Betweenness Ordering Problem: An Efficient Non-Uniform Sampling Technique for Large Graphs

Manas Agarwal 2, Rishi Ranjan Singh 1, Shubham Chaudhary 2, Sudarshan Iyengar 1

1 Department of Computer Science and Engineering, Indian Institute of Technology Ropar
Nangal Road, Rupnagar, Punjab, India
2 Department of Mathematics, Indian Institute of Technology Roorkee
Roorkee, Uttarakhand, India

Abstract—Centrality measures, erstwhile popular amongst the sociologists and psychologists, has seen wide and increasing applications across several disciplines of late. In conjunction with the big data problems there came the need to analyze big networks and in this connection, centrality measures became of great interest to the community of mathematicians, computer scientists and physicists. While it is an important question to ask how one can rank vertices based on their importance in a network, there hasn’t been a commonly accepted definition, mainly due to the subjectivity of the term “importance”. Amongst a plethora of application specific definitions available in the literature to rank the vertices, closeness centrality, betweenness centrality and eigenvector centrality (page-rank) have been the most important and widely applied ones. In the current paper, we formulate a method to determine the betweenness ordering of two vertices without exactly computing their betweenness indices - which is a daunting task for networks of large size. We apply our approach to find the betweenness ordering of two vertices in several synthetic and real world graphs. We compare our method with the available techniques in the literature and show that our method fares several times better than the currently known techniques. We further show that the accuracy of our algorithm gets better with the increase in size and density of the network.

I. INTRODUCTION

Centrality of a vertex in a network is the quantification of the intuitive notion of importance of a node in a network. Centrality measures have been extensively used in the analysis of large data available from real world networks in the recent times. Centrality indices, also referred as structural indices, are real valued functions that remain invariant under isomorphic transformation of graphs [1]. Different centrality measures are coined in the literature for application specific reasons. For a detailed study of centrality indices and their applications, one can refer to the books by Newman [2], Jackson [3] and Brandes and Erlebach [1]. Real-world networks are generally very large in size, dynamic in nature and keep changing at a very high rate. In such networks, comparing centrality scores of two nodes is of great importance, consider for example, a production company is finalizing a new brand ambassador for their organization and has two options to choose from. To evaluate which one is better, one might need to compare the importance (in this case popularity) of the two candidate actors in a given social network where the number of nodes are in the order of thousands. Consider two research-papers in the citation network [4] which is of the order of a few million nodes, how can one find which paper is more central than the other one? For example, if one were to compare betweenness centrality in this case, even with the adoption of the best known algorithm, it is a time consuming task for large sized networks. We ask this question Is there a method to compute the centrality ordering of two nodes and declare which one is more central than the other, without actually computing their exact centrality values. More formally, given two nodes $u$ and $v$ in a graph $G$ with centrality values $C(u)$ and $C(v)$ with $C(u) > C(v)$ thus making $u$ superior in rank than $v$, can one get to know which node is of superior rank over the other without computing its centrality values. We call this problem the centrality-ordering problem.

We illustrate our problem in the context of a centrality measure called the eccentricity measure and give a simple approximation approach for eccentricity ordering.

![Image](https://example.com/fig1.png)

Fig. 1. Eccentricity ordering in 2-D Euclidean plane

Eccentricity of a node $v$ in a connected graph $G$ is defined as the shortest distance to the farthest node from $v$ in $G$. Center of a graph which is a solution to the facility location problems, is calculated by picking the nodes with least eccentricity. Finding eccentricity of all nodes is as expensive as finding closeness, betweenness or stress centrality for all nodes in respect of time. Given a graph in the two dimensional euclidean space, if we were to solve the eccentricity ordering problem of two nodes in that graph without computing the
betweenness centralities we would go about the following way: drawing a minimum circle (Disk) covering all the nodes is a very well known problem called smallest-circle problem or minimum covering circle problem. A linear \((O(n))\) time randomized algorithm by Welzl\[6\] can find the smallest circle covering \(n\) points on a 2-D euclidean plane. Once we find the smallest circle, an approximate solution to the eccentricity comparison problem is to compare the distance from the center of smallest circle to the nodes. If the nodes are evenly distributed in the smallest circle, then the node closer to the center of that smallest circle is likely to have smaller eccentricity and vice versa. We just noted that we could solve the eccentricity ordering problem in linear time as opposed to finding it the conventional way by considering all possible distances from the given vertex to all other vertices.

The most common method for exact centrality ordering will calculate the centrality score of both the nodes and then compare the values in order to answer which one is more important. To reduce the time required for ordering, one can approximately compute the individual centrality scores and find the ordering with a high probability. There are two reasons why the current state of the art algorithms for exact calculation of the centrality measures will not be time efficient. Firstly because of the large size and the dynamic nature of networks. In large dynamic networks, we have to recompute the centrality scores each time the network changes, which is evidently expensive. Secondly because of the global characteristics of some centrality measures. For example, closeness centrality and degree centrality computation of a single node takes very less time as compared to the computation of the respective centrality measures for all the nodes in network. But unlike degree and closeness centralities, computing betweenness centrality of a node is conjectured to be as expensive as computing it for all the nodes in any network \([7]\). Since, computing the betweenness centrality of one node is equivalent to computing the betweenness centrality of all nodes according to the currently known deterministic algorithms, we are motivated to address the problem of betweenness ordering of two vertices.

Throughout this paper, we will restrict to the problem of finding the betweenness ordering of two vertices (here onwards called the betweenness-ordering problem). Betweenness centrality was proposed by Freeman \([8]\) and Anthonisse \([9]\) independently. Betweenness centrality of a node \(v\) is defined as the relative fraction of shortest paths passing through \(v\). It is calculated as \(BC(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}\), where \(\sigma_{st}\) is the total number of shortest paths from vertex \(s\) to vertex \(t\) and \(\sigma_{st}(v)\) is the total number of shortest paths from vertex \(s\) to vertex \(t\) passing through vertex \(v\). Unlike degree centrality, betweenness centrality covers more global characteristics and unlike closeness centrality works even on disconnected networks. Betweenness centrality has found many important applications in diverse fields. It has been used in the biological networks\([10]\), protein-protein interaction (PPI) networks \([11]\), analyzing communication system networks \([12]\), identifying critical nodes in the electrical & electronic systems (EES systems like Electronic Control Units used in vehicles)\([13]\), analyzing supply chain networks \([14]\), identifying bottleneck in supply chain networks\([15]\), planning a better public transit system networks; for example metro networks \([16]\), measuring load at a node in gas pipeline network \([17]\), waste-water disposal system networks etc.

