The Delta-Betweenness Centrality

Alexander Plutov and Michael Segal, Senior Member, IEEE

Communication Systems Engineering Department
Ben-Gurion University of the Negev

Abstract—In this paper we consider the extension of the betweenness centrality measure which is used in social and computer communication networks to estimate the potential monitoring and control capabilities a node may have on data flowing in the network. Unlike the standard betweenness centrality measure which takes into account only the shortest paths between the nodes of given network, our, so-called $\Delta$-betweenness centrality measure is based on short paths between the nodes. We present an efficient algorithm for computing this measure and show experimental results supporting the importance of newly defined measure.

I. INTRODUCTION

In order to determine the relative importance of every node in a given network (or graph), various centrality measures such as degree, closeness, and betweenness [1, 4, 5] have been proposed. In this paper we extend the standard definition of betweenness centrality measure, defined by Freeman [1] in order to estimate the control an individual may have over communication flows in social networks. For a given undirected, unweighted graph $G=(V,E)$, denote the number of shortest paths from $s \in V$ to $t \in V$ by $\sigma_{st}$. Let $\sigma_{st}(v)$ be the number of shortest paths from $s$ to $t$ that some $v \in V$ lies on. The standard (shortest path) betweenness centrality of a node $v$ is defined as the sum of fractions of all shortest paths between each pair of vertices in a network that traverse a given node: $\sum_{s \in V, t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$. Brandes [2] and Newman [9] have shown that shortest paths from a single source $s$ to all other vertices can be efficiently aggregated by traversing the vertices in the order of a non-increasing distance from $s$. Efficient aggregation of shortest paths is used to compute the measure of all vertices in a network in $O(|V||E|)$ time. Below we show how to extend this definition to deal with not necessarily shortest paths, but rather $\Delta$-shortest paths which were introduced by Jiang et al. [10]. However, their proposed solution in [10] works only for walks and has exponential (in $\Delta$) runtime and, thus, cannot be used for real world application. Moreover, their solution in [10] cannot deal with the weighted version of $\Delta$-shortest paths.

The $\Delta$-shortest paths can exist in some types of real networks. For example, in communication networks, the routing protocols with the purpose of disseminating data traffic build the shortest path, but link fading, node loading, and load balancing problems cause the use of other alternative paths. In the real network the traffic could flow along the $\Delta$-shortest paths with non-uniform distribution for different values of $\Delta$. The traffic flow in the network may use 60% of the shortest paths, 20% of the 1-shortest paths, 15% of the 2- shortest paths and 5% of $k$-shortest paths, $k \geq 3$. In this case, in order to approximate the real flow in the network, we need to take into account this distribution. Another example is coming from the field of well-known reactive routing protocols (DSR, AODV): route discovery and route maintenance problems. After each link failure, the rediscovery process is necessary in order to establish alternate paths. The practical implementation of $\Delta$-shortest paths in this case is that a routing protocol may be a priori oriented not only by using the shortest paths, but also to find alternative $\Delta$-shortest paths. The main goal of this routing protocol is to provide an efficient and low cost routing for MANET. Using $\Delta$-shortest paths may reduce the number of active path failures, implicitly inducing the increasing path lifetime with minimal increase in the control traffic consumption.

II. PRELIMINARIES

Define a path $P(s,t)$ from $s \in V$ to $t \in V$ as an alternating sequence of edges $e = (i,j) \in E$ and distinct nodes $i \in V$, beginning with $s$ and ending with $t$, such that each edge connects it's preceding with its succeeding node. The length of a path in unweighted graph is the number of its edges. We use $d_{st}$ to denote the distance between nodes $s$ and $t$, i.e., the minimum length of any path connecting $s$ and $t$ in $G$. Define a shortest path between nodes $s$ and $t$ as any path of length equal to $d_{st}$. Let $N(v)$ denote the number of neighbors of some $v \in V$. Define the set of predecessors of a node $v$ on shortest paths from $s$ as $P_{s}(v) = \{ (i,j) : (i,j) \in E, d_{ij} = d_{st} + \omega_{ij} \}$, where $\omega_{ij}$ is a weight of edge $(j,v)$. Note, that in (our) case of unweighted graph we have $\omega_{ij} = 1$ and, thus,
Let $\sigma_{st}(v)$ be the number of shortest paths from $s$ to $t$ that some $v \in V$ lies on. The pair-dependency of a pair $s, t \in V$ on intermediate node $v \in V$ is defined as

