Electric Power Systems Research 147 (2017) 145–153

Contents lists available at ScienceDirect

Electric Power Systems Research

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

Graph Trace Analysis: An object-oriented power flow, verifications and comparisons

Ahmad Tbaileh [a,*], Himanshu Jain [a], Robert Broadwater [a], Jose Cordova [b], Reza Arghandeh [b], Murat Dilek [c]

[a] ECE Department, Virginia Tech, Blacksburg, VA 24061 USA
[b] ECE Department, Florida State University, Tallahassee, FL 32310 USA
[c] Electrical Distribution Design, 820 University City Blvd, Blacksburg, VA, 24060, USA

ARTICLE INFO

Article history:
Received 12 October 2016
Received in revised form 27 February 2017
Accepted 28 February 2017
Available online 7 March 2017

Keywords:
Generic programming
Graph Trace Analysis
Integrated T&D
Load flow

ABSTRACT

This paper presents an alternative approach to power system computations, Graph Trace Analysis (GTA), and applies this approach to solving the power flow problem. GTA is derived from the Generic Programming Paradigm of computer science, and uses topology iterators to move through components in a model and perform calculations. The implementation of KVL and KCL in GTA is described. The GTA based power flow algorithm is shown to solve IEEE standard transmission models, IEEE standard distribution models, and integrated transmission and distribution models (hybrid models) constructed from modifying IEEE standard models. Comparisons with widely used, primarily matrix based, power flow algorithms are provided. Two advantages that GTA brings are the separation of system equations from component equations and the ability to distribute calculations across processors.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

In power systems it is common to use different algorithms to solve the power flow problem for transmission and distribution systems. Algorithms used for solving transmission networks apply matrix based algorithms [1,2]. These algorithms work well for meshed networks where transmission lines have small R/X ratios. However, they have poor convergence when the system is heavily loaded, and with systems that have large R/X ratios, such as distribution systems. In addition, while convergence rates are excellent for these algorithms under normal operating conditions, they require a Jacobian matrix to be factorized at every iteration. Also, convergence may not occur if initial conditions for the power flow algorithm are not sufficiently close to the solution, which can often be the case for extreme operating conditions.

For distribution networks, transmission network algorithms show slow to no convergence [3,4]. This is due to the radial, or lightly meshed, nature of the network, and large R/X ratios. Distribution networks are unbalanced in many ways—number of phases, unbalanced construction, unbalanced loading, and unbalanced distributed generation. Power flow algorithms used for distribution networks are different from transmission system algorithms. The forward/backward sweep method for distribution networks has been used by many distribution power flow algorithms [5].

Based on the sweep method, power flow solution methods have also been developed for lightly meshed networks [6,7]. With these algorithms the system is broken into radial networks and solved using sweep-based methods. A compensation technique is then used to solve the mesh equations. However, these algorithms become inefficient for heavily meshed systems. Improvements have been developed for this method by constructing a sensitivity matrix that saves time in execution [7,8].

In this paper an alternative approach for power systems computations, Graph Trace Analysis (GTA), is described and used to solve the power flow problem. GTA and topology iterators have been used for over 20 years to develop many different algorithms [9]. GTA based power flow algorithms have been implemented that solve transmission, radial distribution, lightly meshed distribution, and heavily meshed distribution, all in the same model [10]. With the growth of generation resources at the distribution level, power flow solutions of hybrid, or integrated transmission and distribution models, are needed [11,12], including the ability to solve hybrid models that include detailed substation models and even downtown network models.

It has been demonstrated that GTA algorithms can solve multi-domain models, and that the same algorithm can be used to analyze

* Corresponding author.
E-mail address: atahm12@vt.edu (A. Tbaileh).

http://dx.doi.org/10.1016/j.epsr.2017.02.034
0378-7796/© 2017 Elsevier B.V. All rights reserved.

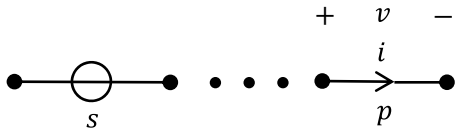

Fig. 1. Voltage and current conventions for GTA model, where *S* is the reference source for component with current *i*.

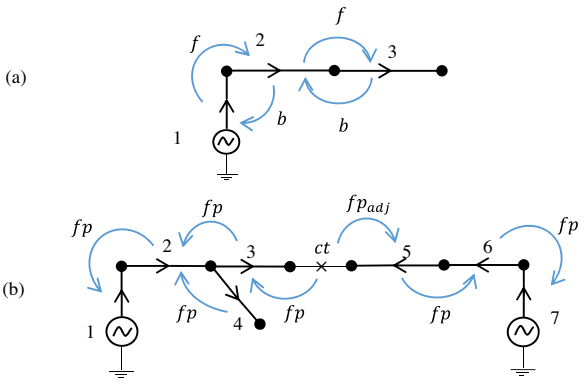

Fig. 2. Basic circuits. Part (a) shows a radial circuit, and (b) shows a looped circuit.

across different engineering domains [13]. These abilities derive from the generic programming architecture that GTA is based on, and the object oriented approach used in existing GTA algorithms, where system equations can have no knowledge of the internal behavior of the components.