We present a real world example that better motivates the betweenness-ordering-problem. Cascading failure in networks is a phenomenon where a node’s failure triggers the failure of successive nodes. After a node fails, the load of that node gets distributed among the neighbour nodes. This event may overload few neighbour nodes, which, in turn, may overload their neighbour nodes and goes on and on. This process of successive failure of nodes may cause the whole network to fail. Blackouts (power outage) are the most common outcomes of cascade failure. A list of major blackouts and other examples of cascading failure can be found at the wikipedia web page\([1]\). The most recent blackout due to cascading failure had happened in India on July 30, 2012. Blackout in Italy on September 28, 2003 resulted in a widespread failure of other dependent systems; railway network, health care systems, financial services, communication networks \([18]\). The cascading failure must be avoided. Several models have been proposed for cascading failure \([19]\), \([20]\), \([21]\), \([22]\). Some of them have shown that the failure of a node having high betweenness score may cause greater collapse of the network. Supposedly, two nodes in such a network failed at the same time, one should focus on the recovery/maintenance of the node with higher betweenness. This problem requires a faster betweenness ordering of those two nodes. Let Fig. 2 be a power grid network (a rectilinear grid of dimension \(2 \times 2\) where the number of nodes are \(9\) and the number of edges are \(12\)). The betweenness score of each node is written next to it. Now if nodes 2 and 8 fail at the same time and we have a single resource available to recover only one of them, then according to the above theory we should send the resource to node 2.

In this paper, we propose a novel non-uniform sampling technique which works very close to the optimal sampling explained in \([23]\). We approximately compute the betweenness score of a given nodes using the proposed sampling model incorporated within the approximation algorithm, given in section 3.

\[\text{http://en.wikipedia.org/wiki/Cascading\_failure}\]
The contribution of this paper are as follows.

1) We devise an approach for betweenness ordering of two nodes. First we develop a very efficient non-uniform sampling technique to choose the source nodes (also called pivot nodes) for single source shortest path computation. Then we use the model to approximately compute the betweenness score of a given node without computing betweenness of all the nodes in the graph.

2) To the best of our knowledge, it is the first study that focuses on the ordering of nodes based on centrality scores.

3) The developed sampling model provides the best approximation of the optimal sampling model given in [23].

4) We conduct extensive simulations on real and synthetic networks. The results show that our sampling model outperforms the uniform and distance based non-uniform sampling model. We measure the performance of proposed ordering approach using a performance tool defined in the paper and find that the proposed approach is very efficient.

We organize the rest of the paper as follows. In next section, we briefly review the approaches related to the betweenness computation. In section 3, we define basic terms used in the paper and explain the previous concepts, based on which we develop our sampling model. In section 4, we develop our model based on the analysis of random networks and some observations. All the details about simulations, data sets used in simulations, performance tools used for evaluation and extensive results in the form of plots and tables are compiled in section 5. We discuss the possible future directions of work in section 6. We conclude the paper in section 7.

A. Related work

Ordering of betweenness centrality can find several applications in various real scenarios. To the best of our knowledge, it is the first work which considers and motivates the study on the betweenness ordering of two nodes. Most of the studies done so far considered only computation of betweenness scores or ranking all the nodes based on their betweenness score. We discuss in brief about few of the computation algorithms. All the exact computation algorithms are based on either single source shortest path (SSSP) computation algorithms from all sources or all pair shortest path computation algorithms. The most trivial algorithm is a modified version of the Floyd-Warshall’s APSP algorithm [24, 25] to compute the betweenness scores for all nodes [8]. But this takes $O(n^3)$ time where $n$ is the number of nodes. In year 2001, Brandes [26] introduced an algorithm based on Dijkstra’s algorithm [27] which computes the exact betweenness score of all nodes in unweighted graphs in $O(mn)$ time, where $m$ is the number of edges.

Due to the size of current real world networks, even the state of art (Brandes’) algorithm was very expensive in terms of time. This motivated the researchers to develop faster exact or approximation algorithms. Several exact algorithms for large graphs (Sariyüce et al. [28]) and dynamic graphs (Lee et al. [29], Green et al. [30], Kas et al. [31], Goel et al. [32], Nasre et al. [33]) have been developed. These algorithms improved the computation time experimentally on special type of graphs but in worst case they all were as expensive as Brandes [26]. Several approximation algorithms were also proposed. These algorithms ran much faster and computed centrality scores close to the exact centrality scores. Two types of approximation algorithms exists. The First type consists of algorithms that focus on computing the approximate betweenness of all nodes together. The second type comprises of algorithms that approximate the betweenness score of a given node. Our goal is also to develop an approximation algorithm, therefore we summarize most of the approximation ideas developed so far for the betweenness computation.

Eppstein and Wang [34] first proposed the idea of sampling to approximately compute centrality indices for which SSSP computation is required from all the nodes. They suggested to compute SSSP from only a few nodes (called pivot) and discussed how to approximate the closeness centrality. Brandes and Pich [35] extended the idea of sampling given by Eppstein and Wang for approximating betweenness centrality. They gave different pivot selection strategies. SSSP from each pivot node were computed to estimate the contribution of each pivot node in the betweenness score of all nodes. By extrapolating the average contribution from pivot nodes, the approximate betweenness centrality was computed.

Bader et al. [36] proposed an adaptive sampling based approximation algorithm to compute the betweenness score of a given node. In his study, uniform probabilities were considered to sample the nodes. The number of sampled nodes were depended on the importance of the considered node, i.e. for highly central nodes, the algorithm requires to sample less number of nodes as compared to the nodes sampled for less central nodes. They also provided theoretical bound for their approximation algorithm. Geisberger et al. [37] generalized the approach given by Brandes and Pich [35] and observed that the betweenness centrality score of unimportant (less betweenness central) nodes which are near to pivot nodes, get overestimated. They provided an unbiased betweenness estimator framework which overcomes the observed problem. Gkorou et al. [38] developed two approximation approaches to compute approximate betweenness. Their first approach was for the dynamic networks and was based on the observation that very highly central nodes remain almost invariant over dynamic operations. The second approach was for large networks and considered only $k$-length shortest paths for the computation of approximate betweenness score. Their algorithms did not perform well on random graphs. Riondato and Kornaropoulos recently [39] developed two randomized algorithms to approximate betweenness score based on sampling of shortest paths. First algorithm approximates the betweenness score for all the nodes and the second algorithm approximates the betweenness score for top-$k$ nodes based on betweenness scores.

Recently, Chehreghani [23] proposed a new idea of approximating the betweenness score of a given node. He used
non-uniform sampling and then unlike [35], [37], he scaled the contributions from sampled nodes with respect to the probabilities. Finally, he averaged all the scaled values to achieve the approximate score. He used a very trivial model for generating the non-uniform probabilities but has not given any theoretical proof for the formula used in that model.