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} = \left(1 + \delta_{st}(v, \bar{v})\right).$$

We denote by $d_{st}(\Delta) = d_{st} + \Delta$ the strong $\Delta$-shortest distance between nodes $s$ and $t$ in $G$ if there exists a path $P_{st}(\Delta) = (s, t)$ between $s$ and $t$ of length $l_{st} = d_{st} + \Delta$ for any positive integer $\Delta$. Such a path is called a strong $\Delta$-shortest path. The set of strong $\Delta$-shortest paths between nodes $s$ and $t$ is $P_{st}(\Delta) = \{P_{st}(\Delta, s, t), \ldots, P_{st}(\Delta, s, t)\}$ is denoted by $P_{st}(\Delta)(s, t)$. Let $\sigma_{st}(\Delta)$ be the number of strong $\Delta$-shortest paths from $s \in V$ to $t \in V$; therefore, $\sigma_{st}(\Delta) = \sum_{P_{st}(\Delta)}(1 + \delta_{st}(v, \bar{v}))$.

Let $d_{st}(\Delta)$ denote the number of strong $\Delta$-shortest paths from $s$ to $t$ that some $v \in V$ lies on. We define the strong $\Delta$-predecessors of node $v$ on $\Delta$-shortest paths from $s$ to $t$ from $P_{st}(\Delta)(s, t)$ between nodes $s$ and $t$ if there exists $k$ such that $0 \leq k < \Delta$. Let $\sigma_{st}(\Delta)$ be the number of strong $\Delta$-shortest paths from $s$ to $t$ that some $v \in V$ lies on. We define the strong $\Delta$-pair-dependency of a pair $s, t \in V$ on intermediate node $v \in V$ as

$$\delta_{st}(v) = \frac{\sigma_{st}(\Delta)}{\sigma_{st}} = \left(1 + \delta_{st}(v, \bar{v})\right)$$

i.e. the ratio of the number of strong $\Delta$-shortest paths from $s$ to $t$ that $v \in V$ lies on to the total number of strong $\Delta$-shortest paths from $s$ to $t$. The $\Delta$-shortest path $P_{st}(\Delta)(s, t)$ between nodes $s$ and $t$ is any path of length $l_{st}^{\Delta}$ not exceeding $d_{st}(\Delta), i.e.$ $l_{st}^{\Delta} \leq d_{st} + \Delta$. The set of $\Delta$-shortest paths between nodes $s$ and $t$ is $P_{st}(\Delta) = \{P_{st}(\Delta, s, t), \ldots, P_{st}(\Delta, s, t)\}$ is denoted by $P_{st}(\Delta)(s, t)$. Let $\sigma_{st}(\Delta)$ be the number of $\Delta$-shortest paths from $s \in V$ to $t \in V$; therefore, $\sigma_{st}(\Delta) = \sum_{P_{st}(\Delta)}(1 + \delta_{st}(v, \bar{v}))$. Let define the set of $\Delta$-predecessors of node $v$ on $\Delta$-shortest paths from $s$ as $P_{st}(\Delta)(v) = \{i \in V; \bar{v}, j \in E, d_{ij} + \Delta \leq d_{st}(\Delta) + \omega_{ij}, 0 \leq k \leq \Delta\}$ and the set of $\Delta$-successors of $v$ on $\Delta$-shortest paths from $s$ as $S_{st}(\Delta)(v) = \{j \in V; \bar{v}, i \in E, d_{ij} + \Delta \leq d_{st}(\Delta) + \omega_{ij}, 0 \leq k \leq \Delta\}$ where $\omega_{ij}$ is a weight of edge $(j, v)$. Let $\delta_{st}(\Delta)$ denote the number of $\Delta$-shortest paths from $s$ to $t$ that some $v \in V$ lies on; thus, $\delta_{st}(\Delta) = \sum_{v \in V} \sigma_{st}(\Delta)(v)$. Let define the $\Delta$-pair-dependency of a pair $s, t \in V$ on an intermediate node $v \in V$ as

