International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:5, No:2, 2011

Distributed Load Flow Analysis using Graph Theory

D. P. Sharma, A. Chaturvedi, G.Purohit , R.Shivarudraswamy

***Abstract*—**In today scenario, to meet enhanced demand imposed by domestic, commercial and industrial consumers, various operational & control activities of Radial Distribution Network (RDN) requires a focused attention. Irrespective of sub-domains research aspects of RDN like network reconfiguration, reactive power compensation and economic load scheduling etc, network performance parameters are usually estimated by an iterative process and is commonly known as load (power) flow algorithm. In this paper, a simple mechanism is presented to implement the load flow analysis (LFA) algorithm. The reported algorithm utilizes graph theory principles and is tested on a 69- bus RDN.

***Keywords*—**Radial Distribution network, Graph, Load-flow, Array.

I. INTRODUCTION

GENERALLY, radial distribution system (RDS) has high R/X ratio. Due to this conventional load flow algorithm [1,2] shows convergence problem. Some researchers try to address high R/X ratio of RDN and suggested some modified load flow algorithm. However it fails to converge for some cases. Baran and Wu [3], proposed a Newton rapson based load flow analysis, but it requires large number of computation due to involvement of Jacobin matrix. In most of the reported studies so far in literature are based upon forward-backward sweep mechanism. Moving from substation to towards leaf is called forward approach and moving form leaf to towards substation is called backward approach. According to various reported studies; these two mentioned schemes used by many researchers. According to Nanda et. al reported work [4], branch current is calculated by using reverse sweep approach and a separate reverse sweep is required for individual branch, so this repetition shows the inefficiency of this method. Mok et. al suggested [5] that, in order to calculate a branch current, reverse sweep is used till the concern branch arrives. Arvindhababu et. al have [6] used matrix with reverse sweep for the branch current computation. So in this approach the use of matrix gives improper use of memory. Venkatesh et.al reported load flow study [7], in this reported work author exploit tree like structure of RDN with efficient use of dynamic
datastructure. However, it uses so many recursive calls for voltage calculation makes it inefficient. Prasad et al suggested a simple algorithm for load flow Analysis [8]. In this work, author exploits the properties of tree with efficient use of data structure. However, it is based upon an inefficient procedure for finding leaf node, and same procedure when used repeatedly in convergence loop, further made it more inefficient. The present paper describes a new efficient load flow algorithm. Due to tree type structure of RDN, it can be modeled as graph and finally adjacency list is used for the efficient representation of RDN in computer memory.
This proposed algorithm built individual leaf node structure for each leaf (terminal) node present in the RDN network. In this proposed work, authors have used the mathematical treatment for load flow analysis as given in [8].

D.P.Sharma is with the National Institute of Technology Karnataka, Surathkal- 575025, India; (e-mail: dps158@ rediffmail.com).
A.Chaturvedi, is with the National Institute of Technology Karnataka, Surathkal- 575025, India; (e-mail: ashvini@nitk.ac.in).
G.N.Purohit is with the Banasthali University, Rajasthan-304022, India (e-mail: gn_purohitjaipur@yahoo.co.in).
R.Shivarudraswamy is with the National Institute of Technology Karnataka, Surathkal- 575025, India; (e-mail: swamysrs@ rediffmail.com).

II. LOAD FLOW ANALYSIS DESIGN

Based upon graph –theoretical approach, the implementation of proposed algorithm is presented on modular basis in various subsections. Initially, a given RDN is represented as a directed weighted graph G (V, E) with set of vertices (V) and edges (E); the modeling of a RDN as an equivalent graph is presented in section A. The proposed LFA algorithm makes use of BFS algorithm [9] to create two arrays, here after referred as Leaf Node Structure (LS) array and Universal Junction node (UJ) array and the formulation of these two arrays are discussed in section B. Using reduction module, the molded graph (network) is further reduced into more simple and small graph and is presented in section C. The proposed LFA algorithm's pseudo code is reported in section D, which calls reduction module repeatedly until the RDN transformed into a single branch (line) network.