II. PRELIMINARY

In this section, we introduce some basic terms related to the betweenness centrality which have been used throughout the paper. We also discuss the previous concepts that motivated our sampling technique.

A. Terminology

We use following terms interchangeably: node or vertex and graph or network. For simplicity, we consider only undirected graphs until mentioned explicitly. All the concepts discussed in this paper can be easily generalized for weighted or directed graphs. Given a graph $G = (V, E)$, $V$ is the set of nodes with $|V| = n$ and $E$ is the set of edges with $|E| = m$. A (simple) path is a sequence of edges connecting a sequence of vertices without any repetition of vertices. The length of a path is the number of edges in the path. Shortest paths between two vertices are the smallest length paths between them. Distance between two nodes $i$ and $j$, $d(i,j)$, is the length of shortest path between $i$ and $j$.

Let $\sigma_{st}$ be the number of shortest paths between $s$ and $t$, for $s,t \in V$. Let $\sigma_{st}(v)$ be the number of shortest paths between $s$ and $t$ passing through $v$, for $v \in V$. Betweenness centrality score of a node $v \in V$ is calculated as:

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

Pair dependency of a pair of vertices $(s,t)$ on a vertex $v$ is defined as: $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$. Betweenness centrality of a vertex $v$ can be defined in terms of pair dependency as:

$$BC(v) = \sum_{s \neq t \neq v \in V} \delta_{st}(v).$$

Let $BFT_r$ denotes the breadth-first traversal (BFT) of the graph rooted on vertex $v$. In $BFT_r$, we assume that $v$ is at level 0 and the next levels are labelled by natural numbers in an increasing order. Dependency of a vertex $s$ on a vertex $v$ is defined as: $\delta_{s*}(v) = \sum_{t \in V \setminus \{s,v\}} \delta_{st}(v)$. Let us define a set $P^*(w) = \{v : v \in V, w is a successor of v in $BFT_r$\}$. Brandes [26] proved that:

$$\delta_{s*}(v) = \sum_{w : w \in P^*(w)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_{s*}(w)). \quad (1)$$

B. Previous work

In this section we will briefly describe the work of Chehrehgani [23]. He gave an approximation algorithm to compute betweenness score of a given node $v$ where a non-uniform sampling technique has been used to sample nodes. The algorithm is summarized as Algorithm 1. For a given node $v$, the algorithm takes the sampling probabilities as input and output the approximate betweenness score of node $v$. Step 2 initializes the betweenness score of node $v$, $T$ number of times and takes the average of all $T$ estimations. In each iteration of the algorithm, it samples a pivot node and computes the dependency of the pivot node on node $v$ using a single iteration of Brandes’ algorithm [26]. Then it estimates the betweenness score of node $v$ by, dividing (scaling) the computed dependency by the sampling probability of that pivot node.

Algorithm 1 Approximation algorithm to compute betweenness score of a given node $v$ [23].

1: Input. Graph $G$, probabilities $P = \{p_1, p_2, \ldots, p_n\}$, node $v$.
2: $BC(v) = 0$.
3: for $i=1$ to $T$ do
4: Select a node $i$ with probability $p_i$.
5: Compute $\delta_{i*}(v)$ in the $BFT_i$, using equation (1).
6: $BC(v) \leftarrow BC(v) + \frac{\delta_{i*}(v)}{p_i}$.
7: end for
8: $BC[v] \leftarrow BC(v)/T$.
9: Return. $BC(v)$.

He has motivated his paper with the idea of optimal sampling that is stated in the following theorem.

**Theorem 1.** Let the sampling probability assigned to each node $i$ be

$$p_i = \frac{\delta_{i*}(v)}{\sum_{j=1}^{n} \delta_{j*}(v)}$$

then, betweenness score of node $v$ can be exactly calculated in $O(m)$ time using single iteration of Algorithm 1.

**Proof:** In Algorithm 1, each iteration calculates $\delta_{i*}(v)$ value as the estimated betweenness score of node $v$. If we consider $p_i = \frac{\delta_{i*}(v)}{\sum_{j=1}^{n} \delta_{j*}(v)}$, then in single iteration we get

$$\frac{\delta_{i*}(v)}{p_i} = \sum_{j=1}^{n} \delta_{j*}(v),$$

which is the exact betweenness of node $v$.

We refer the probability defined in Lemma 1 as optimal probability and call a model optimal model (OPT) if it can generate optimal probabilities. Calculating optimal probabilities is as expensive as computing exact betweenness using Brandes’ algorithm [26]. Thus some model was desired which can generate the probabilities close to optimal probabilities $(p_i \propto \delta_{i*}(v))$ very fast.

Chehrehgani noted that any such model should satisfy at least the following relation for most of the vertex pairs $(i,j)$:

$$p_i < p_j \iff \delta_{i*}(v) < \delta_{j*}(v) \quad (2)$$

Chehrehgani has given a simple distance based model (DBM) [23] to generate the sampling probabilities. He proposed to take the probabilities as the normalized value of the inverse of distance from node $v$ to node $i$.
be given as:

\[ \alpha_{m+1} \approx np\left(1 - \frac{\sum_{j=0}^{m} \alpha_j}{n}\right)\alpha_m. \]  

(3)

Proof of Lemma 1:
Van Der Hofstad \[41\] explained the BFS traversal as Exploration Technique (ET) in random graphs. In this technique all the vertices are initially inactive except \(i\) (the root node on which ET has to be applied). The vertex \(w\) is chosen which was discovered first among the current active vertices, and its neighbourhood is explored for inactive vertices. All the inactive vertices found are made active. Node \(w\) is made inactive and is labelled as processed. In the paper we refer exploring the neighbourhood of \(t\)th vertex as \(t\)th exploration.

Let \(S_t\) be the number of active vertices after \(t\)th exploration, and \(X_t\) be the number of vertices discovered (converted from inactive to active) in \(t\)th exploration. Then, following relation holds for any iteration (exploration) \(t\):

\[ S_t = S_{t-1} + X_t - 1 \]  

(4)

After \(t - 1\) explorations, we are left with \(n - (t - 1) - S_{t-1}\) vertices \((t - 1)\) processed vertices, \(S_{t-1}\) active vertices. If \(p\) is the probability of existence of an edge between any two nodes in the graph then we have:

\[ X_t \sim Bin(n - (t - 1) - S_{t-1}, p) \]  

(5)

Next, we state a very well known binomial relation in mathematics as lemma. Then we will use it to compute the expected number of nodes at a level of BFS traversal.