$$\delta_{st}(v) = \frac{\sigma_{st}(\Delta)(v)}{\sigma_{st}(\Delta)} = \sum_{v \in V} \sigma_{st}(\Delta)(v)$$

i.e. the ratio of the number of $\Delta$-shortest paths from $s$ to $t$ that $v \in V$ lies on to the total number of $\Delta$-shortest paths from $s$ to $t$. The $(\Delta, m)$-shortest path from $s$ to $t$ that some $v \in V$ lies on is a $\Delta$-shortest path from $s$ to $t$, composed of combination of $(\Delta - m)$-shortest path from $s$ to $v$ and $\tilde{m}$-strong shortest path from $v$ to $t$, where $m = d_{st} - d_{sv} - d_{vt} + m$ for any positive integer $m$ such that $m \leq \Delta$. The length of the $\Delta$-shortest path from $s$ to $t$ in this case is bounded by $d_{st} - d_{sv} - d_{vt} + m$ since $l_{sv} + l_{vt}$ and $d_{st} + m \leq l_{sv} + d_{VT} + \Delta$. In addition, the length of a path from $v$ to $t$ is equal to $l_{st} = l_{st} - d_{sv} - d_{vt} + \tilde{m}$. Let $\sigma_{st}(\Delta,m)$ denote the number of $(\Delta, m)$-shortest paths from $s$ to $t$ for some particular $v \in V$ that lies on a $\Delta$-shortest path from $s$ to $t$. Let us define $\tilde{\sigma}_{st}(\bar{v}, \bar{v}) = \tilde{\sigma}_{st}(\bar{v}, \bar{v})/\delta_{st}(\bar{v})$ be the $(\Delta, m)$-pair-dependency score of a pair $s, t \in V$ on an intermediate node $v \in V$. To eliminate the need for explicit summation of all $\Delta$-pair-dependencies, we introduce the $(\Delta, m)$-dependency score of a node $v \in V$ on a single node $\tilde{m}$-shortest paths from $s$ to $v$ and $\tilde{m}$ strong shortest paths from $v$ to $t$. The DBC
measure is defined as:

\[
\hat{C}_B^{(A)}(v) = \sum_{s \in \Gamma(v)} \hat{\delta}_s^{(A)}(v) = \sum_{s \in \Gamma(v)} \frac{\Delta}{\sum_{s \in \Gamma(v)} \sigma_s^{(n)}}
\]

where \( \Delta \) is positive integer constant. Notice that we can add weight to each shortest path, thus, obtaining the weighted version of the problem.

III. FAST 1-BETWEENNESS CENTRALITY ALGORITHM IN UNWEIGHTED UNDIRECTED GRAPH

In this section, we describe fast algorithm for computing DBC in an unweighted undirected graph in the case when \( \Delta = 1 \). We start by performing a Breadth First Search (BFS) to determine the number of the shortest paths and strong 1-shortest paths to other nodes from source \( s \) (distance zero). For this purpose, for all nodes of \( \sum \) we execute two calculations 1-\( \hat{\delta}_s \) to \( \sigma_s^{(n)} \), for some current distance from \((1,0)\) and \((1,1)\)-pair-dependency of node \( s \) from other vertices having the same centrality for each node. Algorithm requires \((n+h)\) time and \(O(n+h)\) space, where \( n = |V| \) and \( h = |E| \).

