shown in Fig. 11. If the switch status is 0, the topology engine finds the corresponding line through a search in the *switch table matrix*, and enters its number to the list of disconnected lines. If the Type 2 switch status has changed from 0 to 1, the sections search for all the other Type 2 switches corresponding to the same line. If all of

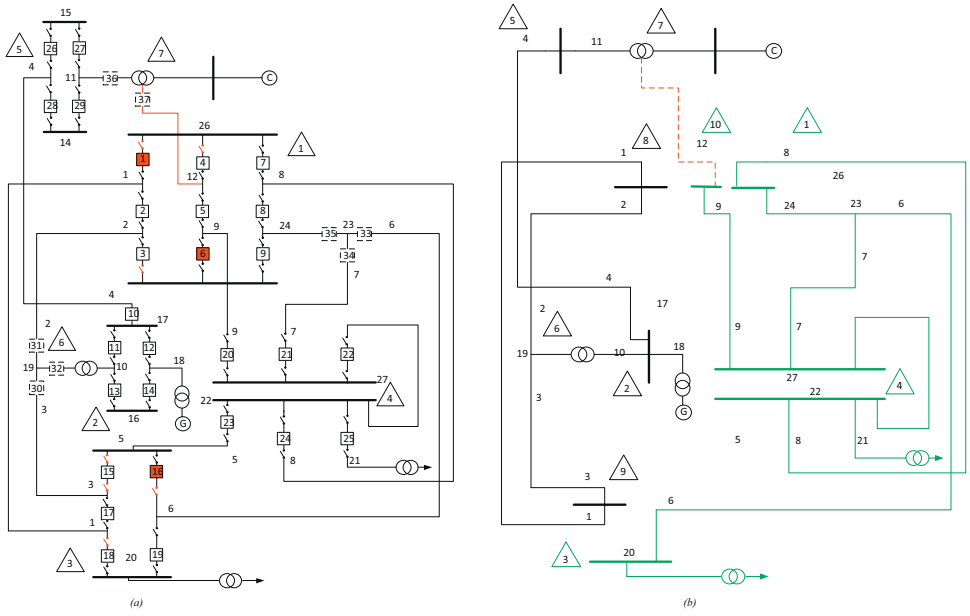

Fig. 15. Second scenario, (a) bus-section/switching-device model; (b) bus/branch model. Elements in red indicate open breakers/disconnectors.

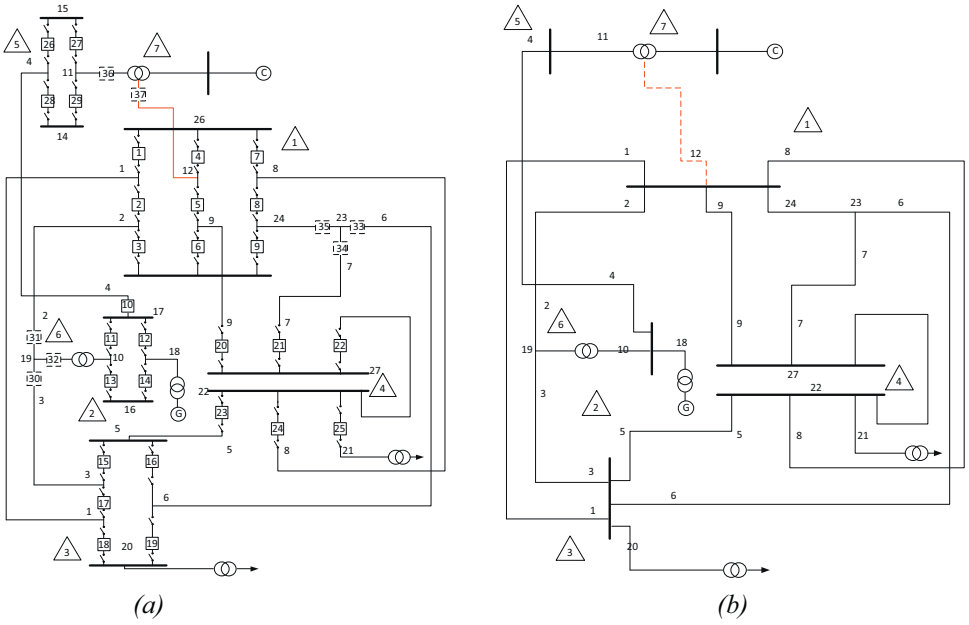

Fig. 16. Third scenario, (a) bus-section/switching-device model; (b) bus/branch model.

