International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN : 2278-800X, www.ijerd.com
Volume 5, Issue 1 (November 2012), PP. 29-37

Grouping Of Components in an Electrical Network by Zones Using Topological Analysis by DFS and BFS Techniques

Saraín Montero Corzo [1], Nayeli Ramón Lara [2],
Sergio BaruchBarragán Gómez [3]
[123]Instituto Politécnico Nacional, México.

Abstract:- Components and loads grouping in an electrical system has a large range of application in electrical studies and supervised control centers that are used to complete tasks in real time. Locating electrical zones can be useful to get some kind of information such as the amount of load that exists in each zone, energy losses, the identification of electrical resources, faulted zones, and so on. In this work, two kind of topological analyzers are presented, one of them uses DFS (Depth-First-Search) and the other, BFS (Breath-First-Search) in order to detect and create electrical zones for components and loads that are feeded by distribution transformers. The algorithms presented were proved using the case IEEE37 bus test case modified to be able to notice the functionality of these programs.

Keywords:- electrical network, DFS(Depth-First-Search), BFS (Breath-First-Search), topological analysis, electrical zones, graph, digraph, vertex, limit, buses, stack, queue, zone identifier.

I. INTRODUCTION

Areas definition or grouping components in electrical networks has become to be essential in the development of applications. The criteria to generate the area or zone of interest will depend of the kind of application to be needed, for example in [1] electrical zones are determined due to the function of switching devices to evaluate the reliability in an electrical network. To be able to do this task, the use of topological analyzers in electrical networks is essential. A topological analyzer utilizes a network representation that is basically a diagram constructed by a set of points which are joined by lines. This representation is known in mathematics and computer science as graph or digraph. There are two types of methodology to carry out a topological analysis; these methods are Breath-First-Search (BFS) and Depth-First-Search (DFS). Both algorithms are widely used and they have special features that differ one from another, and the choice of which one is going to be used depends on the task of interest. Some applications that employ these algorithms in electrical Engineering are, the identification of faulted devices and sections in the presence of electrical faults [2], [3], and the establishment of optimal paths of energy due to restoration studies [4]. The use given to the topological analyzers in this work is to identify electrical components, which are feeded by transformers, and to group them by electrical zones. Trough Components identification by zones it is possible to get various benefits, such as: Coloring the electrical zones in electrical diagrams on control centers to known the installed load in each zone, to do energy losses countings, to have better control of identification of special loads in the network.

The criterion to make this arrangement is to declare the transformers as elements which isolate electrically by one attribute of logical state. This feature can be employed in planning to know the amount of load lost due to electrical faults, and with these scenarios locate switching devices which satisfy a cost-benefits condition. The only modification would be to change the attribute condition of lines as if they were transformers, and finally to apply the topological analyzer to know the lost load.

In this work are applied BFS and DFS algorithms to show the ability that they have to solve problems in electrical engineering. These algorithms were developed with Visual FORTRAN 2008 program, and were tested with IEEE37 bus test case.

II. TOPOLOGICAL ANALYSIS

In the mathematical-computational language, a Graph is a finite series of points where some of them are joined by lines, and a digraph is a finite series of points where some of them are joined by arrows, creating this what is known as a data tree. Graphs and digraphs are extensively used to represent in an abstract way numerous problems and structures in operations research, computer science, electrical engineering, economics, mathematics, physics, chemistry, communications, game theory, and many other areas. In the Fig. 1 it is shown a digraph example.

Formally, a graph G is a pair (V,E) where V is a finite set whose elements are called vertices and E is a set of subsets of V of order two which represent a combination pair of adjacent vertices [5].The elements of E are called limits. Almost any algorithm that solves a graph problem requires the exploration or processing of each vertex or limit. The way that these algorithms travel trough each vertex gives the name of the method.

