International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

Sparse Networks-Based Speedup Technique for Proteins Betweenness Centrality Computation

Razvan Bocu, *University College Cork,* Dr Sabin Tabirca, *University College Cork*

Abstract—The study of proteomics reached unexpected levels of interest, as a direct consequence of its discovered influence over some complex biological phenomena, such as problematic diseases like cancer. This paper presents the latest authors' achievements regarding the analysis of the networks of proteins (interactome networks), by computing more efficiently the betweenness centrality measure. The paper introduces the concept of betweenness centrality, and then describes how betweenness computation can help the interactome network analysis. Current sequential implementations for the betweenness computation do not perform satisfactory in terms of execution times. The paper's main contribution is centered towards introducing a speedup technique for the betweenness computation, based on modified shortest path algorithms for sparse graphs. Three optimized generic algorithms for betweenness computation are described and implemented, and their performance tested against real biological data, which is part of the IntAct dataset.

Keywords—Betweenness centrality, interactome networks, protein-protein interactions, sub-communities, sparse networks, speedup technique, IntAct.

I. Introduction

A. Interactome networks and their importance

THE concept of interactome networks represents a very important biological construct. It is widely used to describe the protein interactions that determine the organization and function of a biological organism. These networks feature a complex structure that makes any research endeavour to be complex. Nevertheless, a proper understanding of the general structure of the protein interactions is necessary, as they consistently influence the function of a biological organism as a whole, from the simplest to the most complex ones. Therefore, it is mandatory to discover more efficient techniques that can be applied to the study of the structure and properties of the interactome networks.

Betweenness centrality is one of the centrality measures that allows for the interactome networks to be properly analyzed, because it essentially allows for various functional protein clusters to be determined with a high degree of accuracy. A classical betweenness algorithm: the Brandes algorithm, which computes the betweenness centrality of the nodes (proteins). The Brandes algorithm [13] proves to be efficient enough in practice, featuring a complexity of $O(nm + n^2 \log n)$, where n is the number of vertices and m is the number of edges. It normally processes a network with thousands of nodes and tens of thousands of edges in a few hours. The Newman-Girvan algorithm [1, 6] is another classical construct dedicated to computing the betweenness centrality for edges. Although it is an important algorithmic construct, it still does not perform as expected in terms of execution times, as it processes a network with thousands of nodes and tens of thousands of edges in more than ten hours on a standard Intel Pentium Dual Core machine. One of the authors' previous papers proposes a new algorithm that is able to optimize the betweenness computation for a network featured by thousands of nodes and tens of thousands of edges. This approach is based on the Dijkstra's algorithm and reduces the computation time for a network with thousands of nodes by up to 90% in the worst case. This paper extends further the analysis regarding the optimization of betweenness computation algorithms. The additional gain in performance is significant and the underlying mechanism is worth to be explained thoroughly.

Razvan Bocu is a PhD Researcher and demonstrator in the Department of Computer Science, University College Cork, email: razvan.bocu@cs.ucc.ie
Dr Sabin Tabirca is a researcher and Senior Lecturer in the Department of Computer Science, University College Cork

The optimization technique takes into account an essential experimental observation of the authors and of some other preceding authors that are acknowledging the sparse nature of the interactome networks. Therefore, classical shortest path and betweenness algorithms perform sub-optimally when applied to these large-scale biological networks. This remark led us to the creation of three novel betweenness computation algorithms that are based on three shortest path algorithms, which optimize the computation process in the case of large sparse networks.

The paper will conduct, at first, a brief literature review on the existing relevant works on betweenness and shortest path computation. The second section will expand on the theoretical aspects related to our research. Thus, it will introduce the concepts of graph and betweenness. Moreover, the measures that are used by any betweenness computation algorithm are introduced and explained. The second section will also review the classical sequential solutions usually used to compute the betweenness centrality measure. We shall move on then and introduce the concept of sparse networks and explain why they are suitable for optimizing the betweenness centrality computation for large-scale biological networks. The third section will describe in detail the sparse networks-based optimization technique, ehich is based on three modified shortest path algorithms. The fourth section will present the testing procedure that will prove the effectiveness of the optimization technique. The tests are run on real biological data that is part of the IntAct dataset [15]. The performance gain is analyzed in depth and the suitability of the optimization technique fully assessed.