A. Algorithm Analysis

We split the analysis according to the two major steps our algorithm consists of: top-down and bottom-up BFS with the corresponding calculations.

Step 1

In this step a number of shortest paths and strong \( \Delta \)-shortest paths to other nodes is computed. In addition, in this step we assign initial values of a dependency scores \( \hat{\delta}_s^{(A,0)}(i), \hat{\delta}_s^{(A,1)}(i) \) for step 2 as explained later.

Property 1. The number of strong \( \Delta \)-shortest-paths is \( \sigma_s^{(A)}(\Delta) = 0 \) for \( \Delta < 0 \).

Property 2. The number of strong \( \Delta \)-shortest-path from \( s \) to \( t \) that some \( v \in V \) lies on is \( \sigma_v^{(A)}(\Delta) \) for \( \Delta < 0 \).

Lemma 2 (Bellman-like criterion). A node \( v \in V \) lies on a \( \Delta \)-shortest path between nodes \( s, t \in V \) if and only if \( d_{sv} + d_{tv} - d_{st} \leq \Delta \).

Lemma 3 (Combinatorial strong \( \Delta \)-shortest-path counting for unweighted graph). The number of strong \( \Delta \)-shortest-paths

\[
d_s^{(\Delta)} = \sum_{j \in P^{(\Delta)}(s)} (d_{sj} - d_j^0) + \sum_{j \in P^{(\Delta)}(s)} (d_{sj} - d_j^1) + \sum_{j \in P^{(\Delta)}(s)} (d_{sj} - d_j^2)
\]

for \( s \neq v \in V \).

Following Lemma 3 above we can calculate the combinatorial number of strong \( \Delta \)-shortest paths for \( \Delta < 1 \). The number of strong \( \Delta \)-shortest paths for \( \Delta = 0 \) is

\[
\sigma_s^{(0)} = \sum_{j \in P^{(0)}(s)} (\sigma_j^{(0)} - 0) + \sum_{j \in P^{(0)}(s)} (\sigma_j^{(1)} - 0) + \sum_{j \in P^{(0)}(s)} (\sigma_j^{(2)} - 0)
\]

where \( \sigma_j^{(0)}(v) = 0 \) and \( \sigma_j^{(1)}(v) = \sigma_j^{(2)}(v) = \sigma_j^{(-1)} = \sigma_j^{(-2)} = 0 \) (by Properties 1, 2). Hence, the number of strong 0-shortest paths is \( \sigma_s^{(0)} = \sum_{j \in P^{(0)}(s)} \sigma_j^{(0)} \) as in Lemma 1.

The number of strong \( \Delta \)-shortest paths for \( \Delta = 1 \) is

\[
\sigma_s^{(1)} = \sum_{j \in P^{(1)}(s)} (\sigma_j^{(1)} - 0) + \sum_{j \in P^{(1)}(s)} (\sigma_j^{(0)} - 0) + \sum_{j \in P^{(1)}(s)} (\sigma_j^{(-1)} - 0)
\]

where in this case \( \sigma_j^{(1)}(v) = \sigma_j^{(0)}(v) = 0 \) and \( \sigma_j^{(-1)} = \sigma_j^{(-2)} = 0 \) (Properties 1, 2). Hence, the number of strong 1-shortest paths is

\[
\sigma_s^{(1)} = \sum_{j \in P^{(1)}(s)} \sigma_j^{(1)} + \sum_{j \in P^{(1)}(s)} \sigma_j^{(0)} + \sum_{j \in P^{(1)}(s)} \sigma_j^{(-1)}
\]

The Lemma 1 and equation (4) are used for calculations in Algorithm 1 at Step 1.

Step 2

Property 3. The \( \Delta \)-pair-dependency \( \hat{\delta}_{sv}^{(A)}(v) = \hat{\delta}_{sv}^{(A)}(v) / \sigma_{sv}^{(A)} = 1 \), for any target node \( v \in V \).