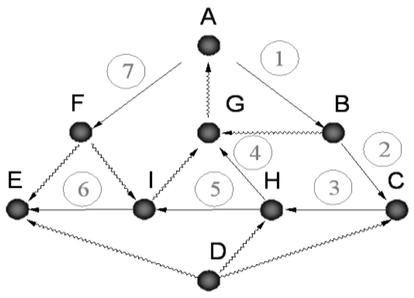


Fig. 1 Digraph example including trajectories made by DFS algorithm

A. Depth-First-Search

The DFS algorithm starts at the beginning of a vertex (which can be selected for a specific problem or in an arbitrary form), and after this point, the analyzer traverses a path as far as possible, visiting and evaluating all the vertices on it, until it reaches a final point. A final point is a vertex such that all the adjacent vertices have already been visited. Once the final point has been reached the return begins and, in every preceding vertex the algorithm makes an analysis of adjacent points that have not been analyzed and if they exist, the analyzer goes through this new path. The process ends when all the vertices trough the graph or digraph have been visited or processed.

The last description can be ambiguous in a way when the analyzer tries to visit two vertices (even more) adjacent to v, which depends of the searching criteria. To assure the nearest vertices to v are visited at first, it must be constructed a queue Q.

Applying the BFS algorithm on the Fig. 1 digraph, the vertex visited sequence is ABCHGIEF. None priority criterion searching was taken so the visiting of each vertex was done in an arbitrary way, but any criterion to prioritize the path selection can be used.

The DFS algorithm is shown next:

Being $G = (V, E)$ a graph or digraph, where $v \in V$ is the vertex where the search starts.

Where:

Q is a stack initially empty. The vertex on the top of Q is referred as the top.

```
1.  Visit and mark $v$ . Insert $v$ in $Q$ .
2.  While $Q$ is not zero, do:
a.  While exists a vertex without visiting $w$ adjacent to the top
b.  Visit, mark $w$ and store $w$ (now $w$ i son top of the stack)
3.  Jump to $Q$ (point 2)
```

B. Breath-First-Search

The BFS algorithm consists in the exploration of each vertex on the same level before changing to another one. The vertices are visited in order according to the increasing distance from the initial vertex, where the distance represents the number of limits in the shortest route.

The main step of BFS is to consider each vertex x of distance d from the initial vertex v and traverse trough all vertices adjacent to x, find an evaluate all the vertices of $d+1$ distance from v. This starts with $d=0$ and is repeated until there are no new vertices.

The BFS algorithm is shown next:

Being $G = (V, E)$ a graph or digraph where $v \in V$ is the vertex from the search starts

Where:

Q is a queue initially empty. "$x \Leftarrow Q$" means to remove from Q the item on the front and store it in x.

```
Visit and mark $v$. Insert $v$ in $Q$
While $Q$ is not zero, do:
"$x \Leftarrow Q$"
For each vertex without a mark $w$ adjacent to $x$ do the next:
Visit and mark $w$
Insert $w$ in $Q$
```

Applying the BFS algorithm on the digraph of the Fig. 3, the sequence of visited vertices is ABFCGEIH. The searching sequence on the same level was done in an arbitrary way, but it can be used some criterion to prioritize the store of each vertex visited in Q.

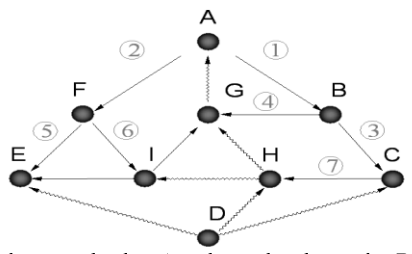


Fig. 2 Digraph example showing the paths chosen by BFS algorithm

III. METHODOLOGY

The proposed algorithm for the identification of zones in an electrical network is shown en the Fig. 3. Following subsections explain the purpose of each block.