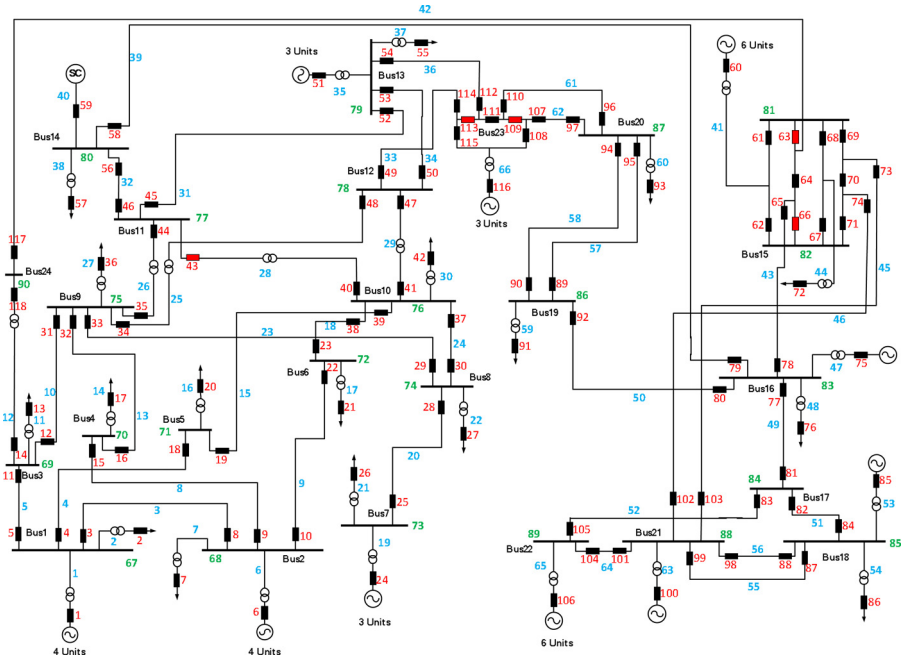

Fig. 17. First scenario. Unified switches in red indicate an open status.

them are closed, the section reports that the line has transitioned from disconnected to connect.

3.5. Islanding check

The approach of this step is very similar to the one used when checking for changes within substations. As shown in Fig. 12, a list starts with the first substation and then all mutually connected substations are added. The same analysis is performed on all added stations and at the same time a redundancy check is performed so that there is no additional entry of the same station number in the same list twice. If more than one list is created that means there is more than one island in the network.

3.6. Energization check

As shown in Fig. 13, an energization check is performed by searching for any closed Type 4 switch connected to one of the stations belonging to the island. This is done by scrutinizing the *switch table matrix*.

4. Testing results

This section provides results from a series of rigorous tests on the new topology-processing algorithm. The topology engine is tested using two different test systems. The test systems are introduced and information regarding their respective topological

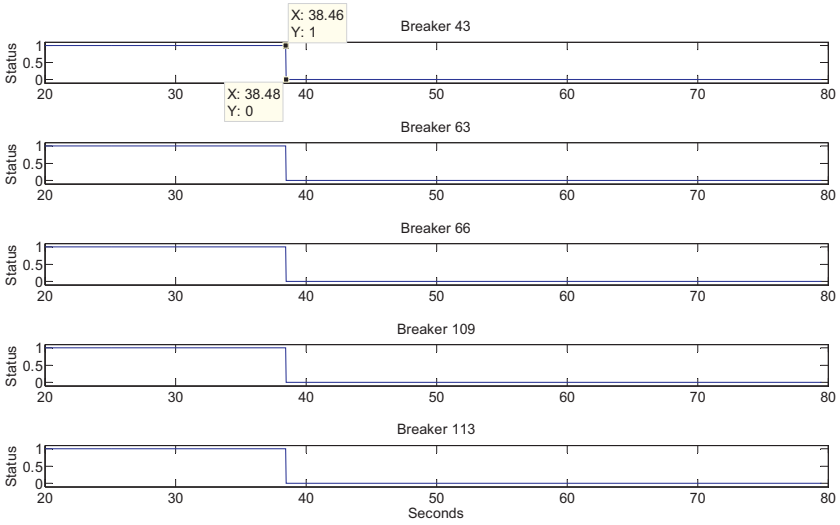

Fig. 18. Breakers status.