Lemma 2. If \(X \sim Bin(n, p)\) then, \(Pr(X = k) = \binom{n}{k}p^k(1-p)^{n-k}\). The expected value of \(X\), \(E[X]\) is:

\[ E[X] = \sum_{k=1}^{n} k \binom{n}{k}p^k(1-p)^{n-k} = np. \]

Using lemma 2 we can write the expected value of \(X_t\) as:

\[ E[X_t] = (n - (t - 1) - S_{t-1})p \]  

(6)

Equation 6 can be used in the following way to calculate the expected number of nodes at a BFS level. Initially, there is a single (source) node as an active node i.e. \(S_0 = 1\). The expected number of nodes discovered in the first exploration will be:

\[ E[X_1] = (n - 1)p \]

After first exploration, the total number of active nodes will be \(S_1 = (n - 1)p\). The expected number of nodes discovered in the second exploration will be \((n - 1 - (n - 1)p)p\) or

\[ E[X_2] = (n - 1)(1 - p)p. \]

Similarly, we can calculate following values: \(S_2 = (n - 1)(2 - p)p - 1, E[X_3] = (n - 1)(1 - p)^2p, E[X_4] = (n - 1)(1 - p)^2p\).
\[(n-1)\sum_{k=1}^{n-1}(1-p)^k p = (n-1)(1-p)[1-(1-p)^{(n-1)p}]\]

Now, we can derive the formula for the general case. Let \(\alpha_m\) be the expected number of nodes at level \(m\). Exploring the first vertex of level \(m\) discovers \((n - \sum_{j=0}^{m} \alpha_j)\) nodes for level \(m + 1\). Exploration of the next vertex discovers \([n - \sum_{j=0}^{m} \alpha_j] - (n - \sum_{j=0}^{m} \alpha_j)p\) nodes and so on. So the expected number of nodes at level \(m + 1\) will be:

\[
\alpha_{m+1} = \sum_{k=0}^{m-1} (n - \sum_{j=0}^{m} \alpha_j)(1-p)^k p
\]

or

\[
\alpha_{m+1} = (n - \sum_{j=0}^{m} \alpha_j)[1 - (1-p)^{\alpha_m}]
\]

Most of the real world graphs are sparse. Hence, without the loss of generality, we can assume that \(p \ll 1\). Applying Binomial expansion and neglecting higher order terms of \(p\), we can rewrite equation (8) as:

\[
\alpha_{m+1} \approx (n - \sum_{j=0}^{m} \alpha_j)\alpha_m p = n(1 - \sum_{j=0}^{m} \alpha_j n)\alpha_m p.
\]

Equation (3) is a recurrence relation to estimate the number of nodes at some level \(m\). Using Lemma 4, we can estimate the ratio between the expected number of nodes at two consecutive levels. The ratio is given in the Lemma 5:

**Lemma 4.** Let \(\alpha_m\) and \(\alpha_{m+1}\) be the number of nodes at level \(m\) and \(m+1\) respectively. Then we have

\[
\frac{\alpha_{m+1}}{\alpha_m} \approx c_{m+1}\lambda
\]

where \(c_{m+1} \in [0,1]\).

Based on Lemma 3, we derive the formula to calculate the expected dependency of a node \(i\) on node \(v\), \(E[\delta_i(v)]\) in next lemma. Then, we establish the ratio between the expected dependency of root node \(i\) on two nodes at consecutive levels in Theorem 2.

**Proof:** Let the BFS traversal rooted at \(i\) consist of \(m + 1\) levels. If \(v\) is at the last level \((m)\), then \(\delta_i(v) = 0\). Now, if \(v\) lies at level \(m - 1\), then we can compute the expected dependency \(E[\delta_i(v)]\) as follows. Let \(A_m\) be the expected number of paths of length \(m - 1\) from \(i\) to any vertex at level \(m - 1\). Bauchkage et al. [22] gave following expression for \(A_m\):

\[
A_m = n^{m-2}n^{m-1}
\]

where \(\pi = \frac{n-1}{n}\). It is simple to observe that any node \(w\) at level \(m\) has \(\alpha_{m-1} \cdot p\) parents (nodes at level \(m - 1\) which are connected to \(w\) by a direct edge), so the expected number of shortest paths from \(i\) to \(w\) will be \(A_m \cdot \alpha_{m-1}p\). Node \(v\) lies only on \(A_m\) paths out of those shortest paths. From equation (1), it is easy to observe that the expected partial dependency from node \(i\) to node \(v\) on node \(v\) is \(E[\delta_{iv}(v)] = \frac{1}{\alpha_{m-1}p}\). Node \(v\) has \(\alpha_m\) children similar to \(w\). Therefore the expected dependency of node \(i\) on node \(v\) is \(E[\delta_i(v)] = \frac{\alpha_m}{\alpha_{m-1}}\) or using equation (10) we can rewrite:

\[
E[\delta_i(v)] = c_m\lambda.
\]

Similarly, if \(v\) lies at level \(m - 2\), then the expected dependency of node \(i\) on node \(v\) can be given as:

\[
E[\delta_i(v)] = \left(\frac{\alpha_{m-1}}{\alpha_{m-2}}\right)(1 + c_m\lambda) = c_{m-1}\lambda(1 + c_m\lambda).
\]

**Theorem 2.** Let \(l\) be the last level in \(BFS_i\). Let \(\delta_{i,v_l-1}\) be the dependency of node \(i\) at a node \(v_l-1\) at level \(l - k\) and let \(\delta_{i,v_l-1}\) be the dependency of node \(i\) at a node \(v_l-1\) at level \(l - k + 1\). Then we have

\[
\frac{\delta_{i,v_l+1}}{\delta_{i,v_l-1}} = c_{l-k+1}\left(\frac{1}{\phi} + \lambda\right)
\]

where \(\phi = (c_l-k+1)(1+c_l-k+3\lambda(1+c_l-k+4\lambda(1+c_l-k+5\lambda(1+\cdots(1+c_l\lambda))\cdots)).

**Proof:** The ratio of expected dependencies of node \(i\) on \(v\), when \(v\) lies at level \(l - 2\), \(E[\delta_i(v_{l-2})]\) to when \(v\) lies in
level \( l - 1 \), \( E[\delta(v_{l-1})] \) is
\[
\frac{\delta_{\bullet}(v_{l-2})}{\delta_{\bullet}(v_{l-1})} = \frac{c_{l-1}}{c_l}(1 + c_l \lambda), \tag{15}
\]
In general, the ratio of the expected dependencies for two successive levels \( l - k \) and \( l - k + 1 \) can be given as equation \( \text{[14]} \).