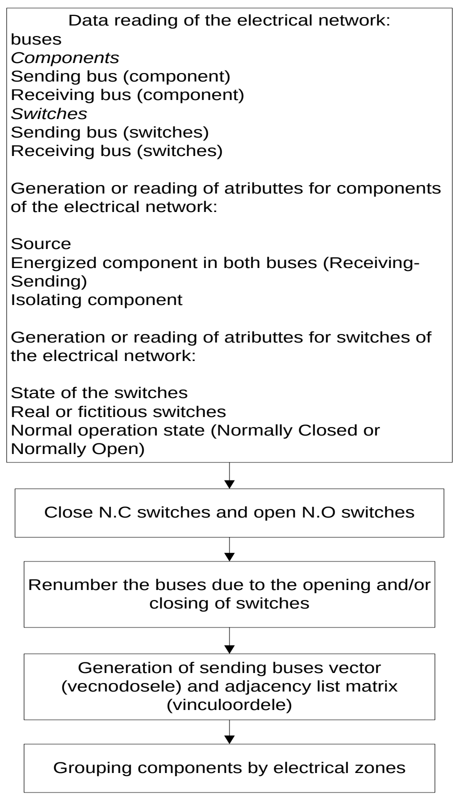

Data reading of the electrical network:
buses
Components
Sending bus (component)
Receiving bus (component)
Switches
Sending bus (switches)
Receiving bus (switches)
Generation or reading of atributtes for components of the electrical network:
Source
Energized component in both buses (Receiving-Sending)
Isolating component
Generation or reading of atributtes for switches of the electrical network:
State of the switches
Real or fictitious switches
Normal operation state (Normally Closed or Normally Open)
Close N.C switches and open N.O switches
Renumber the buses due to the opening and/or closing of switches
Generation of sending buses vector (vecnodosele) and adjacency list matrix (vinculoordele)
Grouping components by electrical zones

Fig. 3 Flow general diagram algorithm

A. Data Reading of the electrical network

This section reads the data of the electrical network. The data are integrated in two files. The first one includes the information of buses, *components*, sending bus (for components), receiving bus (for components), *switches*, sending bus (switches) and receiving bus (switches). This information is declared as character type variables. The *component* notation refers to all components interconnected in an electrical network (transformers, capacitor banks, sources, lines, and so on) excluding disconnection devices. *Switches* notation refers to all disconnection devices in the network (fuses, restorers, blades, and so forth).

The second file includes the attributes of *components* vector (l x 1 dimension) and *switches* (k x 1 dimension), the variable type for these data is logical (0 and 1). The attributes are mentioned next:

- Source
- Energized component in both sides (sending bus and receiving bus)
- Component that isolates electrically
- Switch state (is close or open)
- Real or fictitious
- Normal operation state (N.C or N.O)

B. Close N.C switches and open N.O switches

This program portion closes the switches which have a normally closed operation state, and opens the switches declared as normally opened (load transfer). To do the task previously described it is used the "*state of the switches*" attribute vector. The objective of this task is to leave the electrical network in the normal operation state.

C. Renumber of the buses

This program block is formed by two tasks; the first one renumbers the buses of the electrical network on the switches due the maneuvers over disconnection devices (real or fictitious). The second one uses the gathered information from the first one and renumbers the *components* buses.

D. Adjacency list generation

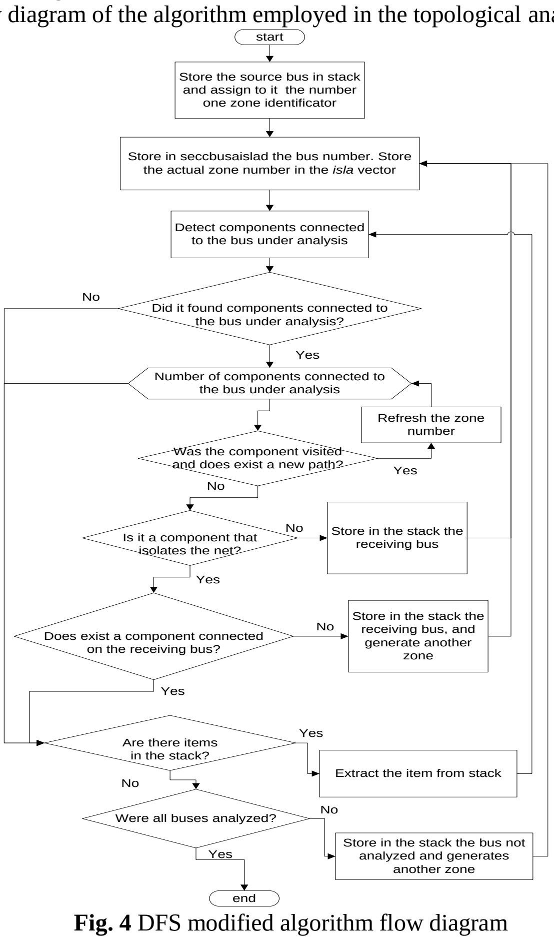
This section generates a components sending buses vector with dimension $n \text{ x } 1$ (it does not include switches). This vector has the name of *vecnodosele* (for reference porpoises). In a parallel form, an adjacent matrix is generated with a $n \text{ x } m$ dimension; this matrix is called *vinculoordele* (for reference porpoises). The adjacent matrix objective is to link each i row of *vecnodosele* vector with the i row of the *vinculoordele* matrix, where each column will have the information of the components which are connected in the same bus. The data stored in a column of *vinculoordele* is the element position inside the *component* vector.