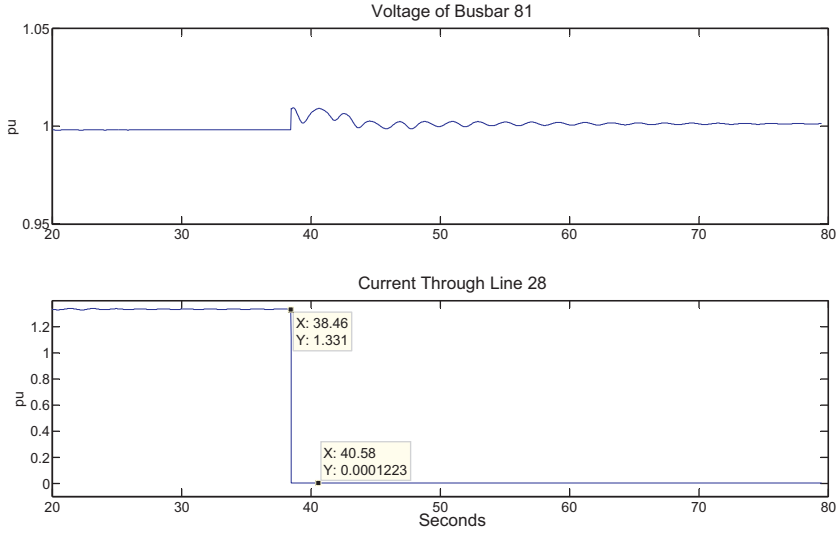

Fig. 19. Sample measurements.

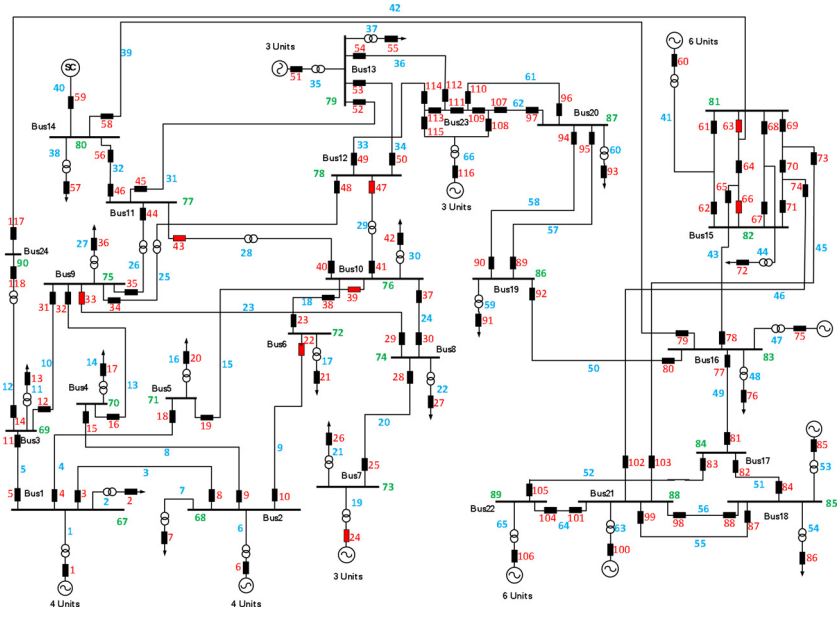

Fig. 20. Second scenario. Unified switches in red indicate an open status.

Table 3
Scenarios' description and results.

Scenario	Description	Results
1	5 unified switches change status simultaneously. 2 stations suffer changes inside.	Substations 1 and 3 are reported as stations with changes inside. For substation 1 two closed paths are found; one contains two circuits: 1 and 2 while the other contains the remaining 6 circuits. For substation 3, three closed paths are found; one of the paths just contains a single circuit, number 5, which means it is disconnected. The other two contain two circuits each; 1 and 3, 6 and 20. Line 5 is reported as "disconnected". Line 12 is reported as "not available" for power flow. Node 8 (lines 1, 2) and 9 (lines 1, 3) are automatically assigned numbers. Two separate islands are reported (black, and green) and assigned numbers automatically; Island 1 consists of stations 1, 3 and 4. Island 2 consists of stations 2, 5, 6, 7, 8 and 9. Computation time: 4.6 ms
2	2 unified switches change status simultaneously. 1 Stations suffer changes inside.	Substation 1 is reported as the only station with changes inside. Three closed paths are found. One has 4 different circuits: 8, 13, 24 and 26, while the other 2 have 2 circuits each: 1 and 2, 9 and 12. Line 12 is reported as "not available" for power flow. Node 10 (circuits 9 and 12) is automatically assigned a number. Two separate islands are reported and assigned numbers automatically. Island 1 consists of stations 1, 10, 3 and 4. Island 2 consists of stations 2, 5, 6, 7, 8 and 9. Computation time: 1.5 ms
3	7 unified switches change status simultaneously. 2 Stations suffer changes inside.	Stations 1 and 3 are reported with changes inside. In both stations just one single closed path can be found containing all the stations circuits. Nodes 1, 8 and 10 are merged and automatically assigned number 1. Also nodes 3 and 9 are also merged and assigned number 3. Line 5 is reported as connected again. There is just one island in the system containing all the system stations. Computation time: 974 us