Property 4. The sum of \( \Delta \)-pair-dependency for any source node \( s \in E \)

\[
\sum_{v \in \Gamma(v)} \hat{\delta}_{sv}^{(A,0)}(v) = |V| 
\]

for any source node \( s \in V \).

Lemma 4. The \( (\Delta,0) \)-pair-dependency \( \hat{\delta}_{sv}^{(\Delta,0)}(v) = 1 \), for any target node \( v \in V \).

The result of Lemma 4 is used when we assign initial values of a dependency score of \( \hat{\delta}_{sv}^{(\Delta,0)}(v) = 1 \), and it is subtracted from values \( \hat{C}_B^{(A)}(i) \).
Lemma 5. The sum of all \((\Delta,0)\) pair-dependencies 
\(\delta_{s}^{(\Delta,0)}(v) = 1\) for any farthest node ("leaf") \(v \in V\).

The claim of Lemma 5 is used in Algorithm 1 Step 2 when we execute the "bottom-up" BFS starting at leaves towards node \(s\), to calculate the contribution (score) of the DBC by splitting/composition of \((\Delta, m)\)-pair-dependency scores over all nodes (Claims 1 and 2 below).

Claim 1. The splitting of \((\Delta, m)\)-pair-dependency score of node \(v\) on \((\Delta, m)\)-edge pair-dependency scores of predecessor's nodes \(l, p \in P_{s}^{(\Delta)}(v)\) calculated as following:

\[
\delta_{l}^{(\Delta,m)}(v) = \sum_{j \in l, p} \delta_{j}^{(\Delta,m)}(v, j) + \sum_{j \in l, t} \delta_{j}^{(\Delta,m)}(v, j) + \sum_{j \in p, t} \delta_{j}^{(\Delta,m)}(v, p).
\]

Claim 2. The composition of \((\Delta, m)\)-pair-dependency score of node \(v\) from edge \((\Delta, m)\)-pair-dependency scores of successor's nodes \(l, p \in S_{v}^{(\Delta)}(v)\) (i.e. node \(v\) is predecessor of one of the node \(l, p\) ) calculated as following:

\[
\delta_{l}^{(\Delta,m)}(v) = \sum_{j \in l, p} \delta_{j}^{(\Delta,m)}(v, j) + \sum_{j \in p, v} \delta_{j}^{(\Delta,m)}(v, j) + \sum_{j \in l, v} \delta_{j}^{(\Delta,m)}(v, p).
\]

Thus, the \(\Delta\)-betweenness centrality

\[
\ell_{b}^{(\Delta)}(v) = \sum_{s \in \mathbb{N}_{V}} \delta_{s}^{(\Delta)}(v) = \delta_{s}^{(\Delta)}(v) - \delta_{s}^{(\Delta)}(v) = \delta_{s}^{(\Delta)}(v) - 1 = \sum_{s = 0}^{\Delta} \delta_{s}^{(\Delta,m)}(v) - 1.
\]

Lemma 6. The \(\Delta\)-dependency score of \(s \in V\) on any \(v \in P_{s}^{(\Delta)}(p)\) and \(\forall m : 0 \leq m \leq \Delta\) obeys

\[
\delta_{s}^{(\Delta)}(v) = \begin{cases} 
\sum_{j \in l, p} \delta_{j}^{(\Delta)}(v, j) & \text{if } d_{w} = d_{p} - 1 \\
\sum_{j \in l, t} \delta_{j}^{(\Delta)}(v, j) & \text{if } d_{w} = d_{l} \\
\sum_{j \in p, t} \delta_{j}^{(\Delta)}(v, p) & \text{if } d_{w} = d_{p} + 1 \\
\sum_{j \in l, t} \delta_{j}^{(\Delta)}(v, j) + \sum_{j \in l, v} \delta_{j}^{(\Delta)}(v, j) + \sum_{j \in p, v} \delta_{j}^{(\Delta)}(v, p) & \text{if } d_{w} = d_{l} + 1
\end{cases}
\]