E. Grouping components by electric zones

This program block has the task to group the electrical components through transformers electrical supply zones. The topological analyzers chosen were DFS and BFS. An explanation of each method used is detailed below.

1) Components grouping using DFS,

In the Fig. 4 it is shown the flow diagram of the algorithm employed in the topological analyzer using the DFS technique.



start
Store the source bus in stack and assign to it the number one zone identificator
Store in seccbusaislad the bus number. Store the actual zone number in the isla vector
Detect components connected to the bus under analysis
No
Did it found components connected to the bus under analysis?
Yes
Number of components connected to the bus under analysis
Refresh the zone number
Was the component visited and does exist a new path?
Yes
No
Is it a component that isolates the net?
No
Store in the stack the receiving bus
Yes
Does exist a component connected on the receiving bus?
No
Store in the stack the receiving bus, and generate another zone
Yes
Are there items in the stack?
Yes
Extract the item from stack
No
Were all buses analyzed?
No
Store in the stack the bus not analyzed and generates another zone
Yes
end

Fig. 4 DFS modified algorithm flow diagram

The traverse starts from the sending bus of the supply source. To carry out this, it is important to store in the row $1 \text{ x } 1$ of *vecnodosele* the sending bus of the supply source, since the first vertex that is stored in the stack Q is the element $1 \text{ x } 1$ of *vecnodosele.*

The first element of Q is stored in a vector of visited vertices (*seccbusaislad*), after this, it is included a zone identifier that starts with a number one and it is assigned to the source and is store un a vector named *isla*.

The searching vertex is analyzed and it is located the row in *vinculoordele* where the items different of zero in each column are extracted (vertices adjacent to the searching vertex). After that, each extracted item is evaluated until it founds one that has not been visited. If the sending bus of one of the components connected to a searching vertex has been visited and it is a new traverse (a final point was reached and the stack Q has been left empty) it is done a refresh of the zone identifier of the current traverse to the incident zone identifier. If the current analyzed component has not been visited, it is compared with a step criterion (“*isolate component”* of the vector *components*) in the traverse. If the component doesn’t match the step criterion, it doesn’t get a new zone identifier but the receiving bus is stored in the stack Q and the process is done again.

If the analyzed element matches with the step criterion and it doesn’t have components connected to the reception bus, it gets a new zone number, this is the new searching vertex and it is stored in Q and the process is repeated. The algorithm is designed to consider this behavior like the final point of the actual traverse.

All sending buses evaluated are stored in the stack Q only if there are no components that match the step criterion and the component that isolates does not have connected components in its reception bus. When the traverse reaches a final point, it is extracted from the stack Q a vertex, and it is analyzed to find components connected to it and the process is repeated.

When the stack Q is empty, all electrical network buses are evaluated to check if all of them have been considered. If exists one that has not been considered, it is inserted in the stack Q and it gets a new zone identifier and a new study is carried out.

The search ends when all buses of the electrical network have been visited. The output of the topological analysis the generation of the vector *seccbusaislad* (buses analyzed) and *isla* (zone identifier of each bus).
The algorithm proposed does not include a stack for zone identifiers.

2) Components arrangement using BFS,

In the Fig. 5 it is shown the algorithm flow diagram using the BFS topological analyzer