Table 4
Scenarios' description and results.

Scenario	Description	Results
1	5 switches are opened simultaneously. 2 Stations suffer changes inside.	Stations 15 and 23 are reported with changes inside. For station 15, 2 closed paths are found. One of them has two circuits 42 and 43 (node 25) while the other one has the remaining 6 circuits of the station (node 15). For station 23, 2 closed paths are found. One of them has two circuits 36 and 61 (station 26), while the other one contain 3 circuits 62, 33, and 64 (node 23). Line 28 is reported as disconnected. There is just one island in the system containing the whole system stations. Computation time: 5.4 ms
2	7 switches change status simultaneously. 1 Stations suffer changes inside.	Station 23 is reported as the only one with changes inside. There is one single closed path containing all of the station circuits (nodes 23 and 26 are merged). Lines 23, 9, 29, 15, and 19 are reported as disconnected. Lines 28 was disconnected previously. There are two islands in the system; one contains substation 6, 7, 8 and 10 and is de-energized. The other contains the remaining stations and is energized. Computation time: 6.6 ms
3	8 switches are closed simultaneously. 1 Station suffers changes inside	Station 15 is reported as having changes inside. For this station, just one single closed path is found which includes all its circuits. Nodes 25 and 15 are merged and formed the original station 15. Lines 9, 19, 23, 15, 28, 29, and 41 are reported as being connected again. There is just one island in the system containing the whole system stations which is energized. Computation time: 6.4 ms

characteristics is provided. In addition, the designed testing scenarios are introduced. Finally, the outputs of the topology engine for each scenario are provided in tabular form.

The test systems are not of the size of realistic large-scale power networks. To address this limitation in testing of the topology processor, we instead developed complex test scenarios that involve sequential and multiple-simultaneous switching. The degree of complexity of these testing scenarios serves to compensate for the fact that the test systems are not representative of large-scale power networks.

4.1. Tested scenarios and results: fictitious test system

This system consists of 7 different substations and was introduced in Ref. [8]. There are 37 unified switches in total and 27 circuits in the system. The switches (shown by dashed lines) are the phantom switches which are introduced according to rule 5 in Section 4.1. Please note that there are only Type 3 switches used in this fictitious test system. Differently from the original system in Ref. [8], disconnectors are considered in this test system to illustrate the flexibility of the newly proposed algorithm.

Three consecutive switching scenarios are tested. For example, the second switching scenario is carried out immediately after the system reaches a steady state following the changes made by the first scenario. The three test scenarios and their respective bus/branch model are shown in Figs. 14–16. The switch table and configuration matrices for the first scenario are provided in the Appendix. Descriptions of the switching patterns, as well as the results of the topology processor, are shown in Table 3. The computer that is used to execute the topology processor has an Intel® Core™ i7-2600 CPU @ 3.40 GHz.

Please note that the figures do not show the unified switch model. The aim is to demonstrate the changes both in breakers and disconnectors.

4.2. Tested scenarios and results: IEEE reliability test system 1996

This system consists of 24 substations, 90 circuits (66 lines and 24 buses) and 119 unified switches. This test system has all 4 types of switches, and the most common configurations for substations, i.e. One and a Half Switch, Double Bus Double Switch, and Ring. Note that this is a modification of IEEE Realiability Test System 1996. The original version has three areas (all of which are identical); just one of the areas is considered here.

