A GraphBLAS Implementation of Triangle Centrality

Fuhuan Li, David A. Bader
Department of Data Science
New Jersey Institute of Technology
Newark, New Jersey, USA
{fl28,bader}@njit.edu

2021 IEEE High Performance Extreme Computing Conference (HPEC) | 978-1-6654-2369-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/HPEC49654.2021.9622806

***Abstract*—Identifying key members in large social network graphs is an important graph analytic. Recently, a new centrality measure called triangle centrality finds members based on the triangle support of vertices in graph. In this paper, we describe our rapid implementation of triangle centrality using GraphBLAS, an API specification for describing graph algorithms in the language of linear algebra. We use triangle centrality's algebraic algorithm and easily implement it using the SuiteSparse GraphBLAS library. A set of experiments on large, sparse graph datasets is conducted to verify the implementation.**

***Index Terms*—Graph algorithms, Sparse matrix computations, High Performance Data Analytics**

I. Introduction

Graphs are ubiquitous data structures in numerous domains, such as cybersecurity, finance, and social media. Given a specific graph, finding the most important vertices has many applications in real-world entities like web searching and is a crucial part in graph analysis. The fundamental mathematical concept of the centrality is not well defined, hence the notion of centrality has led to many different graph centrality measures such as degree centrality, closeness centrality, and betweenness centrality.

A triangle in graph is a subset of three vertices where all three edges exist between the vertices. Triangles represent cohesiveness, and higher triangle counts in a graph means increased interconnections among vertices. A vertex is important if the vertex and its neighbors are regarded as a more cohesive group where information could spread from any individual to another much faster. In other words, a vertex can be important because it, as well as its neighbors, are in many triangles. Recently, Burkhardt [1] introduced *triangle centrality* as a new centrality measure that captures the influence of triangles on the importance of vertices.

In this paper, we describe triangle centrality of a graph and its algebraic form. Based on the algebraic algorithm, we describe our new GraphBLAS implementation and give performance results using an implementation of GraphBLAS from Tim Davis's SuiteSparse.

II. Triangle Centrality

A. Definition

The notion of triangle centrality is based on the triangle counts from a vertex and each of its neighbors. Hence a vertex is important if it is in many triangles or if its neighbors are in many triangles. The precise definition of the triangle centrality from [1] is as follows.

$$TC(v) = \frac{\frac{1}{3}\sum_{u\in N^+_\triangle(v)} \triangle(u) + \sum_{w\in\{N(v)\setminus N_\triangle(v)\}} \triangle(w)}{\triangle(G)}$$

for a simple undirected graph denoted by $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. The neighborhood of a vertex v is $N(v) = \{u|(u, v) \in E\}$. The subset of neighbors in triangles with v will be denoted by $N_\triangle(v) = \{u \in N(v)|N(u) \cap N(v) \neq \emptyset\}$ and if v is included then the closed triangle neighborhood of v is $N^+_\triangle(v) = \{v\} \cup N_\triangle(v)$. We denote local triangle count for a vertex v by $\triangle(v)$ and the total triangle count of G by $\triangle(G)$.

By definition, the centrality values indicate the proportion of triangles centered at a vertex which is bounded in the range $[0, 1]$.

B. Algebraic Derivation

The triangle centrality can also be formulated in linear algebra (see [1]). From theorem 1 in [2], given G and Hadamard product $T = (A^2 \circ A)$, then $\triangle(v) = \frac{1}{2}\|T_v\|_1 = {T_v}^T 1$ and $\triangle(G) = \frac{1}{6}1^T T1$. Here, T represents the triangle neighbors of each vertex. T_v is a column vector of T, and the indices of T_v corresponding to nonzeros represent the triangle neighbors of v. Hence $N_\triangle(v) = supp(T_v)$. Then we can make substitution using T

$$TC(v) = \frac{\sum_{u\in\{v\}\cup supp(T_v)}\|T_u\|_1 + 3\sum_{w\in supp(A_v - T_v)}\|T_w\|_1}{1^T T1}$$

since the sum $\sum_{u\in\{v\}\cup supp(T_v)}\|T_u\|_1$ can be achieved by $(T\breve{T}_v + T_v)^T 1$, where $\breve{T}$ denotes the binary form of the matrix or vector. Similarly, the sum of triangle counts from non-triangle neighbors can be obtained by $(T(A_v - \breve{T}_v))^T 1$. Then the triangle centrality formulation can be transformed into matrix vector product.

$$TC(v) = \frac{(3A_v - 2\breve{T}_v + I_v)^T T1}{1^T T1}$$

This leads to the linear algebraic formulation for triangle centrality in matrix notion as follows

$$C = \frac{(3A - 2\breve{T} + I)T1}{1^T T1}$$

978-1-6654-2369-4/21/$31.00 ©2021 IEEE

Based on the linear algebra form, we can use algebraic algorithm of triangle centrality.