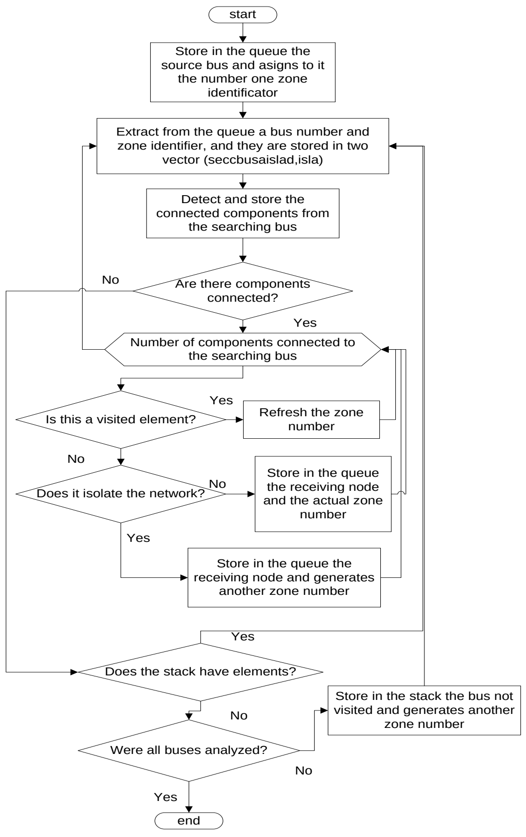

start
Store in the queue the source bus and asigns to it the number one zone identificator
Extract from the queue a bus number and zone identifier, and they are stored in two vector (seccbusaislad,isla)
Detect and store the connected components from the searching bus
Are there components connected?
No
Yes
Number of components connected to the searching bus
Is this a visited element?
Yes
Refresh the zone number
No
Does it isolate the network?
No
Store in the queue the receiving node and the actual zone number
Yes
Store in the queue the receiving node and generates another zone number
Yes
Does the stack have elements?
No
Store in the stack the bus not visited and generates another zone number
Were all buses analyzed?
No
Yes
end

Fig. 5 BFS modified algorith flow diagram

The search begins from the bus source. To carry out this, it must be stored the bus source in the 1 x 1 row of *vecnodosele*, since the first stored vertex in the queue Q is the 1 x 1 element of *vecnodosele*.

The actual element of Q is extracted and stored in a visited vertices vector (*seccbusaislad*). After this, it is created an identifier zone which starts with the number one and it is assigned to the source component and stored in a vector named *isla*. The searching vertex (the extracted element from Q) is used to find the row of *vinculoordele* that includes the components that are connected to that searching vertex and stored in the columns of the *vinculoordele* matrix. Next, each item is extracted from each column of *vinculoordele* matrix and those that are different from zero are analyzed and then compared with a step criterion (“*isolate component”* of the vector *components*) in the search. If the component matches the step criterion, it gets a new zone identifier and the receiving bus and the new zone identifier are stored in queues, one for zone numbers and another for receiving bus and the process is done again for the components extracted. If not, the receiving bus is stored in the queue and it maintains the actual zone identifier that is store in the zone identifier queue.

At the end of the analysis of each bus adjacent to the searching bus, the process is repeated by extracting another vertex stored in the queue. The process ends when all buses of the electrical network are considered. The output of the topological analysis is the generation of *seccbusaislad* vector (visited buses) and *isla* vector (zone identifier vector of each bus).

The algorithms presented here include an additional function which is the consideration of components in the electrical network which its sending bus is not linked to a receiving bus of another component. This case can be seen in the Fig. 1 in the vertex D (for the case of the DFS analyzer). In this situation it can be appreciate that the vertices H, E and C are adjacent to D vertex, but their paths do not point out to vertex D, on the contrary, vertex D point out to them. Due this problem, the searching sequence of DFS traditional algorithm does not consider D vertex as it can be seen by the presented sequence ABCHGIEF.

For the BFS case, it can be seen in the Fig. 2 that the same problem as described for DFS exists. Due to the searching sequence the traditional BFS algorithm does not consider the vertex D, as it can be seen in the given sequence ABFGCEIH. The solution for the incidences to the same vertex was performed by doing comparative tasks about components of the *vecnodosele* against the items of *seccbusaislad* (visited buses). If any bus has not been considered, it is stored in the queue Q and it gets a new zone identifier (a new number that has not been assigned to another electrical zone) and y the searching process is done again.

Another implemented function over the traditional DFS and BFS is the verification of whether the analyzed vertex has been visited yet. This is a fundamental task for the case of incident paths to the same vertex (it has another zone identifier). If the searching vertex is already contained in the *seccbusaislad* vector, it is done a replacement of the zone identifiers (*isla* vector) that have the same number on the actual traverse to the zone identifier that has the found vertex inside the *seccbusaislad* vector. Finally, it is performed a comparative analysis on sending and receiving buses of electrical network *components* with the items stored in the *seccbusaislad* vector. When matches occur the component gets the zone identifier of the coincidence of the *isla* vector.