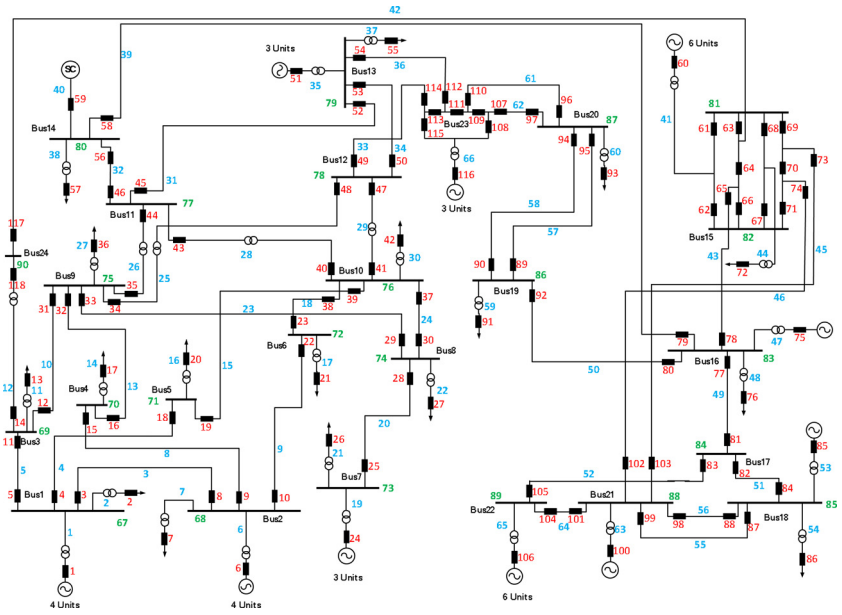

Fig. 21. Third Scenario. Unified switches in red indicate an open status.

As mentioned before, the IEEE Reliability Test System 1996 was simulated in real-time using the eMegaSim Opal-RT real-time simulator. This has provided the possibility of having the simulation outputs as synthetic measurements (both phasors and digital statuses) very similar to PMU data, with a rate of 50 samples per second. As with the fictitious test system, three test scenarios are performed; a description of each scenario and corresponding results are shown in Table 4. Bus/branch diagrams for each scenario are shown in Figs. 17, 20 and 21. Similar to the previous testing cases, the topology processing time is also reported. Note that only the unified switch model is shown in this case.

Due to space limitations, the Bus-section/switching-device model is not shown, however please note that the model has been implemented using this modeling approach. The first scenario is shown using a bus/branch diagram in Fig. 17, and the corresponding synthetic measurements are provided in Figs. 18 and 19. As it can be observed from Fig. 18, 5 different switches are opened simultaneously; Fig. 19 shows the voltage at bus bar 81 (Station 15) and the current through the line 28.

As shown in Tables 3 and 4, the computation time is in the scale of milliseconds. However, the complexity of the test scenarios is considerably high compared to other studies. For simple test scenarios (such as the switching of one single breaker) the computation time is expected to be around several hundred microseconds (this has been verified through multiple tests). In addition, the CPU utilized for the TP engine is a typical 3.4 GHz, and the tests were carried out using MATLAB under the Windows 7 OS. The computation time will further decrease using a production-grade platform for performing TP and SE calculations. In addition, parallelizing the TP execution so that the tasks are separated to independent core (multi-core computation) will significantly decrease the computation time.

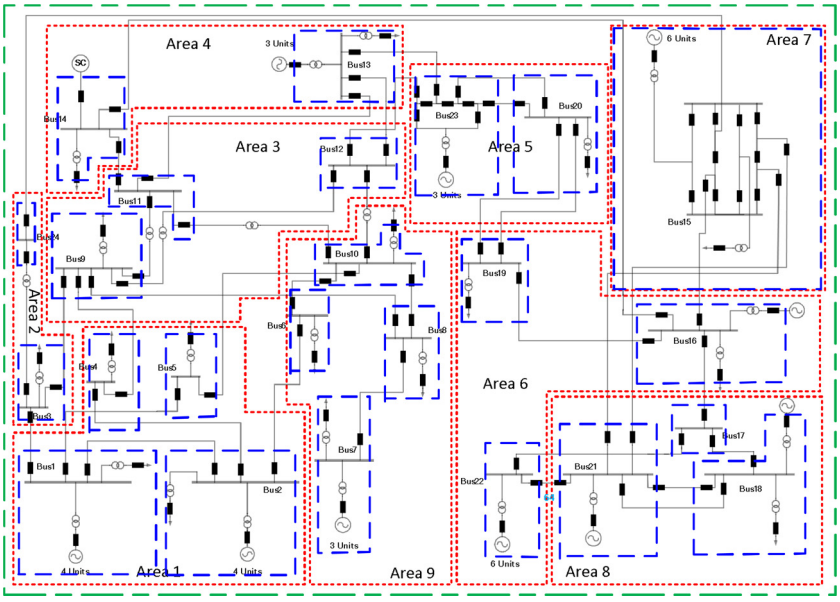

Fig. 22. Example of different power system hierarchies under a partially centralized/ partially decentralized paradigm.

