Transitive Closure Using Path Algebra Algorithm

Md. Yasin Ali Khan[1] and Md. Abu Sayed[2]

[1]Department of Computer Science and Engineering,
Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh

[2] Department of Computer Science and Engineering,
American International University-Bangladesh, Banani, Dhaka-1213, Bangladesh

Abstract*—computing transitive closure of graphs is a well-studied area of computer science. In literature different researchers applied different techniques for finding transitive closure of some arbitrary graph. There also exist some good algorithms dealing with this kind of problem like Warshall's algorithm etc. Finding transitive closure in some graph has numerous application starting from computer network design to analysis of molecular interaction. In this paper problem of finding transitive closure of graph is formulated in reach-ability matrix form. Here basic problem of finding transitive closure is dealt with Max-Plus Idempotent semi-ring and Idempotent Mathematics. A simple algorithm is described that can calculate transitive closure of graphs using Path Algebra or Max-Plus Algebra. Simple methods of linear algebra is used for different level of computation.

Keywords: transitive closure, Max-Plus algebra, algorithm.

I. INTRODUCTION

Generic Programming needs generic algorithm and easy to use data structure. For making robust software within time bound generic algorithms may be used to reduce reinvention of code[1]. In search for generic algorithms researchers have found linear algebra algorithms used on idempotent semi-rings. For example, all dynamic programming algorithms can be reformulated in terms of linear algebra. Researchers are trying to develop different libraries to support generic algorithms[2]. Using standard methods of linear algebra we can solve different problems that are not from any particular numerical domain. Again OO(Object Oriented) concept promotes reusing of code that provides a bridge between numerical solutions and problem formulations in real life. In this short paper Path Algebra[3][4] is used for computing transitive closure of graphs that has some dynamic flavor. In literature more applications of Max-Plus algebra can be found [5][6].

The remaining part of this paper is constructed as follows. Section II will give problem description, Section III and IV for some basics on Max-Plus Algebra and its useful properties, Section V will describe about linear equation solving by Max-Plus Algebra, Section VI and VII for transitive closure finding algorithm with corresponding description, Section VIII for an illiustratve example and Section IX for conclusion.

II. PROBLEM FORMULATION

Formal definition of transitive closure of graph is given as below.

o TRANSITIVE CLOSURE

Let a graph G = (V, E) is given where V = set of nodes or vertices of graph G and E = set of edges of graph G. Also let there are n vertices and V = {1, 2, 3, . . . ,n}. We've to answer if starting from one particular node, say 'i', is it possible to reach another node, say 'j' using edges in E where i, j ϵ V. Actually it is necessary to formulate another graph Ĝ from G such that, Ĝ has the same node as G. There is an edge (a, b) in Ĝ if there is a path from 'a' to 'b' in G where a, b ϵ V. So, the path matrix P of the graph G is the Adjacency matrix of Ĝ. In this respect Floyd-Warshall's algorithmic technique costs Θ (n^3) for finding Ĝ. In literature we may find different algorithms giving better time bound and space consumption. Here no algorithm is presented for improving speed or consuming more memory. Only another representation is considered that can be used to support unifying approach to software and hardware design[7]. Max-Plus Algebra and Matrix format representation is considered for this purpose.

III. MAX-PLUS ALGEBRA

This section will give some basic notion of Max-Plus Algebra. For all real numbers including minus infinity, define R_{max} as:

$$R_{max} = R \cup \{-\infty\}$$

We also define operator $\oplus$, for $\forall a, \forall b \in R_{max}$ such as:

$$a \oplus b = \text{maximum\_of}(a,b)$$

and operator $\odot$ for $\forall a, \forall b \in R_{max}$ such as:

$$a \odot b = (a + b)$$

We'll use in this algebra φ and ε as respectively for zero and unit element. $-\infty$ will play role of zero value(additive identity). Traditional '0' will play role of unit element(multiplicative identity). Next section is for some properties of Max Algebra, that will later be deployed for finding transitive closure of a graph.

IV. Properties Of Max Algebra

Operator $\oplus$ maintains commutative, associative and idempotent rules. Operator $\odot$ is associative and distributive over $\oplus$ [8].
Some important relations hold such as,

If a $\in R_{max}$ then,

$$\varphi \oplus a = a$$
$$\varphi \odot a = a \odot \varphi = \varphi$$
$$\varepsilon \odot a = a \odot \varepsilon = \varepsilon$$

All of these are examples of idempotent semi-ring. They are in active use in different fields [9]. Also other relations hold such as,

$$a^n = a \odot a \odot a \ldots . a \odot a \quad \text{(taking a for n times)}$$
$$a^0 = \varepsilon$$
$$a^+ = a^1 \oplus a^2 \oplus a^3 \oplus \ldots ..$$
$$a^* = \varepsilon \oplus a^+$$

V. Linear Equation Solving By Max-Plus Algebra

This section will deal with internal mechanism of how Max-Plus Algebra can be used to solve some linear equation within this algebraic formulation.