B. Relevant existing works

The research workflow that produced the results that are presented in this paper is based on valuable results that are the consequences of a thorough and extensive research activity. Therefore, this subsection will enumerate and succinctly describe the main existing research works, which contributed to the advances proposed in this paper.

Although the scientific literature related to betweenness is not excessively extensive, there are enough papers and research projects that are worth to be mentioned. Among these, we shall select the ones that had a decisive influence on our research pathway.

One of the first extensive works on betweenness belongs to Newman and Girvan. The Newman-Girvan algorithm is one of the methods used to detect communities in complex systems. The notion of a ”community structure” is related to that of clustering, though it isn’t quite the same. A community consists of a subset of nodes within which the node-node connections are dense, and the edges to nodes in other communities are less dense. There are numerous alternative method for detecting communities in networks. These include hierarchical clustering, partitioning graphs to maximize quality functions such as network modularity, k-clique percolation, etc [2]. Nevertheless, we preferred to make use of the Newman and Girvan conceptual system due to its structural articulation and practical usage in many situations. The Newman-Girvan algorithm is particularly used to compute betweenness for edges (links) that connects the nodes (proteins) in a network.

The Brandes algorithm proves that betweenness can be computed exactly even for fairly large networks. It introduces more efficient algorithms based on a new accumulation technique that integrates well with traversal algorithms solving the single-source shortest-paths problem, and thus exploiting the sparsity of typical instances. The range of networks for which betweenness centrality can be computed is thereby extended significantly [13]. Moreover, it turns out that all standard centrality indices based on shortest paths can thus be evaluated simultaneously, further reducing both the time and space requirements of comparative analyses.

It has a significantly improved structure, which makes it run faster and improves its general readability and usability. As a natural consequence, the algorithm is able to enlarge the O(n3) bottleneck and requires O(nm+n2logn) to execute. This is a major improvement, which can prove very important for a high-scale network, featured by thousands of nodes. Before the Brandes algorithm was released, the analysis of a large network featured by thousands of nodes and tens of thousands of edges was prohibitive. The Brandes algorithm scales sensibly better than any previous implementation of an algorithm that computes the betweenness centrality measure. As an example, processing networks with thousands of nodes was previously a challenging task, which is made an accessible one using the new Brandes algorithm.

The idea of betweenness is tightly related to the idea of shortest path computation, as we shall see in the next section. As a consequence, it is very important to compute the shortest paths in the analyzed network as efficiently as possible. Following a series of theoretical and experimental activities carried on interactome networks, it was concluded that interactome networks feature a sparse nature, and we’ll also expand on this in the following section. Therefore, it is essential for an efficient sequential betweenness algorithm applied on interactome networks, to use a shortest path algorithm that is designed to optimize computations on sparse networks. Essentially, we created three novel betweenness computation algorithms that make use of three shortest path algorithms that are specially designed for sparse networks, the Wagner algorithm, the Ramalingam-Reps algorithm and the Pettie algorithm. The following sections will describe these three algorithms in more detail.

II. Theoretical background

A. Basic theoretical concepts

In the most common sense of the term, a graph is an ordered pair G=(V,E), comprising a set V of vertices or nodes together with a set E of edges, which are two-element subsets of V. To avoid ambiguity, this type of graph may be described precisely as undirected and simple. Using the terminology peculiar to interactome networks, proteins are modeled as vertices and the biological links as edges.

Within graph theory and network analysis, there are various measures of the centrality of a vertex within a graph that determine the relative importance of the vertex within the graph. For example, applied to the social networks study, centrality may offer an accurate measure of how important is a person in a certain network. Moreover, centrality is an essential concept for other types of networks, such as biological networks or interactome networks. In this particular case, the centrality may measure the importance of a certain protein in the network, or the relative importance of a certain sub-community (group of proteins) in the network. In the theory of space syntax, centrality specifies how important a room is within a building or how well-used a road is within an urban network. Basically, there are four measures of centrality that are widely used in the network analysis: degree centrality, betweenness, closeness, and eigenvector centrality [1].

