COMPARISON OF BETWEENNESS AND CLOSENESS CENTRALITIES USING INCREMENTAL ALGORITHMS IN DYNAMICALLY GROWING NETWORKS

Priyanka, Anand.R[1], Dr.Rajashekhar M.Patil[2]
BMSIT College, Bangalore
Email: priyankanadgoud2@gmail.com[1],anandor@bmsit.in[2]

Abstract

Centrality is one of the important concepts in dynamic social networks. One key aspect of social network analysis is to understand the central nodes in the network. Centrality is defined to identify the most central node in the network. In a graph representing a social network however dynamic calculation of centrality values for rapidly growing networks might be unfeasibly expensive, especially if it involves recalculation from scratch for each time period. In this paper, we provide an fast incremental algorithms comparison for closeness centrality and betweenness centrality computation. The algorithms proposed here supports efficiently computation of all pairs shortest paths and centrality values based upon changes in network topology, that is edge insertion and deletion. Our performance results provides substantial performance speedup on the order of thousands of times, including the best performing non-incremental centrality algorithms, and newly proposed centrality update algorithms.

Keywords-Closeness centrality, betweenness centrality, incremental algorithms, dynamic networks, all-pairs shortest paths, update algorithm.

I. INTRODUCTION

In social network analysis, indicators of centrality identify the most important nodes within a network. For decades, it has been an important and active research field for solving a number of problems like revealing patterns of information dissemination, identifying the individuals who are best placed to influence or influential actors, assessing the impact of business decisions in organisational structures, and even for finding the best store locations in cities etc. We proposed incremental algorithms for evaluating quickly the effects of topology modification on centrality values.

The closeness centrality of a node x is defined as length of their shortest paths. The far ness of node x is defined as the sum of its distances from all other nodes, and its closeness was defined by Bavelas as reciprocal of the farness .Closeness centrality algorithm that handles various types of network updates including addition, removal and modification of nodes and edges. The betweenness centrality of node x is defined by Freeman as the fraction of the shortest paths that pass through x across all pairs of nodes in a network.

The traditional techniques used for computing both the centralities of all vertices can be calculated by solving all pairs shortest-path problems, which can be solved by various algorithms taking $O(V^2 \log V + VE)$ time, where V is the number of vertices and E is the number of edges of a graph.

To provide solutions to costly problems on continually changing networks, incremental algorithms have been most commonly used. An incremental algorithm is an algorithm that updates the solutions to a problem after an incremental change is made on its input.

The initial design point for all of these centrality metrics, including closeness and betweenness was static snapshots of small networks and the limiting algorithmic complexities and computation times of centrality

measures was not a significant problem for such small, static networks and to quantify the effects of topology modification and to find exact centrality scores existing algorithms are not efficient enough to be used in practise. So, incremental algorithms are designed to evaluate the effects of topology modification. The main contribution of this paper are incremental algorithms which efficiently update the closeness and betweenness centralities upon edge insertion and deletion. Incremental algorithms are used to reduce the cost of computation and they arrive at solution for computationally complex problems in an efficient manner. The main desire to select closeness and betweeness centrality is, these are one of the most commonly used metrics in social network analysis and both the centralities depends entirely on the shortest path information across all pairs of nodes .Therefore this paper compares the betweenness and closeness centralities as the example incremental metrics. Nowadays, many real life networks that can obtained online evolve only by growing network updates. Handling growing network updates is important and can be handled by a single incremental algorithm.

II. COMPUTATION OF CLOSENESS AND BETWEENESS CENTRALITY

A.NOTATION

A directed network G consists of a set of nodes V(G) and edges E(G) where n is the number of nodes, and m is the number of edges in the network. $x \rightarrow y \in E(G)$ represents an edge directed from node x to node y, where $x \in V(G)$ is a predecessor of y, and $y \in V(G)$ is a successor of x. Pred(x)is used to denote all predecessors of x in the network. Pred(y) denotes the set of predecessors of node y on the shortest paths from node x. G is the transpose of network G where all edges in network G are reversed in direction. The set of edges, nodes, and edge costs are also defined for network G. In weighted networks, each edge e in the network has a traversal cost of C(e) where $C(x \rightarrow y) > 0$ for $x \rightarrow y \in E(G)$. The length of a path is the sum of the costs of the edges on Path. The distance from node x to y is the length of the minimum-length path from x to y that is also called the shortest path. D(x, y) denotes the shortest distance while σ(x, y) denotes the number of shortest paths from node x to y.