5. Application of the TP in decentralized or partly centralized/partly distributed state estimators

Topology processing is also needed in paradigms for state estimation that consider a decentralized architecture [15] or a hierarchical (two-level) architecture [16–18] to carry out the state estimation process. Thus, these approaches require topology processors that can be executed under two different paradigms: a partially centralized/partially distributed paradigm, or a completely distributed paradigm.

Indeed, the design of the algorithm for topology processing presented above allows for a natural implementation in a partially centralized and partially decentralized paradigm. In the typical centralized paradigm [1], a control center receives data from different substations in the system; afterwards the topology processor analyses all substation configurations sequentially or in parallel (but at a central location) – this step is known as substation analysis. After substation analysis has been carried out, the next steps taken by the TP are to check for islanding and to determine the way that different nodes are connected to each other (referred to as Islanding Analysis and Energization Check).

Using a partially centralized/partially decentralized paradigm, a large-scale network can be divided into several sub-systems with their own hierarchies, as shown in Fig. 22.

The sub-systems at the lowest-level hierarchy are defined by blue boxes and consist of single substations. Substations with the same voltage level and/or geographically vicinity form a subsystem at a higher-level hierarchy, as defined in the red boxes in Fig. 22. Observe that multiple hierarchies can be defined arbitrarily, with a minimum of 2 hierarchical levels. The top-level hierarchy corresponds to the complete power system.

The topology for the sub-systems at the lowest hierarchical levels can be determined at decentralized local computers (e.g. at the same place as the substation is located).

Then the sub-systems at the lowest level hierarchies that are part of a subsystem at a higher hierarchy can send their local topology (determined by the TP running at the local substation) to one of the sub-stations at a higher hierarchical level. The same process can continue with sub-systems at higher hierarchies until the topology of the whole system is determined (possibly at a transmission system operator control center). The TP can also support a local substation SE process as those required in Refs. [17,18]. In this case the topology at the local substation does not need to be transmitted to an upper level hierarchical level, but the result of the SE process does.

In the case of the distributed paradigm, the sub-systems will form their topology locally (at each substation). The resulting local topologies will be shared among each local computer in peer-to-peer fashion. This would allow each local substation to build the entire power system's topology locally (if needed), and can also provide redundancy in data transfer. The HV network control center will receive the topology as well and thus will be able to carry out its conventional function. The TP could instead support a fully distributed SE process as the one suggested in Ref. [15], in which the topology of the local substation will be used strictly in the SE process and topologies from other parts of the network can aid for external network modeling (if needed).

Which paradigm is more suitable for large-scale power networks is not completely clear to the authors. The communication needs for the completely distributed paradigm might be too large in comparison with the centralized architecture, with the benefit of redundant data paths to deliver the topologies.

On the other hand the partially centralized/partially distributed paradigm, such as the one proposed in Refs. [17,18] seems to be a more practical approach. Regardless, communication delays will still be of concern. Future work must first focus on the testing of the topology processor under these paradigms, and afterwards it could focus on optimizing the number of hierarchies and compare it with the completely centralized and completely distributed paradigms.

6. Conclusion

This article has presented a new algorithm for topology processing that can support both conventional and PMU-only state estimators. Key features of this topology processor are: (i) ability to deal with arbitrary substation configurations, (ii) facilities for managing changes within a local substation efficiently, (iii) the possibility of taking into account disconnectors and different switch configurations, (iv) the capability of tracing the TP outputs at different execution cycles, and (v) rapid execution times making it suitable for PMU-based state estimators.

The authors have attempted to provide a rigorous description of the algorithm with dialectics suitable for a translation into computer software. The goal of such presentation approach has been to enable any other researcher to make an independent implementation into computer software of our algorithm. Indeed, the algorithm has been implemented in the MATLAB numerical computing environment and language. The authors will be releasing this implementation as a Free and Open Source Project during 2014, so that other researchers improve the proposed topology processor or build applications that require it. We hope that this implementation can be useful for facilitating research on PMU-data applications that require the determination of the power system's topology.

Acknowledgements

Luigi Vanfretti is supported in part by Statnett SF, the Norwegian transmission system operator, the STandUP for Energy collaboration initiative, and through the *STRONg^{2}rid* project funded by Nordic Energy Research. M. Farrokhabadi was partly supported by the KTH EPS Department.