Betweenness is a centrality measure based on shortest paths, widely used in complex network analysis. One of the fundamental problems in network analysis is to determine the importance (or the centrality) of a particular vertex (or an edge) in a network. Some of the well-known metrics for computing centrality are closeness, stress and betweenness. Of these indices, betweenness has been extensively used in recent years for the analysis of social interaction networks, as well as other large-scale complex networks. Some applications include lethality in biological networks, study of sexual networks and AIDS, identifying key actors in terrorist networks, organizational behavior, and supply chain management processes. Betweenness is also used as the primary routine in popular algorithms for clustering and community identification in real-world networks. For instance, the Girvan-Newman algorithm iteratively partitions a network by identifying edges with high betweenness scores, removing them and re-computing centrality scores.

Betweenness centrality can be computed both for nodes and for edges. The computation technique is exactly the same both for nodes and for edges, as it involves the computation of the distance matrix for a certain node or edge. Therefore, we shall briefly describe the betweenness centrality for nodes (vertices), which is a centrality measure of a vertex within a graph. Vertices that occur on many shortest paths between other vertices have a higher betweenness than those that do not. For a graph G=(V,E) with n vertices, the betweenness $C_B(v)$ of the vertex v is given by the following formula:

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

where σ_{st} is the number of geodesic shortest paths from vertex s to vertex t, and $\sigma_{st}(v)$ is the number of shortest geodesic paths from vertex s to vertex t that pass through a vertex v. This may be normalized by diving through the number of pairs of vertices excluding v, which is $(n-1)(n-2)$.

B. Review of the classical betweenness solutions

The Brandes algorithm proves that betweenness can be computed exactly even for fairly large networks. It introduces more efficient algorithms based on an accumulation technique that integrates well with traversal algorithms solving the single-source shortest-paths problem, and thus exploiting the sparsity of typical instances. The range of networks for which betweenness centrality can be computed is thereby extended significantly [13]. Moreover, it turns out that all standard centrality indices based on shortest paths can thus be evaluated simultaneously, further reducing both the time and space requirements of comparative analyses.

It has a significantly improved structure, which makes it run faster and improves its general readability and usability. As a natural consequence, the algorithm is able to enlarge the $O(n^3)$ bottleneck and requires $O(nm + n^2 logn)$ to execute. This is a major improvement, which can prove very important for a high-scale network, featured by thousands of nodes. Before the Brandes algorithm was released, the analysis of a large network featured by thousands of nodes and tens of thousands of edges was prohibitive. The Brandes algorithm scales sensibly better than any previous implementation of an algorithm that computes the betweenness centrality measure. As an example, processing networks with thousands of nodes was previously a challenging task, which is made an accessible one using the Brandes algorithm.

The algorithm introduces the concept of the dependency of a vertex $s \in V$ on a single vertex $v \in V$, defined as:

$$\delta_{s\bullet}(v) = \sum_{t \in V} (\delta_{st}(v))$$

The crucial observation is that these partial sums obey a recursive relation. Following, we'll present some important relations, without demonstration, as this is beyond the scope of this paper. If there is exactly one shortest path from $s \in V$ to each $t \in V$, the dependency of s on any $v \in V$ obeys [13]:

$$\delta_{s\bullet}(v) = \sum_{w: v \in P_s(w)} (1 + \delta_{s\bullet}(w))$$

The dependency of $s \in V$ on any $v \in V$ obeys:

$$\delta_{s\bullet}(v) = \sum_{w: v \in P_s(w)} \left(\frac{\sigma_{sv}}{\sigma_{sw}} \cdot (1 + \delta_{s\bullet}(w))\right)$$

Betweenness centrality can be computed in $O(nm + n^2 \log n)$ time and $O(n+m)$ space for weighted graphs. For un-weighted graphs, the running time reduces to $O(nm)$ [13].

The practical relevance of the asymptotic complexity improvement achieved by accumulating dependencies was evaluated. Also, weighted and un-weighted versions of this algorithm for directed and undirected graphs using various datasets in order to test each implementation, were implemented. Thus, we determined that a Brandes-based betweenness computation application performs well in terms of processing times for networks featured by up to 4500 nodes.