A.*Graphical Modeling Of Rdn*

The substation, load buses (nodes) and branches of a RDN are represented by vertices (V) and edges (E) respectively in modeled directed weighted Graph, G. The index (name to edge) assign to the edges of the graph is same as the corresponding branch number in a RDN. Generally, two approaches are used for graph representation, and these are adjacency matrix and adjacency list. In this proposed work, approach based on adjacency list is used. For N- node RDN, adjacency list is an array of length N denoted by Adj [V] , where V=1,2…..N. The each element of array Adj [V], represents a vertex of graph G, points to a list of its adjacent nodes. Each node in the list corresponds to one successor of vertex V and every node has four fields. The first field of the node represented by A, stores node adjacent to the node V. The second field represented by B and third field represented by C stores index (label) assigned to the incoming and

outgoing edge, corresponding to the node V respectively. When V represents a main substation then the value of field B is zero, as there is no incoming edge and using the same logic way if V represents leaf (terminal) node then the value of C is zero, as there is no outgoing edge from V .The fourth field denoted as D points to the next node if any more node is adjacent to V, otherwise it stores a null value. Adjancey list representation of one example RDN (Fig. 1a) is shown in Fig. 1b. In Fig 1b, for node number 2, i.e. Adj [2].B is x1 and also Adj [2].C is x7 and x2. Although, on traversing towards leaf nodes either of these two nodes, i.e. x2 or x7 is considered as a successor at a time.

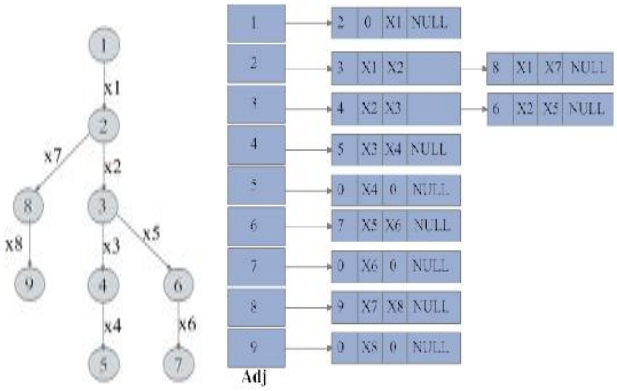

Fig. 1(a) Fig. 1(b)

B Leaf Node Structure and Universal Junction Nodes

BFS algorithm is used in the proposed scheme to estimate LS and UJ for a given RDN (Graph). The definition of LS and UJ arrays are defined in the following section.

B.1. LEAF NODE STRUCTURE

LS is a record which has following four fields.

(a) Leaf node says L.

(b) Path of the leaf node (the path from leaf node to the root node) i.e. An array of path vertices say P.

(c) It is an array of those nodes/vertex in current path P whose out degree is more than one say J.

(d) Branch Current for which leaf act as source i.e. say Cleaf whose initial value is zero.

For example, if we apply BFS algorithm for fig1a , it returns three LSs corresponding to each leaf. The details of all the field of one of the LS say for leaf 7 are as follows:

(1) L=7; (2) Path =7□6□3□2□1 i.e. P= [7, 6, 3, 2, 1]

(3) J= 3, 2 (4) Cleaf=0

B.2. Universal Junction Nodes

It is an array of nodes whose outgoing degree is more than one in the whole network. So BFS will return array UJ=[3, 2].

C. Reduction

The main objective of this reduction is to reduce the number of Leaf node structures (LSs). This is done by selecting two or more leaf node structures whose junction nodes are common i.e. it is a function which takes two or more Leaf node structures as input, based upon common junction nodes and reduces into a single node structure with creating a virtual leaf node. All the four attributes of leaf node structure, for this virtual leaf is same as described in section B.1. To understand the reduction process, a graph is taken with nine nodes as shown in the Fig.1a .Initially the three LS are identified corresponding to each leaf present in the Graph G. In the first iteration of the reduction process, two leaf structures corresponding to common junction node say 3 in our graph are identified as an input to reduction process. To reduce the graph, it will take one LS at a time and compute branch currents from the leaf node to the node whose level is one less than the junction node. This process is repeated in for all other remaining input LSs. The next step is to sum up all the branch currents that lie beneath the junction node. The node 3 in the given Fig.2 is called as Virtual leaf node as it is obtained by the reduction process whose brief algorithm is stated below

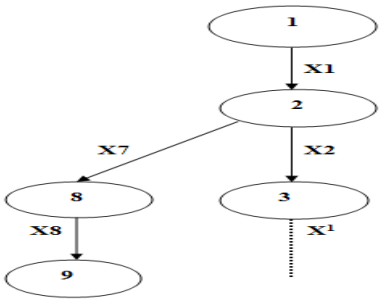

Fig.2

REDUCTION (Leaf Structures, UJ)

// i=It is a variable that tell current considered LNS such that $1 \leq i \leq$ No_of _LNS pass as an input for the reduction process. So LNS_i will represent different LNS for different values of i.