It is simple to observe that \( c_m \) decreases continuously as \( m \) increases. So as \( v \) becomes closer to \( i \), \( E[\delta(v)] \) increases steeply, proportional to the average degree \( \lambda \). Therefore, on the basis of lemma \( \text{[2]} \), we can assign a probability \( p_i \) to the node \( i \) as in the following theorem.

**Theorem 3.** Suppose, we have to compute the betweenness score of node \( v \). Then the sampling probability assigned to node \( i \) is:
\[
p_i \propto (\lambda)^{-d(i,v)} \tag{16}
\]
where \( d(i,v) \) is the distance between \( v \) and \( i \).

**B. Further Tweak**

In this section, we discuss some of the observations and propose some possible solutions to tackle an observed problem.

In \( BFT_v \), nodes at same level are called siblings. We define successors of a node \( j \) in \( BFT_v \), \( \text{Succ}_v(j) \), as the set of nodes to which at least one shortest path from \( v \) passes through \( j \). We can also define number of successors of a node \( j \) as the number of nodes in the subgraph (sub traversal) rooted at node \( j \) in \( BFT_v \).

**Observation 1.** In the BFS tree rooted at the given node \( v \), nodes at the same level get equal probabilities by equation \( \text{[16]} \) but might not contribute equally in the betweenness of \( v \).

For example, Fig. 3 shows a BFS traversal rooted at node \( v \) of some graph. We can notice that node \( i \) and node \( j \) are at the same level. So, according to Equation \( \text{[16]} \), equal probabilities will be assigned to both nodes. But \( \delta_{\bullet}(v) = 1 \) and \( \delta_{\bullet}(v) = n - 2 >> 1 \). Thus we need to tweak the formula to resolve this problem.

**Observation 2.** In \( BFT_v \), no node from \( \text{Succ}_v(i) \) will contribute in \( \delta_{\bullet}(v) \).

Observation 2 infers that in \( BFT_v \), a node \( i \) with larger number of successors will contribute \( (\delta_{\bullet}(v)) \) lesser i.e. the relation can be assumed as \( \delta_{\bullet}(v) \propto \frac{1}{\text{Succ}_v(i)} \). Then based on the assumption, the probability assigned to node \( i \) should also satisfy the following relation:
\[
p_i \propto \frac{1}{\text{Succ}_v(i)}. \tag{17}
\]

Counting the number of successors of all nodes in \( BFT_v \) (graph) can not be achieved in linear time. At place of \( \text{Succ}_v(i) \), we use degree of node \( i \). One reason for using degree is that it can be linearly computed and most of the time, it is the best forecast for the number of successors in random graphs. Another reason is the high correlation between the Betweenness centrality and the Degree centrality \( \text{[43]}, \text{[44]} \).

Thus, to overcome the problem stated in Observation 1, we will also include the following relation:
\[
p_i \propto \frac{1}{\text{deg}(i)} \tag{17}
\]
where \( \text{deg}(i) \) is the degree on node \( i \) in the given graph. Thus the final relation can be written as
\[
p_i \propto \frac{\lambda^{-d(i,v)}}{\text{deg}(i)} \tag{17}
\]

We use the distance as power in the exponential function over average degree, and degree for modelling the probability generation. Thus we name this model EDDBM (exponential in distance and degree based model). Next, we discuss the steps to generate the probabilities according to EDDBM.

**C. EDDBM**

We generate the probabilities as following. First, we generate the probabilities on the basis of distance relation given in Equation \( \text{[16]} \). Each node \( j \) at level \( d \) in the \( BFT_v \) will get following probability values:
\[
p^d = \frac{(\lambda)^{-d}}{\sum_{j \in V \setminus \{v\}} (\lambda)^{-d}} \tag{18}
\]
Let \( V_d \) be the set of nodes at level \( d \) in the \( BFT_v \) and \( |V_d| \) denotes the number of nodes in set \( V_d \). Then to resolve the problem stated in Observation 1 to best extent, at each level \( d \), we further tweak the formula on the basis of Equation \( \text{[17]} \) and get the assigned probability to node \( i \) at \( d^{th} \) level as:
\[
p_i = \frac{p^d|V_d| \cdot \text{deg}(i)^{-1}}{\sum_{j \in V_d} \text{deg}(j)^{-1}}. \tag{19}
\]

**IV. EXPERIMENTAL RESULTS**

In this section, we discuss the experimental results achieved on extensive real world graphs and synthetic graphs. We have implemented the algorithms in Python Version 2.7.3 and used Networkx library for graph functions. All the simulations were performed on a 32 bit Ubuntu machine with 3.00GHz Intel Core 2 Duo E8400 processor and 3.4 GB RAM. We have not discussed the execution time of Algorithm 1 with our model. It is same \( \text{[O(Tm)]} \) as in \( \text{[23]} \).
A. Data set

1) Real Networks: We have picked several real world networks. \[45\], \[46\], \[47\] can be referred for the description about the considered networks and data. We have shown the betweenness computation results on real graphs in the Table I. Details about the considered graphs and source of the data set are mentioned in the table. We have considered collaboration networks, citation networks, power grid networks, city networks, airline network, road network, random bench-marked networks and many more.

2) Synthetic Networks: For synthetic graphs we considered following types of graphs:

   1) Random Graphs (ER) \[48\]. For generating random graphs, we have considered the most extensively used random graph generation \(G(n, p)\) model given by Erdos Renyi. The model takes as input the number of nodes \(n\) and a probability \(p\). Then for each possible pair of different nodes, it puts an edge with a probability of \(p\) and outputs the generated graph. We also referred this probability as edge existential probability in this paper.

   2) Scale-free Random Graphs (BA) \[49\]. For generating scale-free random graphs, we have considered the Albert Barabasi graph generation model. This model takes as input the number of nodes \(n\) and an integer \(k\). It starts with a complete graph of size \(k\) and keep adding random \(k\) different edges from new coming nodes to the existing nodes with probability as the normalized degree of old nodes. Thus this model is also referred many a times as preferential attachment model.

   3) Small World Graphs (WS) \[50\]. We considered Watts Strogatz model to generate small world graphs. This model takes three values as input; number of nodes \(n\), integer \(k\) and probability \(p\). It first generates a \(k\)-regular graph \((k - 1\) if \(k\) is odd) of size \(n\) which can be represented by ring lattice. Then, it replaces each edge with a new edge between a random pair of nodes with probability \(p\).

B. Performance Measurement Tools

In this section we will define various measures used to evaluate the performance of our model. These tools can be used for any betweenness computation or comparison algorithm to measure its performance.