To sidestep the shortcomings of the hierarchical clustering method, Newman and Girvan propose an alternative approach to the detection of communities. Instead of trying to construct a measure that tells us which edges are most central to communities, we focus instead on those edges that are least central, the edges that are most *between* communities. Rather than constructing communities by adding the strongest edges to an initially empty vertex set, we construct them by progressively removing edges from the original graph. Vertex betweenness has been studied in the past as a measure of the centrality and influence of nodes in networks. First proposed by Freeman, the betweenness centrality of a vertex i is defined as the number of shortest paths between pairs of other vertices that run through i. It is a measure of the influence of a node over the flow of information between other nodes, especially in cases where information flow over a network primarily follows the shortest available path.

To find which edges in a network are most between other pairs of vertices, they generalize Freeman's betweenness centrality to edges and define the edge betweenness of an edge as the number of shortest paths between pairs of vertices that run along it. If there is more than one shortest path between a pair of vertices, each path is given equal weight such that the total weight of all of the paths is unity. In the case a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go through one of these few edges. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, we separate groups from one another and so reveal the underlying community structure of the graph. The algorithm they propose for identifying communities is simply stated as follows, specifying only the main steps, the pseudo-code will follow:

- Calculate the betweenness for all edges in the network
- Remove the edge with the highest betweenness
- Recalculate betweennesses for all edges affected by the removal
- Repeat from step 2 until no edges remain

As a practical matter, we calculated the betweennesses by using the algorithm of Newman and Girvan, which calculates betweenness for all m edges in a graph of n vertices in time $O(mn)$. Because this calculation has to be repeated once for the removal of each edge, the entire algorithm runs in worst-case time $O(m^2n)$, which ensures a decent computation in terms of execution times for networks of proteins featured by up to 4000 nodes. However, after the removal of each edge, we only have to recalculate the betweennesses of those edges that were affected by the removal, which are at most only those in the same component as the removed edge. This means that running time may be better than worst-case for networks with strong community structure (those that rapidly break up into separate components after the first few iterations of the algorithm). To try to reduce the running time of the algorithm further, one might be tempted to calculate the betweennesses of all edges only once and then remove them in order of decreasing betweenness. We find, however, that this strategy does not work well, because if two communities are connected by more than one edge, then there is no guarantee that all of those edges will have high betweenness-we only know that at least one of them will. By recalculating betweennesses after the removal of each edge we ensure that at least one of the remaining edges between two communities will always have a high value.

C. Interactome networks and sparse graphs

The category of sparse networks includes many remarkable networks that are extensively used in order to model various phenomena and systems that are essential in relation to different layers of life. Essentially, at an informal level, we can state that a network is sparse when it is featured by a sensibly smaller number of edges than the theoretical number of possible edges. The following definition explains precisely when it can be stated that a network is featured by a sensibly smaller number of edges [15].

Definition 1. A sparse graph is a graph $G = (V, E)$ for which the following relation is true: $|E| = O(|V|)$.

For example, let us consider a graph $G = (V, E)$ that has n nodes. Let us suppose that the out degree of each vertex in G is equal to a certain constant k. The graph G is a sparse one because $|E| = k|V| = O(|V|)$. Although the research activities related to sparse networks do not constitute an impressively extended array, there is a general agreement on this formal definition of a sparse network. Nevertheless, the informal definition is still enough for a good understanding of the main property of a sparse network and the formal definition is provided for the sake of the scientific presentation's clarity.

The sparse nature of the interactome networks has long been ignored or just scarcely exploited in the most favourable case. At the same time, none of the existing researches tries to optimize the betweenness centrality computation making use of sparse networks shortest path algorithms. In fact, none of the betweenness research reports this paper makes use of utilizes the idea of sparse networks.