IV. RESULTS

The algorithms presented were tested with IEEE37 bus test case with modifications for this application. In the Fig. 6 this network is shown. The electrical network was modified to include transformers in some sections; this was done with the purpose to see the arrangement of the components which are supplied for transformers in the distribution network. The output of DFS and BFS algorithms are two vectors called *seccbusaislad* and *isla*. In the first vector the buses analyzed are stored with an appearing order according to the algorithm employed. The second vector (*isla*) keeps the identifier number of each bus stored in the *seccbusaislad* vector. The table I shows the searching sequence of each analyzer and the zone identifier of each bus.

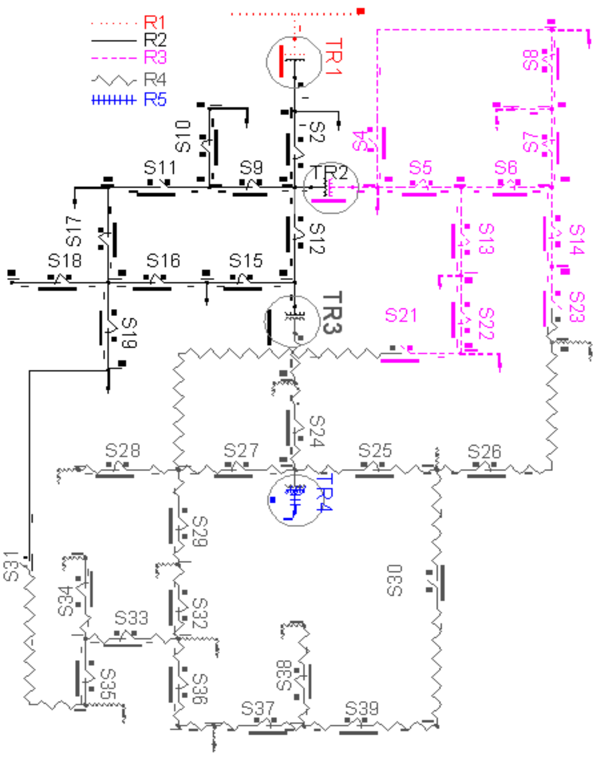


Fig. 6 IEEE 37 bus test case modified

Table I Searching sequence of BFS and DFS

BREATH FIRST SEARCH		**DEEP FIRST SEARCH**	
SEQUENCE (seccbusaislad)	ZONE (isla)	SEQUENCE (seccbusaislad)	ZONE (isla)
1	1	1	1
2	1	2	1
3	2	3	2
4	2	4	2
5	2	5	2
7	2	7	2
8	2	8	2
26	2	26	2
35	2	28	2
9	3	29	2
28	2	31	2
37	2	32	2
10	3	35	2
29	2	37	2
32	2	44	2
44	2	46	2
58	2	47	2
11	3	49	2
13	3	50	2
31	2	34	2
46	2	52	2
59	4	54	2
15	3	55	2
47	2	57	2
60	4	58	2
16	3	9	3
38	3	10	3
49	2	11	3
70	4	13	3
18	3	15	3
40	3	16	3
50	2	18	3
52	2	19	3
55	2	21	3
72	4	22	3
19	3	23	3
41	3	25	3
64	3	41	3
34	2	43	3
54	2	67	3
57	2	38	3
73	4	40	3
78	4	64	3
83	5	66	3
21	3	61	3
43	3	12	3
66	3	33	2
75	4	59	4
63	4	60	4
22	3	70	4
23	3	72	4
67	3	73	4
61	3	75	4
76	4	76	4
80	4	69	4
84	4	78	4
25	3	63	4
69	4	80	4
82	4	82	4
86	4	84	4
91	4	86	4
93	4	91	4
94	4	93	4
103	4	94	4
96	4	96	4
105	4	97	4
97	4	99	4
100	4	100	4
106	4	102	4
99	4	89	4
102	4	103	4
107	4	105	4
89	4	106	4
109	4	107	4
110	4	109	4
113	4	110	4
112	4	112	4
115	4	113	4
87	4	115	4
12	3	87	4
33	2	83	5
62	4	62	4
68	4	68	4
88	4	88	4
90	2	90	2