Algorithm 1: Algebraic triangle centrality

Input: A, adjacency matrix in sparse matrix representation
Input: $T = A^2 \circ A$, graph triangle matrix in sparse matrix representation
Create binary matrix $\check{T}$
set $X := 3A - 2\check{T} + I$
set $y := T1$
set $k := 1^T y$
Output: $C := \frac{1}{k} X y$

III. Implementation

GraphBLAS [3] is an API specification that defines standard building blocks for graph algorithms in the language of linear algebra. It is built upon the notion that a sparse matrix can be used to represent graphs as either an adjacency matrix or an incidence matrix. It provides a powerful framework for creating graph algorithms based on the elegant mathematics of sparse matrix operations on a semiring [4]. In this work, we use the SuiteSparse:GraphBLAS [5] version 5.1.5. We are able to rapidly implement triangle centrality in GraphBLAS due to its concise algebraic form.

GraphBLAS can compute the $T = A^2 \circ A$ with a masked matrix multiply, $C\langle A\rangle = A^2$. This is a much faster method, and saves significant memory since it does not calculate all of A^2, only these entries in the pattern of A. The corresponding code is quite simple. Here is our quite elegant GraphBLAS implementation of triangle centrality based on the algebraic algorithm.

```
GrB_Info triangleCentrality (GrB_Vector *result,
    GrB_Matrix A) {
    GrB_Index n;
    GrB_Matrix_nrows(&n, A);
    GrB_Matrix T;
    GrB_Matrix_new(&T, GrB_FP64, n, n);
    GrB_Vector T_y;
    GrB_Vector_new(&T_y,GrB_FP64,n);
    GrB_Vector y;
    GrB_Vector_new(&y, GrB_FP64, n);
    GrB_Vector_new(result, GrB_FP64, n);
    double k;
    //Compute T, y, k
    GrB_mxm(T, A, NULL, GxB_PLUS_TIMES_FP64, A,
        A, NULL);
    GrB_reduce(y, NULL, NULL, GrB_PLUS_FP64, T,
        NULL);
    GrB_reduce(&k, NULL, (GrB_Monoid)
        GrB_PLUS_MONOID_FP64, y, NULL);
    // Distributed computing for X
    GrB_mxv(*result,NULL,GrB_PLUS_FP64,
        GxB_PLUS_TIMES_FP64,A,y,NULL);
    GrB_apply(*result,NULL,NULL,GrB_TIMES_FP64,3,*
        result,NULL);
    GrB_mxv(T_y,NULL,NULL,
        GxB_PLUS_SECOND_FP64,T,y,NULL);
    GrB_apply(T_y,NULL,NULL,GrB_TIMES_FP64,2,T_y,
        NULL);
    //Add together to get the final result
    GrB_eWiseAdd(*result,NULL,GrB_PLUS_FP64,
        GrB_MINUS_FP64,y,T_y,NULL);
    GrB_apply(*result, NULL, NULL, GrB_TIMES_FP64,
        1 / k, *result, NULL);
    return GrB_SUCCESS;
}
```

To demonstrate the performance of this implementation, we conducted a set of experiments with a four-core Intel CPU, and 16GB of memory. Results are shown in Table I.

TABLE I
Performance of Triangle Centrality Implementation via GraphBLAS (second).

Name	n(Vertices)	m(Edges)	△(G) (triangles)	Time (sec)
com-Youtube	1,134,890	2,987,624	3,056,386	4.08
as-Skitter	1,696,415	11,095,298	28,769,868	16.6
com-LiveJournal	3,997,962	34,681,189	177,820,130	69.4
com-Orkut	3,072,441	117,185,083	627,584,181	686.2

Related work: After Bader shared Burkhardt's triangle centrality paper with the GraphBLAS committee, Pelletier [6] implemented a version in pygraphblas, a high-level Python wrapper for the GraphBLAS C API. Based on the capabilities of pygraphblas, the author modified the algorithm to improve the performance which includes computing $3Ay - 2\check{T}y + y$ rather than $(3A - 2\check{T} + I)y$.

IV. Conclusion

In this paper, we describe our rapid implementation of triangle centrality using GraphBLAS, and demonstrate its performance on several large, sparse graphs that represent real-world scenarios. Triangle centrality, as a new centrality measure, can be a helpful and a complementary tool for graph analysis. Moreover, due to its elegant linear algebraic formulation, it is easily implemented in the GraphBLAS framework.

V. Acknowledgement

This research was funded in part by NSF grant number CCF-2109988.

References

[1] P. Burkhardt, "Triangle centrality," *arXiv preprint arXiv:2105.00110*, 2021.
[2] ——, "Graphing trillions of triangles," *Information Visualization*, vol. 16, no. 3, pp. 157–166, 2017.
[3] J. Kepner, H. Meyerhenke, S. McMillan, C. Yang, J. D. Owens, M. Zalewski, T. Mattson, J. Moreira, P. Aaltonen, D. Bader, and et al., "Mathematical foundations of the GraphBLAS," *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, Sep 2016. [Online]. Available: http://dx.doi.org/10.1109/HPEC.2016.7761646
[4] T.A.Davis, "SuiteSparse:GraphBLAS," 2021. [Online]. Available: http://faculty.cse.tamu.edu/davis/GraphBLAS.html
[5] T. A. Davis, "Algorithm 1000: SuiteSparse:GraphBLAS: Graph algorithms in the language of sparse linear algebra," *ACM Trans. Math. Softw.*, vol. 45, no. 4, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3322125
[6] M. Pelletier, "Triangle centrality pygraphbas code," 2021. [Online]. Available: https://github.com/Graphegon/pygraphblas

978-1-6654-2369-4/21/$31.00 ©2021 IEEE