The IntAct protein database currently contains 56.191 discovered proteins and 188.292 binary interactions. The number of the theoretically possible number of interactions is given by the computation of A^2_{56191}, because a protein network is a directed one, as all proteins can exercise a certain influence on each other. This reasoning directs us to the conclusion that the most up to date interactome network could feature a maximum number of $56190 \cdot 56191$ biological links. Therefore, this interactome network could have a maximum of 3.157.372.190 biological links. Following our informal definition, we can say that the actual interactome network has a sensibly smaller number of nodes than the maximum theoretically possible one. Also, the same conclusion can be easily inferred using the formal Definition 1. In the case of the existing interactome network, $k \approx 3$, $E = 188.292$ and $V = 56.191$. Therefore, $|E| \approx k \cdot |V| = O(|V|)$, and hence the conclusion.

Corollary 1. The interactome of any living organism constitutes a network that is sparse.

The reasoning above proves that interactome networks are sparse and, as a consequence, any betweenness computation algorithm should be specially designed to perform optimally on these remarkable networks.

The last paragraph will further enforce the idea that existing protein databases suggest interactome networks are sparse. Apart from IntAct, some other research groups maintain similar databases that contain proteomic data. Database of Interacting Proteins (DIP) [17] contains information about 20728 proteins and 57684 biological links established between them. Also, the Human Protein Reference Database [18] holds information about 25661 proteins and 38167 biological links. Although it can be noticed there are quantitative differences among various biological databases, the order of measure remains always the same. As a consequence, this empirical information comes to further sustain the idea of sparsity in relation to interactome networks.

III. Betweenness computation in sparse graphs

A. Shortest path algorithms and betweenness

Betweenness centrality relies on the shortest paths computation. A directed graph *G=(V,E)* consists of a set V of vertices (proteins for our purposes) and a set $E \subseteq V \times V$ of directed edges (physiological links, in our case). For this paper's purposes, undirected graphs are equivalently replaced by symmetric directed graphs containing two oppositely directed edges for each undirected edge. Unless explicitly stated otherwise, we shall therefore refer to graphs, edges, when we mean directed graphs, directed edges, etc. Also, it may be safely assumed that there are no loops, i.e. edges connecting a vertex with itself, because they have no influence on betweenness. A path from a source s from V to a target t from V is an alternating sequence of vertices and edges, starting with s and ending with t. The length of a path (s, t) is the number of edges it contains and the distance from s to t is defined as the minimum length of all the paths connecting s to t. Note that $s = t$ implies that $dist(s, t) = 0$.

Denote by $\sigma(s, t)$ the number of shortest (s,t)-paths, and let $\sigma_{st}(v)$ be the number of shortest (s,t)-paths passing through some vertex v, other than s,t. If s=t, then $\sigma_{st}(v) = 1$ and, if

$v \in (s,t)$ then let $\sigma_{st}(v) = 0$. Then, the betweenness $C_B(v)$ of a vertex v from V is defined to be:

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

where $\frac{0}{0} = 0$ by convention. The measure is therefore usually interpreted as the degree to which a vertex has control over pair-wise connections between other vertices, based on the assumption that the importance of connections is equally divided among all shortest paths for each pair. As pointed out already in Freeman [20], the definition of betweenness applies to disconnected graphs without modification. Though the distance between two vertices not connected by a directed path is undefined, this also means that the number of shortest paths between them is zero, so that the resulting zero contribution is exactly what is desired. The betweenness computation algorithms we implemented are based on this strategy.

It doesn't matter what is the shortest paths algorithm being used to compute the betweenness centrality measure, the general structure of the algorithm is described in *Figure 1*. We note that by finding all-pairs shortest paths using Breadth-First Search (BFS) starting from each vertex in the graph, the edge betweenness value can be obtained by summing pair-dependencies [13] over all the traversals. The pair-dependency is defined as $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$, where σ_{st} denotes the number of shortest paths from $s \in V$ to $t \in V$ and $\sigma_{st}(v)$ is the number of shortest paths from s to t that go through v. Pair-dependencies calculated from each BFS for every vertex in the graph are additive. Summations from all traversals will give us the overall vertex betweenness, from which edge betweenness can be obtained by a trivial generalization. Since BFS can be performed independently and simultaneously from each vertex in the graph, the calculation required at each iteration for finding the edge with the highest betweenness value can be done by parallelizing all-pairs shortest paths or by simply utilizing a suitable sequential algorithm.