Appendix A.

Table 5.
Table 6.
Table 7.

Table 5
Switch table matrix for Fig. 14(a).

Switch no.	Near station	Far station	Switch type	Switch status	Circuit no. 1	Circuit no. 2	Original station
1	1	1	3	0	−1	−26	1
2	1	1	3	1	−1	−2	1
3	1	1	3	0	−2	−13	1
4	1	1	3	1	−26	−12	1
5	1	1	3	1	−12	−9	1
6	1	1	3	1	−9	−13	1
7	1	1	3	1	−26	−8	1
8	1	1	3	1	−8	−24	1
9	1	1	3	1	−24	−13	1
10	2	2	3	1	−4	−17	2
11	2	2	3	1	−10	−17	2
12	2	2	3	1	−18	−-17	2
13	2	2	3	1	−10	−16	2
14	2	2	3	1	−16	−18	2
15	3	3	3	0	−3	−5	3
16	3	3	3	0	−6	−5	3
17	3	3	3	1	−1	−3	3
18	3	3	3	0	−1	−20	3
19	3	3	3	1	−6	−20	3
20	4	4	3	1	−9	−27	4
21	4	4	3	1	−27	7	4
22	4	4	3	1	−27	−22	4
23	4	4	3	1	5	−22	4
24	4	4	3	1	−8	−22	4
25	4	4	3	1	21	−22	4
26	5	5	3	1	−4	−15	5
27	5	5	3	1	−11	−15	5
28	5	5	3	1	−4	−14	5
29	5	5	3	1	−11	−14	5
30	6	6	3	1	−3	−19	6
31	6	6	3	1	−2	−19	6
32	6	6	3	1	−19	−10	6
33	1	1	3	1	−23	−6	1
34	1	1	3	1	−23	7	1
35	1	1	3	1	−24	−23	1
36	7	7	3	1	−11	−25	7
37	7	7	3	1	−12	−25	7

Table 6
Configuration matrix for Fig. 14(a).

Circuit no.	Near station no.	Far station no.	Near station no. (redundant)	Near meas. status	Far meas. status	Power flow avail.	Info.	Near station area	Far station area	Orig. near station	Orig. far station
1	1	3	1	1	1	1	1	1	1	1	3
2	1	6	0	1	0	1	0	1	1	1	6
3	3	6	0	1	0	1	0	1	1	3	6
4	2	5	2	1	1	1	1	1	1	2	5
5	3	4	3	1	1	1	1	1	1	3	4
6	3	1	0	1	0	1	0	1	1	3	1
7	4	1	0	1	0	1	0	1	1	4	1
8	1	4	1	1	1	1	1	1	1	1	4
9	1	4	1	1	1	1	1	1	1	1	4
10	6	2	0	1	0	1	0	1	1	6	2
11	5	7	0	1	0	1	0	1	1	5	7
12	1	7	0	0	0	0	0	1	1	1	7
13	1	0	0	0	0	0	0	0	0	1	0
14	5	0	0	0	0	0	0	0	0	5	0
15	5	0	0	0	0	0	0	0	0	5	0
16	2	0	0	0	0	0	0	0	0	2	0
17	2	0	0	0	0	0	0	0	0	2	0
18	2	0	0	0	0	0	0	0	0	2	0
19	6	0	0	0	0	0	0	0	0	6	0
20	3	0	0	0	0	0	0	0	0	3	0
21	4	0	0	0	0	0	0	0	0	4	0
22	4	0	0	0	0	0	0	0	0	4	0
23	1	0	0	0	0	0	0	0	0	1	0
24	1	0	0	0	0	0	0	0	0	1	0
25	7	0	0	0	0	0	0	0	0	7	0
26	1	0	0	0	0	0	0	0	0	1	0
27	4	0	0	0	0	0	0	0	0	4	0

Table 7
Configuration matrix for Fig. 14(b).