The \(\Delta\)-shortest path \(P_{s}^{(\Delta)}(v)\) from \(s \in V\) to \(t \in V\) that \(v \in V\) lies on exists and is equal to 
\(P_{s}^{(\Delta)}(v)\) \(\cup P_{v}^{(\Delta)}(v, t)\), if and only if there is a strong \(k\)-shortest path \(P_{s}^{(\Delta)}(v)\) and strong \(\tilde{k}\)-shortest path 
\(P_{v}^{(\Delta)}(v, t)\), with \(k = d_{s} + \Delta - (d_{w} + k)\) and \(\tilde{k}\) with the strong \(\Delta\)-shortest path criterion \(P_{s}^{(\Delta)}(v, t) = \{v\}\) (we call the last condition as path criterion). The length of the strong \(\Delta\)-shortest path \(P_{s}^{(\Delta)}(v)\) is equal to \(l_{v} = l_{w} + l_{t}\), and therefore \(d_{w} = d_{s} + \Delta = (d_{w} + k) + (d_{w} + \tilde{k})\). The Cartesian product of strong \(k\)-shortest path set \(P^{(k)}(v, t)\) and strong \(\tilde{k}\)-shortest path set \(P^{(\tilde{k})}(v, t)\) is defined by:

\[
P^{(k)}(v, t) \times P^{(\tilde{k})}(v, t) = \{p^{(k)}(v, t) \times p^{(\tilde{k})}(v, t) \mid p^{(k)}(v, t), p^{(\tilde{k})}(v, t) \in P^{(k)}(v, t) \cap P^{(\tilde{k})}(v, t)\}.
\]

The set of strong \(k\)-shortest paths \(P^{(k)}(v, t)\) between nodes \(s\) and \(v\) may contain \(k\)-shortest paths that \(\in V\) lies on, that do not satisfy the strong \(\Delta\)-shortest path criterion (i.e. \(P_{j}^{(k)}(v, t) \cap P_{j}^{(\tilde{k})}(v, t) \neq \{v\}\)). For \(P^{(\Delta)}(v)\) calculation, these paths are needed to be excluded from the result set. Let denote by \(Q^{(k, \tilde{k})}(v)\) the number of elements of set \(P^{(k)}(v, t) \times P^{(\tilde{k})}(v, t)\) that do not satisfy the path criterion above and need to be excluded from result set. Therefore, \(\sigma^{(\Delta)}_{s}(v)\), for any strong \(\Delta\)-shortest path, is

\[
\sigma^{(\Delta)}_{s}(v) = \sum_{k = 0}^{\Delta} (n \times P^{(k)}(v, t) \times P^{(\tilde{k})}(v, t) - Q^{(k, \tilde{k})}(v)) = \sum_{k = 0}^{\Delta} M^{(k)}(1)
\]

Theorem 2. The \(\Delta\)-pair-dependency is \(\delta_{s}^{(\Delta)}(v) = \sum_{m=0}^{\Delta} \delta_{s}^{(\Delta,m)}(v)\)

Theorem 3. The fast DBC (for \(\Delta = 1\)) algorithm in unweighted undirected graph work correctly and can be computed in \(O(n \times h)\) time and \(O(n \times h)\) space.

Proof: Follows from Theorem 2, Claim 1, and Lemma 2.