The pseudo code presented in Figure 1 suggests that the main step of a typical betweenness computation algorithm is represented by the computation of the distance matrix associated to each particular network being processed. Once the distance matrix is computed, the betweenness value for each node can be computed after determining $\sigma_{uv}(node)$ and σ_{uv}. The former is calculated by making use of the Bellman's lemma, which states that a vertex v belongs to a shortest path between two other vertices x and y if and only if $d(x,y) = d(x,v) + d(v,y)$. Here, d represents the distance matrix. In order to compute σ_{uv}, we make use of the algebraic counting technique. Therefore, given the distance p of the shortest path between a pair of vertices, we just need to raise the adjacency matrix to the power p. Thus, the value of any element in the resulting matrix represents exactly the number of the shortest paths between those particular vertices.

We used three shortest path algorithms in order to optimize our betweenness computation techniques, the Pettie algorithm, the Ramalingam-Reps algorithm and the Wagner algorithm. They all compute, as it would be expected from a dedicated shortest path algorithm, the distance matrix, but strictly more efficient than Johnson's algorithm does. The Johnson's algorithm has long been thought of as the most efficient shortest path algorithm for sparse graphs. Our research involved an extensive comparative testing of the Johnson's algorithm and, hence, we can acknowledge its efficient behaviour in practice. Nevertheless, it is outperformed by all three algorithms that were used for optimizing the betweenness computation. Essentially, the novel betweenness computation technique respects the following general structure, regardless of the shortest path algorithm (Pettie, Ramalingam-Reps, Wagner) being used.

```
Input: A list of N vertices
Output: A 2xN matrix containing vertices and their betweennesses
1.  Call sparseShortestPath(D:distance_matrix)
2.  Call shortestPathsNumber(D:distance_matrix)
3.  for_each node in 0,…,N-1 do
4.    CB(node)=0
5.  end_for_each
6.  for_each node do
7.    for_each pair (u,v) of nodes do
8.      CB(node)= $\sum_{u \neq node \neq v} \frac{\sigma_{uv}(node)}{\sigma_{uv}}$
9.    end_for_each
10. Store node and CB(node) in matrix
11. end_for_each
12. return matrix
13. end_function
```

Fig. 1. The faster betweenness calculation technique

The essential part of the algorithm is constituted by the call of the method that performs the computation of the distance matrix in line 1. It is also the section of the code that determines the overall complexity of the algorithm. The Pettie-based version of our optimization technique is the most resource-consuming one, with a complexity of $O(mn + n^2 \log\log n)$, which is strictly faster than Johnson's and Dijkstra's algorithms. Hence, the optimized betweenness computation scheme has a square order complexity in the worst case that makes it more efficient than any other previous betweenness computation algorithm applied on large sparse graphs.

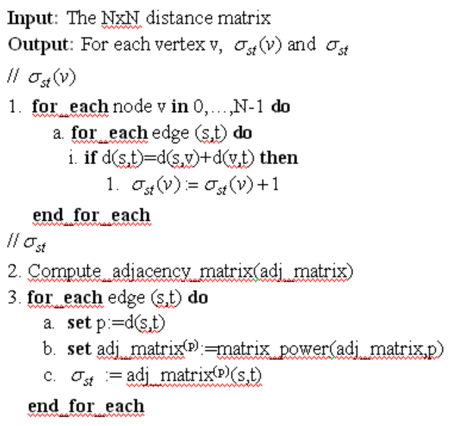
The values that the distance matrix contains after the subroutine called in line 1 is executed, are essential for the computation of $\sigma_{st}(v)$ and σ_{st}. The pseudo-code in *Figure 2* summarizes the computation of these parameters.

B. Remarks on a sparse shortest path algorithm

The Wagner algorithm is one of the theoretical constructs used to create a flavour of the optimization technique. Although it didn't produce the best execution times, it will be briefly described because it features a clear and easy to understand structure.