// j=It work as array index for P attribute in selected LNS. So LNS_iP_j will give one of the vertex in Graph corresponding unique combination of values of variables i and j.

// I_{BR} and I_L represents the branch and load currents.

// *Temp* variable used in summing up all the branch currents that lie beneath the junction node.

```
Temp=0;
for i=1 to no _of_ Leafstructure
{
  for j=1 to no_of_element_in_LNSiP
  {
     if(LNSiPj+1 ==UJ)
        break;
     If (j==1) //\\ current of branch for which leaf act as a
     sink
        {
```

```
        IBR (Adj[LNSiPj].B)=IL(Adj[LNSiPj]) + LNSiCleaf;
      }
     else
      {
       IBR (Adj[LNSiPj] .B) =IL(Adj[LNSiPj]) + IBR
(Adj[LNSiPj] .C) ;
      }
   } //\\ end of Inner for loop
//\\ following code used for summing up all the branch
currents that lie beneath the junction node.

If (j==1) //\\ current of branch for which leaf act as a sink
       {
        IBR (Adj[LNSiPj].B)=IL(Adj[LNSiPj]) + LNSiCleaf;
  Temp=Temp+ IBR (Adj[LNSiPj] .B)
       }
     else
      {
       IBR (Adj[LNSiPj] .B) =IL(Adj[LNSiPj]) + IBR
(Adj[LNSiPj] .C) ;
       Temp=Temp+ IBR (Adj[LNSiPj] .B)
       }
 }//\\ end of outer for loop
```

D. Code for Distributed Load Flow Analysis

1. Initialization of Variables
   V[i] = V_{old}[i] = 1.0 for i=2… N; // Initial guess for the node voltages of RDS
   I_{BR} [j] = 0 for j= 1,…,N_b //Initial guess for the branch currents; N_b = N-1
2. Calculate Leaf node Structures and Universal Junction node i.e. J by Running BFS on RDN/Graph with starting root.i.e from the Main station of RDN.
3. Select LNSs for reduction with the help of UJ.
4. Call Reduction function with Leaf Node structures identified at in previous steps.
5. Built the LNS for virtual Leaf Node and then update the no of LNS, UJ.
6. If no of Leaf Node Structure ≥2 then go to Step 1
7. Calculate the branch currents in last remaining LNS.
8. Calculate node voltage and check convergence condition defined as in [8].If convergence condition satisfied then stop otherwise goto step 3.

III. Simplified Complexity Analysis Of Algorithm

A simplified Analysis of Time and space complexity of Algorithm is briefed as:

A.*Time Complexity*: The proposed load flow analysis has been described in 8 steps. The steps No. 1,2,3,5,6, and 8 are representing constant time operation. Remaining steps 4 and 7 are liner time operation. So the overall time complexity for above discussed Load flow analysis method is order of O (n).

B.*Space Complexity*: The Adjacency List stores data about nodes and edges. Since there are no recursions and dynamic memory allocations, it is apparent that the space complexity is linear.

IV. Test Result

To test the efficiency of the proposed algorithm, 33-bus and 69 bus RDN are considered. System data for 33 bus and 69 bus system are available in [6] and [8]. The proposed algorithm took 300 and 800 steps execution for 33 bus and 69 bus system RDN system respectively for converge solution. Table 1 illustrates the computational efficiency of the proposed LFA and Table 2 represents converge voltage of 69 RDN system.

TABLE I
COMPARISON OF STEPS COUNTS BETWEEN THE PROPOSED ALGORITHM (♣) AND THE ALGORITHM PROPOSED IN [8]

Method	33-bus RDN	69-bus RDN
Proposed	300 approx.	800 approx.
Prasad[8]	9600 approx.	42500 approx.

V. Conclusion

In the proposed idea, concept of graph theory is been utilized for better performance. The proposed LFA algorithm utilizes adjacent list concept for representing RDN. Observation done from the result stored in Table 1 indicating the remarkable saving in the steps of execution to get converge load flow solutions. It is also worth mentioning that the savings further improves when the network's size increases. The time complexity of proposed LFA is order n i.e. O (n).