IV. SIMULATIONS

We conducted extensive simulations to study the performance of \(\Delta\)-metrics introduced in this paper for graphs with different characteristics. The graph is modeled by a connected unit disk graph with limited graph diameter. We simulate \(X\) experiments for each case and the average values for our characteristics have been calculated. The graph \(\Delta\)-metric analyzed for the five different cases of \(\Delta\)’s values. Figures 1-3 summarize average graph \(\Delta\)-metric performance for graphs with different number of nodes \(|V| = n\). The values of different number of nodes in our experiments randomly placed on square with same neighborhood range are correlated with mean graph degree metric \(\bar{d} = \sum_{i=1}^{n} d_{i} / n\) and equals 1)

\[
n = 5(\bar{d} = 5.5) \quad 2) n = 10(\bar{d} = 8.1) \quad 3) n = 15(\bar{d} = 10.5) \quad 4) n = 20(\bar{d} = 13.2) \quad 5) n = 250(\bar{d} = 16.1).
\]

Figure 1(a) plots the average number of strong \(\Delta\)-shortest paths between all pairs of nodes versus total number of
nodes in graph for the five different $\Delta$'s values. We can see that the number of strong $\Delta$-shortest paths $\sum_{v\in V} \sigma^\Delta_{st}(v)$ have increased when the number of the nodes in graph is increased. Also, the number of strong $\Delta$-shortest paths for different $\Delta$'s values with the same number of nodes is different.

Figure 1. The plots of the average number of strong $\Delta$-shortest paths between all pairs of nodes versus the number of nodes in graph (a) and the plots of the overlapping percent between subset of 10% nodes with high DBC ranking from ranking list when $\Delta = 0$ and same nodes in ranking lists for $1 \leq \Delta \leq 4$ (b).

Figure 1(b) plots the overlapping percentage between the subset of nodes with high DBC ranking in ranking list when $\Delta = 0$ and same nodes in ranking lists for $1 \leq \Delta \leq 4$ versus number of nodes in graph. The size of the subset in this experiment is 10% of the total number of nodes.

Figure 2 (a) plots the distance between the node with maximal betweenness centrality measure ($\Delta = 0$) and node with maximal $1 \leq \Delta \leq 4$ DBC versus the total number of nodes in same graph for the five different $\Delta$'s values. After computing $\Delta$-betweenness centrality measure, each node $v$ in graph $G$ has different DBC value. Therefore we can rank the nodes of the graph by DBC values and find node with maximal DBC value (rank=1) in graph. Note, that for the different $\Delta$ values, we have different DBC ranking list. Since, DBC values with different $\Delta$'s are used as a relative measure of how nodes differ from each other, the induced ranking is more important than magnitude of the DBC values.

Figure 2 (b) plots the rank of node $v$ with maximal DBC (rank=1) when $\Delta = 0$ and it ranks for $1 \leq \Delta \leq 4$ DBC in same graph versus number of nodes in graph for the five different $\Delta$'s values. The node's rankings produced by different DBC are not identical. For example, in the graph with $n = 200$ nodes, the node $v$ with DBC rank=1 in case $\Delta = 0$ has rank=9 when $\Delta = 4$. As we can learn from Figure 2(a) the distance between the node with maximal betweenness centrality measure ($\Delta = 0$) and the node with maximal $1 \leq \Delta \leq 4$ DBC is changed when the number of nodes is increased. We also show the change in the distance between the nodes of centrality measures for the different $\Delta$'s values versus the mean degree (Figure 3 (a)) and the diameter of the graph (Figure 3 (b)). Our simulations show that there is a potential importance of the nodes with high $\Delta$-betweenness centrality measure.

Figure 2. The plots of the distance between the node with maximal $\Delta = 0$ BC and the node with maximal $1 \leq \Delta \leq 4$ DBC versus the number of nodes in same graph (a) and the plots the ranks of node $v$ with maximal DBC (rank=1) when $\Delta = 0$ versus the number of nodes (b).

Figure 3. The plots of the distance between node with maximal $\Delta = 0$ BC and node with maximal $1 \leq \Delta \leq 4$ DBC against mean of graph degree (a) and graph diameter (b) in graph with constant number of nodes $n = 100$.

values and this should be investigated further. It would be interesting to generalize the results for weighted graphs and improve the runtime for the large values of $\Delta$.

REFERENCES