For Betweenness Computation : Error and Average Error: Let a graph \(G = (V, E)\) with \(|V| = n\) is given. Let \(BC^e(v)\) be the exact betweenness score of node \(v\) in a given graph. Let \(BC^a(v)\) be the betweenness score of the same node \(v\) computed by Algorithm 1 using probabilities generated by our model. Then, we define error in computation of betweenness score on node \(v\) same as Chehreghani\[23\]:

\[
Er(v) = \frac{|BC^e(v) - BC^a(v)|}{BC^e(v)} \times 100
\]

We can define average error \(E\) in the computation of betweenness score of a set of nodes \(U\), \(U \subseteq V\), over a graph \(G\) as

\[
E = \frac{\sum_{i \in U} Er(i)}{|U|}
\]

where \(|U|\) denotes the number of nodes in set \(U\). Number of iterations used for computation of betweenness score is also referred as number of sampled nodes. We denote it by \(T\), where \(T\) is the percentage of total number of nodes \(n\).

2) For Betweenness Ordering : Efficiency and Relaxed Efficiency: Let \(n\) be the number of nodes in the considered graph. Then, \({n\choose 2}\) different pairs of nodes are possible. Let \(b_{ij} = 1\) if the result of betweenness comparison between node \(i\) and node \(j\) by our algorithm is correct, otherwise \(b_{ij} = 0\). The efficiency of algorithm for betweenness-ordering of two nodes can be given as

\[
\xi = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij}}{\binom{n}{2}}
\]

In real world scenarios when two nodes posses very close betweenness ranks, error in the betweenness-ordering of those two nodes does not matter much. For example importance of the top central node or second top central node is very close. Thus, a relaxed version of the efficiency measure can be modelled. We take a threshold \(t\) and relax the ordering of nodes if the difference between the betweenness ranks of both the nodes is less or equal to \(t\). By relaxing, we mean that we do not consider those pairs for measuring the efficiency of algorithm. Let \(P_t\) be the set of all pairs of nodes with betweenness rank difference greater than \(t\) and \(|P_t|\) denotes the cardinality of set \(P_t\). Let \(b_{ij}\) be a flag variable which gets value 1 for correct comparison and 0 otherwise. Then we can redefine the relaxed efficiency as:

\[
\xi^t = \frac{\sum_{(i,j) \in P_t} b_{ij}}{|P_t|}
\]

At \(t = 0\), \(\xi^t = \xi\). Next, we see various plots for the analysis of our model.

C. Plots

In this section we evaluate the performance of EDDBM through various plots. With the help of different plots on synthetic networks, we compare the accuracy of EDDBM with DBM.

![Fig. 5](image-url)
1) Comparison of probabilities assigned by DBM, EDDBM and optimal model: Plots in this section support our model and shows that EDDBM generates probabilities very close to the optimal probabilities. These plots also compare EDDBM with DBM. The first plot in Fig. 4 is drawn for a synthetic scale free graph with \( n = 200 \) and \( k = 5 \). Here, the x-axis represents the nodes labelling and the y-axis represent the probabilities assigned by DBM, EDDBM and the optimal model. The next two plots in Fig. 5 are drawn for a random synthetic graph with \( n = 200 \) and \( p = 0.05 \). In both the plots in Fig. 5 the x-axis represents the nodes sorted in the order of their optimal probabilities and the y-axis represents the probabilities assigned by DBM / EDDBM and the optimal probabilities. In both the figures Fig. 4 and Fig. 5 it is easy to observe that EDDBM is much better than DBM and EDDBM generates probabilities very close to the optimal probabilities. In Fig. 4 we also note that the plot of probabilities by EDDBM achieves similar characteristic peaks as the plot of optimal probabilities get. This infers that, unlike DBM, EDDBM identifies the nodes with high contribution and assigns large probabilities to them. EDDBM also focuses on the nodes contributing very less and tries to assign smaller probabilities to them which was not well handled by DBM.

2) Average error in the top \( k \) betweenness nodes vs \( k \): We plot the average error in the computation of betweenness centrality of top \( k \) betweenness central nodes. On the x-axis we vary \( k \) from 1 to \( n \), where \( n \) is the number of nodes. On y-axis, we draw the plots for the average error in the betweenness computation of top \( k \) nodes by Algorithm 1 which uses probabilities generated by DBM / EDDBM. The average error is computed by the formula given in section [IV-B1].

In Fig. 6(a) we plotted the average results achieved on a random graph with \( n = 100 \) and \( p = 0.08 \). Fig. 6(b) shows the average results on a scale free graph with \( n = 100 \) and \( k = 5 \). Both the plots in Fig. 6 show that, unlike DBM, our model does not show biasing towards high or less central nodes in the computation of betweenness. Using DBM, the algorithm produces very small error in the computation of betweenness of top central nodes and larger error for less central nodes.

3) Average Error vs Number of Sampled Nodes (T): In this section we plot the average error in the computation of betweenness centrality using EDDBM when the number of sampled nodes (no of iterations) were \( T = X\% \) of the total number of nodes. These plots were drawn to inspect what value of \( T \) can suffice for good result i.e. betweenness computation with less error. In Fig. 7 there are two plots. One for Erdos Renyi graph with \( n = 100 \) and \( p = 0.05 \) in Fig. 7(a) and the other for Barabasi Albert graph with \( n = 100 \) and \( k = 2 \) in Fig. 7(b).

For simplicity, at \( X = 0 \), we have not sampled any node, we considered 100% error. In both the plots in Fig. 7 the average error reduces very sharply when \( X \) varies from 1.
to 4 or 5. After $X = 10$ there is very small change in the average error. Thus, to achieve several of the experimental results in this paper, we have considered $X = 10$ for small graphs (with nodes less than 1000) and smaller values of $T$ for larger graphs (with nodes greater than 1000). By reducing $T$, the error should increase in larger graphs but in larger graphs our model performs much better. The reason is given in next plots that the average error decreases with increase in number of nodes which neutralizes the effect of increase in error by reducing $T$.

Fig. 8. Average Error vs Size of random and scale free graphs

4) Average Error vs Size of Graph ($n$) : In this section we plot the average error in the computation of betweenness score in a graph with respect to the size of graph (number of nodes in that graph). We generated graphs with $n = 100$ to $n = 500$ with a step of 50. For each $n$, we generate 10 graphs and averaged the average error over all 10 graphs. Fig. 8 contains two such plots. In Fig. 8(a) we plotted the results for Erdos Renyi graphs with average degree = 5. In Fig. 8(b) we plotted the results for Barabasi Albert graphs with $k=2$. From the plots, we can infer that the average error in computation of betweenness score decreases with increase in number of nodes.