a) *Single Equation Case*

Let, a linear equation be,

$$y = a \odot y \oplus b$$

Its solution will be,

$$y = a^* \odot b$$

Short proof of this claim is given below.
In the right hand side of equation $y = a \odot y \oplus b$, y can be replaced by $y = a^* \odot b$. Now applying some simple algebraic techniques right hand side can be written as,

$$\begin{aligned} a \odot a^* \odot b \oplus b &= a^+ \odot b \oplus b \\ &= (a^+ \oplus \varepsilon) \odot b \\ &= a^* \odot b \end{aligned}$$

using some simple facts,

$$a \odot a^* = a^+$$
$$a^* = \varepsilon \oplus a^+$$

So, right hand side equals $a^* \odot b$ and initial guess of $y = a^* \odot b$ was correct.

b) *Matrix Case*

Construct Matrix $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$ and define operators $\oplus$ and $\odot$ such that,

$$A \oplus B = \{a_{i,j} \oplus b_{i,j}\}$$

$$A \odot B = \{\oplus \sum_{k=0}^{n} a_{i,k} \odot b_{k,j}\}$$

Here,

$$\oplus \sum_{k=0}^{n} a_k = a_0 \oplus a_1 \oplus a_2 \ldots a_n$$

For completeness Zero Element Matrix and Unit Element Matrix also need to be defined.
Zero Element Matrix ϕ is a matrix having all elements equal to φ. Unit Element Matrix E is a matrix having all elements equal to ε in diagonal positions and all other elements are φ. We also define for repeated operation of operators $\oplus$ and $\odot$ such that,

$$A^0 = E$$
$$A^n = A \odot A \odot A \ldots . \odot A \quad \text{(taking } A \text{ for n times)}$$
$$A^+ = A^1 \oplus A^2 \oplus A^3 \oplus \ldots .$$

Where,

$$A^* = E \oplus A^+$$

Above constructed Max-Plus Algebra of matrices is itself an idempotent semi-ring. By this construction it is possible solve $Y = A \odot Y \oplus B$ in a similar way we solved same type of equation for single equation case previously and the solution will be $Y = A^* \odot B$.
In the next section a formal description of underlying algorithm for computing transitive closure of a graph using above discussed techniques will be presented.

VI. Algorithm

Computing transitive closure of some graph involves three basic steps as below,

a) Constructing adjacency matrix form graph description and calculating all possible reachability matrices in terms of k (where k is path length) that will be needed in step b.
b) Solving corresponding linear equations of matrices.
c) Get transitive closure as output.

The following section will explain the above algorithm in details.

VII. Algorithm Explanation

a) Constructing Adjacency Matrix

Let a graph is given as Figure 1 below.

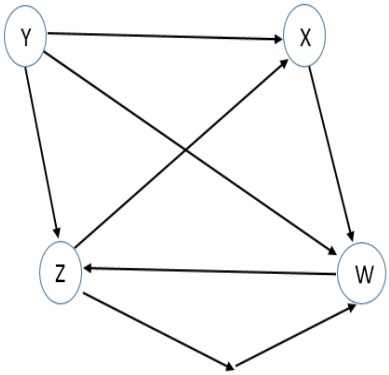

Fig 1: Graph to Find Transitive Closure

For this graph adjacency matrix A can be written as,

$$A = \begin{pmatrix} \varphi & \varphi & \varphi & 1 \\ 1 & \varphi & 1 & 1 \\ 1 & \varphi & \varphi & 1 \\ \varphi & \varphi & 1 & \varphi \end{pmatrix}$$

maintaining order of nodes as [W,X,Y,Z] while constructing column and row vectors of A.

Reachibility matrix provide only information of if it is possible starting from one node reaching another node(test on existence of a path between two nodes), not considering path length or how many edges could be associated with that path atmost. Another destination node could be the starting node itself. Computing A^k will provide information on the existence of path between two nodes having length at most k and also provide corresponding path length for a particular choice of k. Graph theory says matrix multiplication of adjacency matrix to itself will provide this kind of information. Graph theory states to multiply k-1 times adjacency matrix to itself to generate as result matrix A^k.

For example reachability matrix can be written in terms of k (where k is path length) for graph in Figure 1 as below,

$$A^2 = \begin{pmatrix} \varphi & \varphi & 1 & \varphi \\ 1 & \varphi & 1 & 2 \\ \varphi & \varphi & 1 & 1 \\ 1 & \varphi & \varphi & 1 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 1 & \varphi & \varphi & 1 \\ 1 & \varphi & 2 & 2 \\ 1 & \varphi & 1 & 1 \\ \varphi & \varphi & 1 & 1 \end{pmatrix}$$

$$A^4 = \begin{pmatrix} \varphi & \varphi & 1 & 1 \\ 2 & \varphi & 2 & 3 \\ 1 & \varphi & 1 & 2 \\ 1 & \varphi & 1 & 1 \end{pmatrix}$$