Furthermore, the implementation of GTA relies on traces, which allows GTA algorithms to be naturally distributed across multiple processors by just distributing the model. Thus, with distributed computations there is no need for a master processor or parallel processing computer equipment [14].

This paper provides a description of GTA topology iterators and graph traces, including the implementation of Kirchhoff voltage (KVL) and current (KCL) laws. Then the GTA power flow algorithm is compared to other power flow solvers. Power flow solutions of hybrid models, including models with reverse distributed generation flow from distribution to transmission, are provided.

The paper is organized as follows. In Section 2 the concept of GTA topology iterators is introduced. In Section 3 the application of GTA to circuit analysis is discussed, while Section 4 presents a simple example of a GTA based power flow algorithm. GTA power flow solutions of IEEE standard transmission models, IEEE standard distribution models, and models that include both transmission and distribution, are presented in Section 5.

2. GTA and topology iterators

In GTA each component p has one, and only one, reference source. Each component p has a measurement reference which is defined away from its reference source. Thus, current flow i is positive if it is flowing away from its reference source s. This is illustrated in Fig. 1.

GTA traces and trace sets are now illustrated. In describing traces each edge is identified by an integer that is unique within the model. The integer identifiers, as shown in Fig. 2, are placed on top of the edges of the model. Table 1 defines key trace sets. The forward and backward traces are used to trace through every edge referenced to the same source, where each edge referenced to the source is only included in a forward or backward trace once.

The forward and backward traces for the circuit in Fig. 2a can be defined as $FT = \{1, 2, 3\}$ and $BT = \{3, 2, 1\}$, respectively. For

Table 1
Definition of fundamental GTA trace sets.

Trace Set	Definition
FTi	Ordered set created with recursive application of forward iterator *f* starting at component *i*; if *i* is not specified, then the forward trace starts at the reference source under consideration
BTi	Ordered set created with recursive application of backward iterator *b* starting at component *i*; if *i* is not specified, then the backward trace starts at the ending component associated with the reference source under consideration
FPi	Ordered set created with recursive application of feeder path iterator *fp* starting at component *i*
LTct	Ordered set created by the union of the recursive application of *fp*, starting at the cotree edge for *ct*, with recursive application of *fp*, starting at the adjacent edge for *ct*

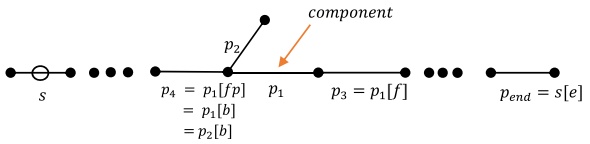

Fig. 3. Illustration of topology iterators.

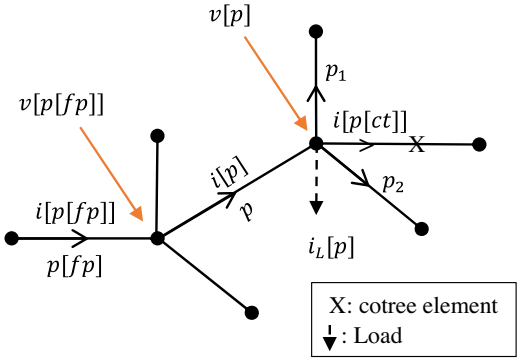

Fig. 4. Implementation of KVL and KCL with GTA.

Fig. 2b the feeder path traces for components 3, 4, and 5 are $FP_3 = \{3, 2, 1\}$, $FP_4 = \{4, 2, 1\}$, and $FP_5 = \{5, 6, 7\}$, respectively. The loop trace for the cotree element of Fig. 4b is created by the union of the feeder path traces of the components connected at the cotree, and is given by $LT_3 = \{3, 2, 1, 5, 6, 7\}$.

We now define the concept of a topology iterator. A topology iterator for a component p returns a component that is topologically related to component p. Four commonly used iterators are:

- $p\,[f]$: forward topology iterator for component p that returns the component in the forward trace direction
- $p\,[b]$: backward topology iterator for component p that returns the component in the backward trace direction
- $p\,[fp]$: feeder path topology iterator for component p that returns the feeder path component for p, a component that is physically connected to p and which supplies power to p from the reference source for p
- $p\,[ct]$: cotree topology iterator for component p that returns the cotree component associated with p, a component that is physically connected to p and which, if removed from the graph, breaks an independent loop.

Thus, $p\,[f]$ and $p\,[b]$ represent the doubly linked list of computer science. $p\,[fp]$ and $p\,[ct]$ provide physical connectivity information for p. The forward and backward traces may be used to process through all the components in the model, and when calculations are

Download English Version:

https://daneshyari.com/en/article/5001180

Download Persian Version:

https://daneshyari.com/article/5001180

Daneshyari.com