Fig. 9. Efficiency of Algorithm 1, when used with Uniform sampling model, DBM, and EDDBM vs Size of Graphs

5) Correlation in ordering / Average efficiency vs Size of Graphs ($n$): In this section we plot the average efficiency of Algorithm 1, when aggregated with our model, with respect to the number of nodes in the graphs as shown in Fig. 9. We have plotted the average efficiency for $n = 50$ to $n = 200$ with a step of 10. For each $n$, we generated 10 graphs and then averaged the efficiency over 10 values. Efficiency exhibits the fraction of correct comparisons over all possible comparisons in a graph and can be computed by the formula given in section IV-B2. These plots investigate the average performance of our model for betweenness-ordering of two nodes. We plotted the efficiency of DBM and the uniform sampling model. Our model performed the best out of the considered models.

We also plotted the correlation (EDDBM-C) between the betweenness ranking of vertices by our model and their exact betweenness rank to evaluate the accuracy of ordering. Ordering all nodes by computing betweenness score for each node by Algorithm 1 and then sorting is very expensive than exactly computing the betweenness scores using Brandes’s algorithm for all nodes and then sorting. The correlation plot is just to understand the accuracy of Algorithm 1, when used with EDDBM. As we can see in the plots, the efficiency of our model for betweenness-ordering in even the small sized graphs is very high (greater than 0.90) and keeps increasing with the size of graph. Even for graph with size 200, it reaches very close to 0.95 which will further increase in large sized graphs. The correlation between the ordering achieved by our algorithm and the exact ordering is also very high and reached very close to 1 for graphs of size just 200.

6) Average Relaxed efficiency ($\xi^t$) vs $t$: In this section we will inspect the plot between the relaxed efficiency $\xi^t$ and $t$. The relaxed efficiency is computed by the formula given in section IV-B2. We vary $t$ from 0 to 10. The plots are compiled in Fig. 10. In Fig. 10(a) for each $t$, we generate 10 random Erdos Renyi graphs with $n = 100$ and $p = 0.05$ and then calculate the relaxed efficiency on each graph and average them to get average relaxed efficiency. Similarly, in Fig. 10(b) we plotted average relaxed efficiency for scale free Barabasi Albert graph with $n = 100$ and $k = 4$. In both plots we compared EDDBM with DBM and uniform sampling model. Our model outperformed all considered models with a very good margin. At $t = 0$, we get the exact efficiency of our model. The plots infer that the efficiency increases with an increase in $t$ i.e. in real world situation where relaxation is allowed in ordering, our algorithm will perform much better than its usual performance. The relaxation in ordering means that betweenness ordering of the nodes with approximately same betweenness rank is ignored and only the betweenness
### TABLE I

**AVERAGE ERROR IN SYNTHETIC GRAPHS**

<table>
<thead>
<tr>
<th>Instance</th>
<th>AD</th>
<th>T</th>
<th>NONC</th>
<th>DBM</th>
<th>EDDBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barabasi(1000,2)</td>
<td>3.99</td>
<td>10%</td>
<td>Top 10%</td>
<td>17.33%</td>
<td>6.32%</td>
</tr>
<tr>
<td>Barabasi(500,2)</td>
<td>3.98</td>
<td>10%</td>
<td>All</td>
<td>40.52%</td>
<td>13.16%</td>
</tr>
<tr>
<td>Barabasi(500,5)</td>
<td>9.9</td>
<td>10%</td>
<td>All</td>
<td>37.20%</td>
<td>7.38%</td>
</tr>
<tr>
<td>Erdos_renyi(500, 0.016)</td>
<td>7.86</td>
<td>10%</td>
<td>All</td>
<td>27.56%</td>
<td>4.37%</td>
</tr>
<tr>
<td>Erdos_renyi(500, 0.008)</td>
<td>3.97</td>
<td>10%</td>
<td>All</td>
<td>21.72%</td>
<td>7.39%</td>
</tr>
<tr>
<td>watts_strogatz(500,4,2)</td>
<td>4</td>
<td>10%</td>
<td>All</td>
<td>19.72%</td>
<td>13.23%</td>
</tr>
<tr>
<td>watts_strogatz(500,6,3)</td>
<td>6</td>
<td>10%</td>
<td>All</td>
<td>21.32%</td>
<td>9.52%</td>
</tr>
</tbody>
</table>