References

[1] W.F Tinny and C.E. Hart, "Power flow solution of the Newton Method", IEEE Transactions, PAS-86, 1967, pp. 1449-1456.
[2] B. Scott and O. Alsac, "Fast decoupled load flow", IEEE Transactions, PAS-93, 1974, pp. 859-869.
[3] M.E. Baran and F. F.Wu, "Optimal capacitor placement on radial distribution systems", IEEE Transactions on Power Delivery, Vol. 4, No. 1, 1989, pp. 725-734..
[4] J. Nanda, M.S. Srinivas, M. Sharma, S.S. Dey, L.L. Lai, "New findings on Radial Distribution system load flow algorithms", Proc. IEEE conference, 2000, pp. 1157-1161.
[5] S. Mok, S. Elangovan, C. Longjian, M. M. A. Salama, "A new approach for power flow analysis of balanced distribution systems" Electric Machines and Power Systems, Vol. 28, 2000, pp. 325-340.
[6] P. Aravindhababu, S. Ganapathy, K.R.Nayar, " A novel technique for the analysis of radial distribution systems", International Journal of Electrical Power and Energy Systems, Vol. 23, 2001, pp. 167-171.
[7] B. Venkatesh and R. Ranjan ," Data structure for radial distribution system load flow analysis ", IEE Proc.-Gener. Transm. Disirih. Vol. 150. No 1. Junuory 2003.
[8] K. Prasad, N. C. Sahoo, A. Chaturvedi and R. Ranjan "A simple approach in branch current computation in load flow analysis of radial distribution systems", IJEEE - International Journal for Electrical Engineering Education, Vol.44/1, PP: 49-63, January 2007.
[9] Cormen T. H., C.E. Leiserson, and R.L Rivest, Introduction to Algorithms,Prentice-Hall of India Pvt.Ltd.,New Delhi,2001

TABLE II
COMPARISON OF LOAD FLOW RESULTS FOR 69-NODE RDN BETWEEN THE PROPOSED ALGORITHM (♣) AND THE ALGORITHM PROPOSED IN [8]

Node Number	Voltage Magnitude (♣) (p.u.)	Voltage Magnitude [8] (p.u.)	Node Number	Voltage Magnitude (♣) (p.u.)	Voltage Magnitude [8] (p.u.)
1	1.00000	1.00000	36	0.99992	0.99992
2	0.99997	0.99997	37	0.99975	0.99975
3	0.99993	0.99993	38	0.99959	0.99959
4	0.99984	0.99984	39	0.99954	0.99954
5	0.99902	0.99902	40	0.99954	0.99954
6	0.99008	0.99009	41	0.99884	0.99884
7	0.98079	0.98079	42	0.99855	0.99855
8	0.97857	0.97858	43	0.99851	0.99851
9	0.97744	0.97744	44	0.99850	0.99850
10	0.97243	0.97244	45	0.99841	0.99841
11	0.97131	0.97132	46	0.99840	0.99840
12	0.96814	0.96816	47	0.99979	0.99979
13	0.96521	0.96523	48	0.99854	0.99854
14	0.96231	0.96233	49	0.99469	0.99470
15	0.95943	0.95946	50	0.99415	0.99415
16	0.95890	0.95893	51	0.97854	0.97854
17	0.95802	0.95805	52	0.97853	0.97853
18	0.95801	0.95804	53	0.97465	0.97466
19	0.95754	0.95757	54	0.97141	0.97141
20	0.95724	0.95727	55	0.96693	0.96694
21	0.95676	0.95679	56	0.96256	0.96257
22	0.95675	0.95678	57	0.94004	0.94010
23	0.95668	0.95671	58	0.92894	0.92904
24	0.95652	0.95656	59	0.92464	0.92476
25	0.95635	0.95638	60	0.91958	0.91974
26	0.95628	0.95631	61	0.91217	0.91234
27	0.95626	0.95629	62	0.91188	0.91205
28	0.99993	0.99993	63	0.91149	0.91167
29	0.99985	0.99985	64	0.90958	0.90977
30	0.99973	0.99973	65	0.90901	0.90919
31	0.99971	0.99971	66	0.97125	0.97126
32	0.99961	0.99961	67	0.97125	0.97126
33	0.99935	0.99935	68	0.96781	0.96783
34	0.99901	0.99901	69	0.96781	0.96782
35	0.99895	0.99895			