Closeness centrality is traditionally best computed by running a single-source shortest path algorithm using each node as the source node once. At each iteration, the distances found are summed up to obtain the total distance from the given source node, and this distance is inverted to obtain the closeness value of the source and betweenness centrality denotes fraction of shortest paths that pass through node across all pairs of nodes. In un weighted networks, a breadth-first search algorithm may be used to discover the shortest paths from a source nodes, which is bounded by $O(n+m)$ time complexity per source node, resulting in $O(nm)$ complexity in total. In weighted networks, Dijisktra's algorithm has $O((n+m)\log n)$ complexity, where n denotes the number of nodes, m denotes the number of edges in the network. This complexity is achieved when a binary min-heap is used in the implementation of the priority queue. A faster run-time of $O(m + n\log n)$ can be achieved by implementing the priority queue using a Fibonacci heap. When Dijisktra's algorithm is invoked using every node in the network as the source node to compute all-pairs shortest paths, the overall complexity is $O(mn + n2\log n)$. Computation of both the centralities can be performed by running an all-pair shortest paths algorithm (e.g.Floyd-Warshall), which results in $O(n3)$ time complexity.The algorithmic complexities of Dijkstra and Floyd-Warshall are sufficiently high that they are very difficult to invoke at every time step in dynamically changing, large-scale networks. Hence, this paper proposes the use of incremental algorithms that avoid the cost of re computing all of the shortest paths from scratch every time period. The algorithms presented here are designed to handle weighted, directional, dynamic networks with no negative edge costs or weights.

B.OVERVIEW OF CLOSENESS AND BETWEENESS CENTRALITY

The closeness centrality of node x, H(x),is defined as the inverse of the sum of the distances from node x and all other nodes in the network:

$$H(x) = \sum_{y \neq x} \frac{1}{d(y,x)}.$$

Where d(y, x) denotes the shortest distance from node *x* to node *y*.

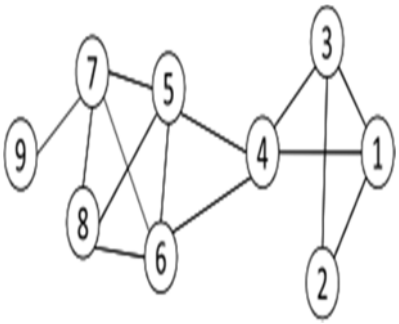
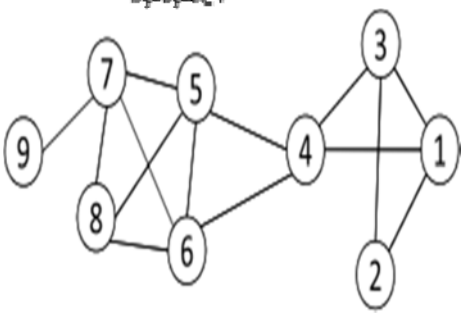


Fig 1: Example Network for Calculating closeness Centrality. For the above fig.1 the most central node is 4 based on values obtained by using the above formulae.

Betweenness centrality of a node s is defined as the fraction of shortest paths that pass through node t across all pairs of nodes. Let $\sigma_{st}(v)$ be the number of shortest paths from s to t and σ_{st} be the number of shortest paths from s to t that contain node s.

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$



For the fig.2 the betweenness centrality foe node 4 is 15, it can be calculated by using the above formulae.

C. ALGORITHM VARIANTS FOR COMPUTING BOTH THE CENTRALITIES.

Many researchers have provided algorithms for variants of closeness and betweenness centralities. Centrality is a commonly used social centrality metric and it can have different uses in different contexts. Closeness centrality of a social actor describes actor's efficiency for information propagation across the entire network. In other words, social actors with high centrality values are considered to be efficient at making contact with others in the network. High centrality is also regarded as representing high potential for independent communication.