### TABLE II

**AVERAGE ERROR IN REAL GRAPHS**

<table>
<thead>
<tr>
<th>Instance</th>
<th>n</th>
<th>m</th>
<th>AD</th>
<th>T</th>
<th>NONC</th>
<th>DBM</th>
<th>EDDBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gset/G1 [47]</td>
<td>800</td>
<td>19176</td>
<td>47.94</td>
<td>10%</td>
<td>All</td>
<td>24.04%</td>
<td>4.33%</td>
</tr>
<tr>
<td>Gset/G10 [47]</td>
<td>800</td>
<td>19176</td>
<td>47.94</td>
<td>10%</td>
<td>All</td>
<td>24.62%</td>
<td>3.88%</td>
</tr>
<tr>
<td>Gset/G14 [47]</td>
<td>800</td>
<td>4694</td>
<td>11.74</td>
<td>10%</td>
<td>All</td>
<td>33.48%</td>
<td>7.13%</td>
</tr>
<tr>
<td>Graph &amp; Digraph Glossary  [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G'D99-Linden Strasse [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mangwet Food Web [51], [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Florida Food Web [51], [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erdos971 [47]</td>
<td>433</td>
<td>1314</td>
<td>6.14</td>
<td>10%</td>
<td>All</td>
<td>28.24%</td>
<td>18.79%</td>
</tr>
<tr>
<td>Baydry Food Web [51], [45]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barfiba980 [47]</td>
<td>480</td>
<td>16408</td>
<td>68.37</td>
<td>10%</td>
<td>All</td>
<td>22.53%</td>
<td>4.99%</td>
</tr>
<tr>
<td>US Air lines [45]</td>
<td>332</td>
<td>2126</td>
<td>12.81</td>
<td>10%</td>
<td>All</td>
<td>35.22%</td>
<td>11.45%</td>
</tr>
<tr>
<td>World Cities [52]</td>
<td>415</td>
<td>7518</td>
<td>36.41</td>
<td>10%</td>
<td>All</td>
<td>24.04%</td>
<td>10.07%</td>
</tr>
<tr>
<td>GD'01-GD proceedings Self Citing [45]</td>
<td>311</td>
<td>640</td>
<td>4.94</td>
<td>10%</td>
<td>All</td>
<td>26.36%</td>
<td>15.85%</td>
</tr>
<tr>
<td>Newman/celegans neural [53], [50], [47]</td>
<td>297</td>
<td>2148</td>
<td>14.46</td>
<td>10%</td>
<td>All</td>
<td>42.02%</td>
<td>13.03%</td>
</tr>
<tr>
<td>Newman/Dolphins [54]</td>
<td>82</td>
<td>159</td>
<td>5.13</td>
<td>10%</td>
<td>All</td>
<td>37.97%</td>
<td>26.69%</td>
</tr>
<tr>
<td>Newman/PollBooks [55]</td>
<td>105</td>
<td>441</td>
<td>8.4</td>
<td>10%</td>
<td>All</td>
<td>48.27%</td>
<td>36.11%</td>
</tr>
<tr>
<td>Pajek/GD00_C [45]</td>
<td>638</td>
<td>1020</td>
<td>3.02</td>
<td>10%</td>
<td>All</td>
<td>24.94%</td>
<td>9.62%</td>
</tr>
<tr>
<td>Pajek/GD01_Acap [45]</td>
<td>259</td>
<td>640</td>
<td>4.94</td>
<td>10%</td>
<td>All</td>
<td>26.66%</td>
<td>18.50%</td>
</tr>
<tr>
<td>Pajek/Roget [56], [45]</td>
<td>1022</td>
<td>4643</td>
<td>9.19</td>
<td>10%</td>
<td>Top 10%</td>
<td>51.60%</td>
<td>21.11%</td>
</tr>
<tr>
<td>Pajek/GD96_A [45]</td>
<td>1096</td>
<td>1677</td>
<td>3.06</td>
<td>10%</td>
<td>Top 10%</td>
<td>25.49%</td>
<td>20.31%</td>
</tr>
<tr>
<td>Pajek/SmaGri [45]</td>
<td>1024</td>
<td>4916</td>
<td>9.6</td>
<td>10%</td>
<td>Top 10%</td>
<td>20.90%</td>
<td>7.67%</td>
</tr>
<tr>
<td>Pajek/GD06_C JAVA [45]</td>
<td>1538</td>
<td>7817</td>
<td>10.17</td>
<td>10%</td>
<td>Top 10%</td>
<td>31.27%</td>
<td>9.81%</td>
</tr>
<tr>
<td>Pajek/Yearst [47], [45]</td>
<td>2284</td>
<td>6646</td>
<td>5.82</td>
<td>5%</td>
<td>Top 200</td>
<td>20.51%</td>
<td>11.22%</td>
</tr>
<tr>
<td>HB/bcsstk08 [47]</td>
<td>1071</td>
<td>5943</td>
<td>11.1</td>
<td>5%</td>
<td>Top 10%</td>
<td>41.60%</td>
<td>30.32%</td>
</tr>
<tr>
<td>Arenas/Email [42]</td>
<td>1133</td>
<td>5451</td>
<td>9.62</td>
<td>5%</td>
<td>Top 10%</td>
<td>27.35%</td>
<td>13.25%</td>
</tr>
</tbody>
</table>

ordering of nodes with difference greater than the threshold value (t) in their betweenness ranks are considered. Thus, we can say that our algorithm results ordering very close to the exact betweenness-ordering.

In next sections, we will discuss the betweenness computation results achieved for some synthetic networks and a number of real world networks.

**D. Average error in Graphs**

Here, we discuss and compare the results achieved by Algorithm 1 when it takes probabilities from our model (EDDBM) and Chehreghani’s model (DBM) on some synthetic graphs and several small and moderate size real graphs.

1) **Average error in synthetic graphs**: In this section, we will analyze the results over some synthetic graphs. The results are summarized in Table I. The first column gives the description of graph considered. Second column contains the average degree (AD) of nodes in the instance graph. Next column is for the number of sampled nodes (T) in terms of percentage of the total nodes n. Next column informs us about the number of nodes considered (NONC) for calculating the average error. The last two columns contain the average error in computation of betweenness due to our model (EDDBM) and average error due to Chehreghani’s model (DBM).

We considered three scale free Barabasi Albert graphs, two random Erdos Renyi graphs and two small world Watts Strogatz graphs. We generated at least one sparse and one dense graph from each model. It is easy to observe that EDDBM performed much better than DBM. Our model performs better in denser graph than in sparser graphs. The point discussed in section IV-C4 can be again noted here. The average error decreased to half when the number of nodes got doubled in Barabasi Albert graph. In the considered instances we reduced the error due to DBM by a maximum of 6.31 times and an average of 3.4 times.

2) **Average error in real graphs**: This section presents and discusses the extensive simulation on real networks. The real networks were picked from [45], [47], [46]. After extracting
the networks, we converted the networks into unweighted undirected networks, if required. Then we removed multi-edges, self-loops and isolated nodes if existing. The results obtained is summarized in the Table IV. The columns are in similar order as in the Table I except there are two more new column entries after the first column. The first new column contains the number of nodes (n) and the second one contains the number of edges (m) in the corresponding networks.

EDDBM performed again better than DBM. In some considered data sets, we reduced the error by more than 6.3 times and on an average reduction of 2.9 times. This error achieved by EDDBM is very less and will decrease in large size graphs. The fact is already observed in section IV-C4.

E. Result Summary

We have evaluated our model through several performance measures and discovered following points:

1. EDDBM assigns probability very close to optimal model.
2. The average error, in the betweenness computation of a given node by Algorithm 1 and EDDBM, decreases with increase in the size of graph.
3. The average error, in the betweenness computation of a given node by Algorithm 1 and EDDBM, decreases with increase in the number of sampled nodes but become constant after a threshold.
4. Efficiency of Algorithm 1, when used with EDDBM, is very high i.e. it orders two nodes based on betweenness centrality in a given graph correctly with a very high probability.
5. Relaxed Efficiency is higher than efficiency which motivates the use of our model for real graphs.

The formulation of EDDBM is based on the analysis of random graphs. Random graphs does not posses high clustering coefficient and thus this model does not perform well on the graphs with high clustering coefficient.

V. CONCLUSION AND FURTHER WORK

In this paper, we coin a new problem called the betweenness-ordering-problem and address its importance with real world examples and provide a feasible and practical solution to it. We present the proof of concept of our technique by applying it to real world graphs as well as synthetic ones.

To the best of our knowledge, this is the first of its kind study addressing the "ordering problem" in centrality measures. While our work is a first attempt to provide a solution to the centrality-ordering-problem for the betweenness measure, we feel this should lead to the asking and answering of this question across several popular measures that have seen its applications in diverse areas.

- By our problem statement, betweenness-ordering-problem refers to the ordering of two nodes. Can one solve the betweenness-ordering-problem for k nodes in general?
- One can attempt to ask a similar question for the Pagerank-ordering-problem, closeness-ordering-problem or any centrality-ordering-problem. Any attempt to address these problems in the same spirit as our addressing the betweenness-ordering-problem would collectively be a great contribution to applied sciences where centrality measures are being increasingly applied.

REFERENCES