Circuit no.	Near station no.	Far station no.	Near station no. (redundant)	Near meas. status	Far meas. status	Power flow avail.	Info.	Near station area	Far station area	Orig. near station	Orig. far station
1	8	9	8	1	1	1	1	1	1	1	3
2	8	9	0	1	0	1	0	1	1	1	6
3	9	6	0	1	0	1	0	1	1	3	6
4	2	5	2	1	1	1	1	1	1	2	5
5	3	4	3	1	1	1	1	1	1	3	4
6	3	1	0	1	0	1	0	2	2	3	1
7	4	1	0	1	0	1	0	2	2	4	1
8	1	4	1	1	1	1	1	2	2	1	4
9	1	4	1	1	1	1	1	2	2	1	4
10	6	2	0	1	0	1	0	1	1	6	2
11	5	7	0	1	0	1	0	1	1	5	7
12	1	7	0	0	0	0	0	2	1	1	7
13	1	0	0	0	0	0	0	0	0	1	0
14	5	0	0	0	0	0	0	0	0	5	0
15	5	0	0	0	0	0	0	0	0	5	0
16	2	0	0	0	0	0	0	0	0	2	0
17	2	0	0	0	0	0	0	0	0	2	0
18	2	0	0	0	0	0	0	0	0	2	0
19	6	0	0	0	0	0	0	0	0	6	0
20	3	0	0	0	0	0	0	0	0	3	0
21	4	0	0	0	0	0	0	0	0	4	0
22	4	0	0	0	0	0	0	0	0	4	0
23	1	0	0	0	0	0	0	0	0	1	0
24	1	0	0	0	0	0	0	0	0	1	0
25	7	0	0	0	0	0	0	0	0	7	0
26	1	0	0	0	0	0	0	0	0	1	0
27	4	0	0	0	0	0	0	0	0	4	0

References

[1] A. Abur, A.G. Exposito, Power System State Estimation: Theory and Implementation, CRC Press, USA, 2004.
[2] L. Vanfretti, J.H. Chow, S. Sarawgi, B.B. Fardanesh, A phasor-data-based state estimator incorporating phase bias correction, IEEE Transactions on Power Systems 26 (2011) 111–119.
[3] A.G. Phadke, J.S. Thorp, K. Karimi, State estimation with phasor measurements, IEEE Transactions on Power Systems 1 (1986) 233–238.
[4] C. Grigg, et al., The IEEE reliability test system-1996. A report prepared by the reliability test system task force of the application of probability methods subcommittee, IEEE Transactions on Power Systems 14 (1999) 1010–1020.
[5] eMegaSim Simulator documentation from OPAL-RT available at: http://www.opal-rt.com/sites/default/files/brochure-eMEGAsim-120507.pdf
[6] A. Monticelli, State Esitmation in Electric Power Systems: A Generalized Approach, Kluwer Academic Publishers, USA, 1999.
[7] IEEE C37.118-2011, IEEE Standard for Synchrophasors for Power Systems (2011).
[8] A.M. Sasson, S.T. Ehrmann, P. Lynch, L.S. Van Slyck, Automatic power system network topology determination, IEEE Transactions on Power Apparatus and Systems PAS-92 (1973) 610–618.
[9] A. Bose, K.A. Clements, Real-time modeling of power networks, Proceedings of the IEEE 75 (1987) 1607–2162.
[10] A. Bose, M. Prais, A topology processor that tracks network modifications over time, IEEE Transactions on Power Systems 3 (1988) 992–998.
[11] C. Qian, Z. Wang, J. Zhang, A New Algorithm of Topology Analysis Based on PMU Information, IEEE 5th International Conference on Critical Infrastructure (CRIS), 2010, pp. 1–5.
[12] R. Kussel, R. Chrustowski, C. Jaborowicz, The Topology Engine: A new Approach to Initializing and Updating the Topology of an Electrical Network, in: 12th PSCC Proceedings, 1996, pp. 598–605.
[13] N. Vempati, C. Silva, O. Alsac, B. Stott, Topology estimation, in: IEEE Power Engineering Society General Meeting, Vol. 1, 2005, pp. 806–810.
[14] M. Farrokhabadi, L. Vanfretti, State-of-the art of topology processors for EMS and PMU applications and their limitatios, in: Proceedings of the 38th annual conference of the IEEE Industrial Electronics Society (IECON), 2012.
[15] G.N. Korres, A distributed multiarea state estimation, IEEE Transactions on Power Systems 26 (1) (2011) 73–84.
[16] T. Van Cutsem, J.L. Horward, M. Ribbens-Pavella, A two-level static state estimator for electric power systems, IEEE Transactions on Power Apparatus and Systems PAS-100 (8) (1981) 3722–3732.
[17] T. Yang, H. Sun, A. Bose, Transition to a two level linear state estimator – Part I: architecture, IEEE Transactions on Power Systems 26 (1) (2011) 46–53.
[18] T. Yang, H. Sun, A. Bose, Transition to a two-level linear state estimator – Part II: algorithm, IEEE Transactions on Power Systems 26 (1) (2011) 54–62.