In the context of technological networks, such as wireless networks, closeness centrality identifies nodes that have rapid access to information (e.g. nodes that are close to many other nodes on average). Since closeness centrality is inversely proportional to the sum of the distances to all other nodes, it also provides an estimate of how long it will take information to spread from a node to all others. Hence, it can also be used as a performance measure in technological networks. As another example, the authors discuss the use of closeness centrality for policy-making networks (e.g. drug policy making). In the context of policy-making networks, the actors that have information that is crucial to all other actors in the network should have high closeness centrality if the network is to function effectively. As we mentioned earlier, closeness centrality is one of the most commonly used metrics in social network analysis. Variants for betweenness centrality focus on incorporating over-time information and we focus here more on faster computation of the original betweenness metric in dynamically growing networks.

D.WORK ON DYNAMIC SHORTEST PATH COMPUTATIONS

We also draw on earlier research on dynamic shortest path computations. Computation of both the centralities are tightly coupled with solving the all-pairs shortest paths problem. In the literature, there are many different techniques proposed for solving the all-pairs shortest paths problem dynamically. However, some of these techniques come with a number of restrictions. For instance, solves the all-pairs shortest paths problem in networks that have positive integer edge costs that are less than a certain number, b, which is inapplicable for networks whose edges are positive real valued numbers. The Demetrescu and Italiano algorithm depends on the notions of locally shortest paths and locally historical paths. The main idea is to maintain dynamically the set of locally historical paths, which is a path that has been identified as a shortest path at some point and has not been modified since then. In this study, we use the dynamic all-pairs shortest path algorithm proposed by Ramalingam and Reps in as our building block to maintain all-pairs shortest paths dynamically[. There are many reasons why we use Ramalingam and Reps algorithm as our

basic building block. First, Ramalingam and Reps algorithm is the most commonly used dynamic all-pairs shortest paths algorithm in the literature. Second, it has good performance compared to other dynamic all-pair shortest path algorithms available in the literature considering the experiments presented. Ramalingam and Reps algorithm performs quite well on sparse, real-life networks and the compute times of Ramalingam and Reps algorithm and Demetrescu and Italiano's algorithm are very close. In experiments done with real life networks, Ramalingam and Reps have the lowest or one of the lowest compute times among all dynamic all-pairs shortest paths algorithms. Third, Ramalingam and Reps algorithm is shown to scale better as the number of nodes increases because Demetrescu & Italiano's algorithm maintains more global structures and requires more memory while Ramalingam and Reps algorithm requires less space and exhibits better locality in its memory access pattern. Since supporting an increasing number of nodes is important for dynamically growing social networks, we decided to use Ramalingam and Reps algorithm as a building block in our algorithm.

III. INCREMENTAL CLOSENESS AND BETWEENESS ALGORITHMS.

An incremental algorithm updates the solution to a problem after an incremental change is made on its input. In the application of an incremental algorithm, the initial run is conducted by an algorithm that performs the desired computation from scratch and the incremental algorithm is used in the subsequent runs (i) using information from earlier computations and (ii) to reflect the update on the network while avoiding re-computations as much as possible.To compute the closeness values incrementally for streaming, dynamically changing social networks, the incremental all-pairs shortest-paths algorithm proposed by Ramalingam and Reps is extended such that closeness values are incrementally updated in line with the changing shortest path distances in the network.

The proposed algorithms depend on the dynamic all-pairs shortest path algorithms proposed to incorporate the computation of both the centralities. Incremental algorithms usually provide faster solutions at the cost of more memory usage. The incremental centrality algorithms also takes quadratic space, using memory on the order of O(n2 + m).

IV. DATASETS AND RESULTS

The goal of this paper is to draw attention to the use of incremental algorithm design in social network analysis methods. In particular, in this paper, we explore how much speedup we obtain due to the use of an incremental algorithm on different types of synthetic and real life networks that are used by social network researchers. Hence, our performance evaluations primarily show how much performance improvement can be achieved over the most commonly used way of computing the all-pairs shortest paths in a network as well as closeness and betweenness centralities as a by-product of it. Here we mainly comparing the two centrality measures based upon various parameters and results obtained.