In Table I it can be seen the searching sequences of each algorithm. In BFS, the searching is spreading trough the network buses from the source bus. For example, from bus 1 to bus 7 (after S2) there are not more than two components connected. From the bus after S2 the receiving buses of three components connected to it are stored in the queue. These components are: LDS3IN (line before TR2, bus 8 to agree with Table II), LDS9IN (line before S9, bus 26 to agree with Table II), and LDS12IN (line before S12, bus 35 to agree with Table II). The buses described are visited and stored in the component visited vector (*seccbusaislad*) in an expansive form, to agree with the BFS. The next step is to consider the buses stored in the queue as sending buses to do the next search which are shown in the Table II, bus 8 for TR2, bus 26 for LDS9IN, and bus 35 for LDS12IN. When the bus 8 analysis is done, it is identified that it has the TR2 connected to it, this component has the *isolate component* attribute (step criterion to assign a new zone identifier) a logical true state, therefore, it can be seen in the Table I and Table II that the receiving bus (bus 9) has assigned another zone identifier.
The next buses that appear in the visited bus vector are 18 and 37 which come from bus 26 for LDS9OUT and bus 35 for LDS12OUT. The sequence is repeated until all buses are analyzed.

The DFS algorithm differs from BFS that it has not a spreading behavior, but it searches into the deepest part of the network until it gets a final vertex. When this point is reached, it comes back by the same path and searches in connection points if there are components not taken into account and in this way the travel trough this new path until it reaches another end point, in this way the process is repeated until there are no more vertices to be visited. This logic can be appreciated in the Table I with the support of the Fig. 6. Where for example, all visited buses after the transformer 2 (TR2) are stored in a sequence (they have the same zone identifier, for this case number 3).

Table II zone assignment for components of the electrical network

component	N.S	N.R	Zone	component	N.S	N.R	Zone
NODTR1IN	1	0	1	LDS22IN	40	64	3
LDS1OUT	3	4	2	LDS22OUT	64	66	3
LDS2IN	4	5	2	LDS23IN	43	67	3
LDS2OUT	5	7	2	LDS23OUT	68	69	4
LDS3IN	7	8	2	LDS24IN	60	70	4
LDS3OUT	9	10	3	LDS24OUT	70	72	4
LDS4IN	10	11	3	LDS25IN	72	73	4
LDS4OUT	12	25	3	LDS25OUT	73	75	4
LDS5IN	10	13	3	LDS26IN	75	76	4
LDS5OUT	13	15	3	LDS26OUT	76	69	4
LDS6IN	15	16	3	LDS27IN	72	78	4
LDS6OUT	16	18	3	LDS27OUT	78	63	4
LDS7IN	18	19	3	LDS28IN	63	80	4
LDS7OUT	19	21	3	LDS28OUT	80	82	4
LDS7OUTA	21	22	3	LDS29IN	63	84	4
LDS8IN	21	23	3	LDS29OUT	84	86	4
LDS8OUT	23	25	3	LDS30IN	115	87	4
LDS9IN	7	26	2	LDS30OUT	88	75	4
LDS9OUT	26	28	2	LDS31IN	102	89	4
LDS10IN	28	29	2	LDS31OUT	90	57	2
LDS10OUT	29	31	2	LDS32IN	86	91	4
LDS11IN	28	32	2	LDS32OUT	91	93	4
LDS11OUT	33	34	2	LDS33IN	93	94	4
LDS12IN	7	35	2	LDS33OUT	94	96	4
LDS12OUT	35	37	2	LDS34IN	96	97	4
LDS13IN	15	38	3	LDS34OUT	97	99	4
LDS13OUT	38	40	3	LDS35IN	96	100	4
LDS14IN	18	41	3	LDS35OUT	100	102	4
LDS14OUT	41	43	3	LDS36IN	93	103	4
LDS15IN	37	44	2	LDS36OUT	103	105	4
LDS15OUT	44	46	2	LDS36OUTA	105	106	4
LDS16IN	46	47	2	LDS37IN	106	107	4
LDS16OUT	47	49	2	LDS37OUT	107	109	4
LDS17IN	49	50	2	LDS38IN	109	110	4
LDS17OUT	50	34	2	LDS38OUT	110	112	4
LDS18IN	49	52	2	LDS39IN	109	113	4
LDS18OUT	52	54	2	LDS39OUT	113	115	4
LDS19IN	49	55	2	TR1	2	3	1,2
LDS19OUT	55	57	2	TR2	8	9	2,3
LDS20IN	37	58	2	TR3	58	59	2,4
LDS20OUT	59	60	4	TR4	72	83	4,5
LDS21IN	66	61	3	LDS1IN	1	2	1
LDS21OUT	62	63	4				