In general A^k will give a matrix where every matrix entry is computed as below,

$$a_{i,j}^k = \begin{cases} \varphi \ \ if\ not\ (i \neg j) \\ compute\ normal\ matrix\ operation \end{cases}$$

Where $i \neg j$ denotes, from i node it is possible to reach j node using at most k steps or k edges where k>1. For reachability matrix calculation compute upto A^k where A^kcontains atleast 1 non-φ element.

b) Solving Linear Equations of Matrices

From Max-Plus algebra described above,

$$A^* = E + A^+$$

where,

$$A^0 = E$$
$$A^+ = A^1 \oplus A^2 \oplus A^3 \oplus \ldots.$$

Here A^* will give reachability matrix and every element will give corresponding maximum length of $i \neg j$. From section V solution of $Y = A \odot Y \oplus B$ is known. Now letting $B = E$,

$$Y = A \odot Y \oplus B$$
$$= A \odot Y \oplus E \ \ (as\ B = E)$$

Its solution is,

$$Y = A^*$$

c) Transitive Closure as Output

Let,

$$T = zero\ matrix\ same\ size\ of\ Y$$

For example,

$$T = \begin{bmatrix} \varphi & \varphi \\ \varphi & \varphi \end{bmatrix} as\ a\ sample\ size\ of\ 2\ x\ 2$$

This matrix has all elements equal to φ.
Then transitive closure T can be calculated as,

$$T = T \oplus (E \odot Y)$$

For computating diagonal elements of T from above equation compute,

$$t_{ij} = t_{ij} \oplus \left(\varepsilon \odot y_{ij}\right) \ \ here\ (i,j)\ \varepsilon\ V$$

For computating non-diagonal elements of T from above equation compute,

$$t_{ij} = t_{ij} \oplus \left(\varphi \odot y_{ij}\right) \ \ here\ (i,j)\ \varepsilon\ V$$

Using the fact that,

$$\varepsilon \odot \varphi = \varepsilon$$

Here t_{ij} will give transitive closure calculation for node ‘i’ to node ‘j’ based on following condition.

$$t_{ij} = \begin{cases} \varepsilon \; if \; (i \neg j) \\ \varphi \; otherwise \end{cases}$$

This signifies that if unit element is found then it is possible to reach from node i to node j anyhow. Otherwise it is un-reachable.

An example will be given in the following section for visual inspection of the above algorithm.

VIII. Illustrative Example

Previously a graph is described in Figure 1 and for that graph adjacency matrix and corresponding reachability matrices were computed. Take that graph for current example. Thus complete step a). Now for step b) compute $Y = A^*$ as all reachability matrices were precomputed in step a) before. After computation,

$$Y = \begin{pmatrix} 1\varphi 23 \\ 5\varphi 68 \\ 3\varphi 35 \\ 2\varphi 33 \end{pmatrix}$$

Now for step c) computing transitive closure T using rules of t_{ij} given before and result is computed as below,

$$T = \begin{pmatrix} \varepsilon\varphi\varepsilon\varepsilon \\ \varepsilon\varphi\varepsilon\varepsilon \\ \varepsilon\varphi\varepsilon\varepsilon \\ \varepsilon\varphi\varepsilon\varepsilon \end{pmatrix}$$

IX. Conclusion

This short paper is influenced by similar Path algebra algorithm for finding longest increasing subsequence[8].This work simply illustrates another example of the power of Max-Plus Algebra solving computing problems and strengthens the possibility of unifying approach to software and hardware design for scientific calculations.

Conflict Of Interest

The author declares that he has no conflict of interest.

References

[1] David R. Musser and Alexander A. Stepanov. Generic Programming.ISSAC1988,pages:13-25.

[2] Boost(C++libraries).http://www.boost.org

[3] Bernard Carrė. Graphs and Networks, Oxford University Press,1979.

[4] B. De Schutter and T. van den Boom, “Max-plus algebra and max-plus linear discrete event systems: An introduction,” Proceedings of the 9th International Workshop on Discrete Event Systems (WODES’08), G¨oteborg, Sweden, pp.36–42, May 2008.

[5] B. Heidergot, G.J.Olsder, J.Woude, Max Plus at Work, Princeton Universty Press,New Jersey, 2006.

[6] Guy Cohen,Stephane Gaubert,Jean-Pierre Quadrat."MAX-PLUS ALGEBRA AND SYSTEM THEORY: WHERE WE ARE AND WHERE TO GO NOW",Annual Reviews in Control (IFAC, Elsevier), 1999.

[7] G.L. Litvinov, V.P. Maslov, A.Ya.Rodionov. A unifying approach to software and hardware design for scientific calculations and idempotent mathematics.arxiv.org/abs/math/0101069.

[8] Anatoly Rodionov, Path algebra algorithm for finding longest increasing subsequence , arXiv: 1409.2928v1,Computer Science, Data Srtucture, September 11, 2014.

[9] Bernard Heidergott, Geert Jan Olsder, Jacob van der Woude. Max Plus at Work . Princeton University Press, 2006.