A.SYNTHETIC NETWORKS

To know the various performance results of the proposed incremental algorithms, we have designed experiments with networks that are generated using different graph generation algorithms and network sizes. To understand the impact of topology, we use synthetic networks using three different topologies, while keeping the number of nodes and the average degree fixed. In our experiments, we use three different topologies: preferential attachment networks(PF), Erdos-Renyi networks(ER) , and small-world networks(SW) . We use networks with 1000, 3000,and 5000 nodes with a step size of 2000 and set the average degree to 6. The average degree of a network is a measure that compares the number of edges against the number of nodes in the network. It is computed as $2.|E(G)| / |N(G)|$ as each edge contributes to the degree of both nodes it is connecting. The rewiring probability for small world networks is set to 0.5.To measure the performance of the algorithms incremental closeness and betweenness algorithm for growing network updates, we build the synthetic networks described above with 100 edges that are selected randomly. We insert incrementally the last 100 edges and get the average update performance in terms of execution time. For instance, if it takes 5 seconds to complete a set of updates using incremental algorithms and 30seconds to

complete the same set using various proposed algorithms, then we conclude that incremental algorithm as 6x times faster .For closeness centrality we are using dijisktra's algorithm and for betweeness algorithm we are using Brande's algorithm as proposed algorithm. For synthetic networks, we use preferential attachment networks (PF), Erdos-Renyi networks(ER), and small-world (SW) networks. We vary the number of nodes from 1000 to 5000 with a step size of 2000, and fix the average degree to 6. For small world networks, the rewiring probability is 0.5. We generate these synthetic networks with all but 100 edges that are selected randomly. We insert the last 100 edges incrementally and get the average update performance in terms of execution time over the repeated invocations of Brandes' and dijisktra's algorithm, which are the best performing algorithm used in standard implementations. The performance benefits of the incremental betweenness and closeness algorithm increase with the increasing network size.

As shown in table 3, diameter and average path length of the network are inversely related with the performance obtained. For instance, in networks generated using a preferential attachment generation model, characteristic path length and diameter are lower compared to other topologies. When the paths are short in a network, an update on the shortest paths cannot propagate very far, resulting in quick return from the update and a very limited number of affected nodes (e.g. less than 5% in the case of preferential attachment networks).The results obtained for the centralities are almost nearer so both centrality measures, performance results obtained are nearly equal in value for synthetic networks as shown in below tables.

TABLE 1 - PERFORMANCE IMPROVEMENTS OBTAINED ON DIFFERENT NETWORKS WITH DIFFERENT TOPOLOGIES/SIZES.

#(Nodes)	PF	ER	SW
1000	1178.66 x	7.99 x	17.48 x
3000	971.40 x	18.98 x	18.53 x
5000	3760.48 x	31.19 x	22.54 x

TABLE 2 - PERCENTAGE OF AFFECTED NODES (AFFECTEDSINKS + AFFECTEDSOURCES)

#(Nodes)	PF	ER	SW
1000	3.54%	79.16%	32.52%
3000	1.98%	85.7%	33.35%
5000	1.16%	87.36%	31.86%

TABLE 3-NETWORK STATISTICS.

Topology	Size	Min Deg	Max Deg	Degree StdDev	Diameter	Avg Path Len	Clustering Coeff
Pref. Attach.	1000	3	89	6.822	10	3.45	0.014
Pref. Attach.	3000	3	233	8.064	14	4.126	0.007
Pref. Attach.	5000	3	212	8.251	16	4.442	0.005
Erdos-Renyi	1000	1	14	2.498	15	6.305	0.003
Erdos-Renyi	3000	2	13	1.572	14	7.086	0.001
Erdos-Renyi	5000	2	11	1.362	14	7.492	0.001
Small World	1000	3	12	1.545	33	7.612	0.044
Small World	3000	3	12	1.526	55	10.333	0.039
Small World	5000	3	4	1.502	71	11.934	0.039