It can be seen in the Table I that the buses 33 and 90 (with a zone identifier 2) do not appear in the searching sequence when the zone 2 is examined. This is because the distribution lines LDS11OUT (connected to bus receiving of S11) and LDS31OUT (connected to bus receiving of S31), are used to do load transfer, therefore when the analyzer reaches these components it does not find the sending bus, but the receiving bus, and in this way, the analyzer cannot continue with the analysis. The mentioned buses appear when the analysis of a path has been done and the sending bus vector named *vecnodosele* has been analyzed to assure that all buses has been taken into account in the study. Once the zone identifiers for all buses were gotten, it was carried out an assignment of zones for electrical components. This assignment can be seen in the Table II. In the Fig. 6 each component is colored according to the zone identifier assigned.

V. CONCLUSIONS

The utilized analyzers to arrange electrical components feeded by specific transformers trough zone identifiers gave satisfactory results at testing them with the IEEE37 bus test case modified. These tools offer the possibility to identify electrical components and loads in a graphic form. With the zone identifier it can be done the account of load that a transformer provides, as well as the electrical losses of each zone. Another application that can be done with these tools is to change the step criterion for some lines to be driven as transformers, with this attribute change, it could be analyzed the benefit-cost that would be expected when switching devices are inserted in the network to isolate electrical faults.

REFERENCES

[1]. Yufeng Huang, Zongqi Liu, Wenxia Liu, Jianhua Zhang, "Design and Implementation of Reliability Evaluation Module for Medium Voltage Distribution Networks by Objectarx Technology", Advances in Power System Control, Operation and Management (APSCOM 2009), 8th International Conference on.

[2]. ZHANG Yagang, ZHANG Jinfang, MA Jing, WANG Zengping, "Fault Diagnosis Based on BFS in Electric Power Syste", Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on

[3]. QIN Lijun, HAO Cuijuan, JIN Huawei, LI Meng, WANG Ying, "CIM-Based Fault Detection Analysis of Distribution System", 2011 The International Conference on Advanced Power System Automation and Protection.

[4]. P. Ravi Babu, K.Vamshi Krishna, W.Shirisha, P. Naga Yasasvi,"Heuristic Search Strategy for Service Restoration Using DFS and BFS Techniques", Power Electronics (IICPE), 2010 India International Conference on.

[5]. Sara Baase, Computer Algorithms: Introduction to Design and Analysis, Addison-Wesley Publishing Company, 1978,ch.3.

Authors:

Saraín Montero Corzo. M.Sc. in Electrical Engineering from SEPI-ESIME-IPN, México in 2006. Electrical Engineer graduated from Instituto Tecnológico de Tuxtla Gutiérrez, México in 1998. Nowadays is an associated professor at ESIME-Zacatenco. The interest areas for him are Electrical System analysis for Power and Distribution, Power Electronics and Energy Quality.

Nayeli Ramón Lara. Is an associated professor at the ESIME-IPN. M.Sc. in Electrical Engineering at SEPI-ESIME-IPN in 2006. Electronics and Communications engineer graduated from Instituto Tecnológico y de Estudios Superiores de Monterrey en 2002. The interest areas for her are, Electrical Machines Control, Power Electronics and Education.

Sergio Baruch Barragán Gómez. Titular professor level A from ESIME-IPN. M.Sc. in Electrical Engineering from SEPI-ESIME-IPN in 2004. Electrical Engineering from ESIME-IPN en 2001. The Interested areas for him are free Software, Analysis and Economic Operation of Electrical Systems.