B.REAL LIFE NETWORKS

Here, we evaluate the performance of the algorithms that we are used , using a number of real life networks that are of different magnitudes and consists of various parameters and that grow drastically in incremental order over time. The weighted networks are considered for evaluation, where the cost of an edge is inversely proportional to the strength of relationship. We consolidate multiple updates for the same pair of nodes in a single edge. For instance, if an interaction between two nodes x and y has been recorded twice up to a certain point, then the edge $x \rightarrow y$ has the cost of 1/2. When a third update is recorded between x and y, then the cost of the edge $x \rightarrow y$ is updated to be 1/3. We first describe the datasets we have used, and then compare the performance of our incremental betweenness and closeness update algorithm against the best performing non-incremental betweenness and closeness algorithms (Brandes'algorithm and Dijisktra's algorithm). We use four different real life networks: Socio Patterns , Facebook-like (online-forum communication between students) , HEP Co-Authorship Network (co-authorship relations between High-Energy Physics researchers), and P2P Communication Network (P2P file sharing) .

Next, we report our performance results and explain them in line with the topological features of the networks described above. For the real life network, in order to perform with the growing network updates, we start with an earlier version of the network that has all the information expect the last 100 updates. Table 4 presents the performance improvements obtained along with basic information on the networks. The avg. speedup column in Table 4 describes the speedup obtained over Brandes' and Dijisktra's algorithms averaged across 100 updates on the network. For instance, for a single update the incremental value for both the algorithms is 9.58 times faster on average than invoking both the algorithms for the same update. The performance benefits improve with the increasing network size and decreasing characteristic path length, diameter, and average betweenness. For instance, on the HEP co-authorship network, there are several close-knit groups and it is a relatively more connected network than the P2P communication network, where only a few users act as servers for the other users providing them with files to download. Hence, in the P2P communication network, very few nodes can lie on the shortest paths between other nodes. Consequently, when a network update occurs, few shortest paths tend to be changed,and thus few centrality values are affected, resulting in a dramatic average speedup per each update over both Brandes's and Dijisktra's algorithms.

TABLE 4- TOPOLOGICAL FEATURES OF REAL LIFE NETWORKS, CORRESPONDING PERFORMANCE RESULTS.

	N	E	Avg Deg	Max Deg	Std. Dev. Deg	Diameter	Avg Path Len.	Clus Coef
SocioPatterns	113	4392	38.8	98	18.3	3	1.656	0.53
OnlineForum	1897	20290	21.4	339	35.6	8	3.196	0.08
HEPCoauthor	7507	38804	5.16	64	6.14	15	5.742	0.45
P2P Comm.	6843	7572	2.21	2185	38.3	3	1.248	0

IV. CONCLUSION

This paper proposes an comparison of incremental algorithms for both betweenness and closeness centrality measures in dynamic social networks. The main goal is to avoid re-computations involved and reflect the changes triggered by network update as efficiently as possible .The performance results obtained here indicate substantial performance improvements over the state of including non-incremental and dynamic update algorithms on realistic network data. Hence incremental algorithms offers larger potential to allow dynamic social networks to be applied to real time data and too much larger datasets that would have been possible using traditional centrality metric computation. Therefore in conclusion the incremental algorithms are applied and compared with the other centrality measures for updating the dynamically growing networks and to obtain the desired results.

REFERENCES

[1]M.Kas,K.M Carley,and Matthew Waches,L.R Carley "Incremental Algorithm for Updating the Betweenness Centrality in Dynamically Growing Networks"2015.

[2]M.Kas,K.M Carley,and Matthew Waches,L.R Carley "Incremental Algorithm for Updating the Closeness Centrality in Dynamically Growing Networks"2015.

[3]M.Kas, K.M Carley, and L.R Carley,"Incremental Closeness Centrality for Dynamically Changing Social Networks," in Workshop on the Semantic and Dynamic Analysis of Information Networks(ASONAM).,2013.

[4]G. Ramalingam and T. Reps, "On the Computational Complexity of Incremental Algorithms," CS, Univ. of Wisconsin at Madison, Tech.Report 1991.

[5] U. Brandes, "A Faster Algorithm for Betweenness Centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163--177, 2001.

[6] M. Kas, "Incremental Centrality Algorithms for Dynamic Network Analysis," ECE, Carnegie Mellon University, Pittsburgh, PA, Ph.D.Dissertation 2013.