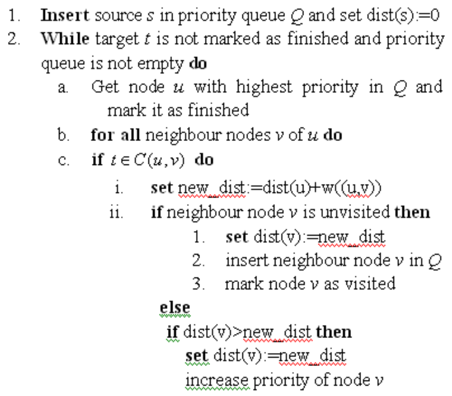
The algorithm is based on a modified version of Dijkstra's algorithm. The goal of Dijkstra's algorithm with pruning is to decrease the number of visited nodes, the search space, by visiting only a subset of the neighbours. The algorithm is outlined in *Figure 3*.

The execution of the algorithm generates in an efficient manner the distance matrix of the analyzed interactome network. Considering the algorithm described in *Figure 2*, the



```
Input: The NxN distance matrix
Output: For each vertex v, $\sigma_{st}(v)$ and $\sigma_{st}$
// $\sigma_{st}(v)$
1. for_each node v in 0,…,N-1 do
    a. for_each edge (s,t) do
        i. if d(s,t)=d(s,v)+d(v,t) then
            1. $\sigma_{st}(v) := \sigma_{st}(v)+1$
   end_for_each
// $\sigma_{st}$
2. Compute_adjacency_matrix(adj_matrix)
3. for_each edge (s,t) do
    a. set p:=d(s,t)
    b. set adj_matrix$^{(p)}$:=matrix_power(adj_matrix,p)
    c. $\sigma_{st}$ := adj_matrix$^{(p)}$(s,t)
   end_for_each
```

Fig. 2. Computation of $\sigma_{st}(v)$ and σ_{st}



```
1. Insert source $s$ in priority queue $Q$ and set dist(s):=0
2. While target $t$ is not marked as finished and priority
   queue is not empty do
    a. Get node $u$ with highest priority in $Q$ and
       mark it as finished
    b. for all neighbour nodes $v$ of $u$ do
    c. if $t \in C(u,v)$ do
        i. set new_dist:=dist(u)+w((u,v))
        ii. if neighbour node $v$ is unvisited then
            1. set dist(v):=new_dist
            2. insert neighbour node $v$ in $Q$
            3. mark node $v$ as visited
           else
            if dist(v)>new_dist then
              set dist(v):=new_dist
              increase priority of node $v$
```

Fig. 3. The Wagner shortest path algorithm

subroutine is called in line 1. Thus, the overall betweenness computation time is significantly reduced.

The previous stage of the research [19] proved the performance of the Brandes algorithm can be improved using a betweenness computation algorithm that is based on a Dijkstra-generic computation scheme. This paper describes a further and significant improvement of the previous computation scheme. The reasoning that generated the idea of the optimization is based on the very important empirical remark that interactome networks are sparse. Therefore, three novel betweenness computation algorithms for sparse networks were created. In the next section, the behaviour of the optimized computation scheme will be assessed on some real biological data.

IV. Evaluation of the novel scheme

A. Description of the datasets

We have extensively tested the suitability of the novel betweenness computation scheme. The optimization technique was implemented using the C language and the gcc compiler under CentOS. The machine is a dual processor Opteron clocked at 2.5 GHz with 4 GB of RAM. Also, portions of the Leonardo library were used [16]. We made use of real biological data that is part of the IntAct database [15]. Currently, IntAct holds information about 56.191 proteins and 188.292 biological links. In order to test our computation technique, we extracted seven datasets from the IntAct database. All these subsets consist of functionally-related clusters of proteins. In order to create the functionally-related subsets, we used a partitioning algorithm that we created, which is based on the Newman-Girvan edge betweenness computation algorithm. The description of this algorithm will not be provided, as it is beyond the scope of this paper. We used the same subsets that provided the data for the testing process that finalized the previous stage of our research, in order to ensure a perfect comparative analysis of the efficiency of the two methods. Essentially, the datasets feature the following number of proteins and biological links: 1000 (1207), 2000 (4823), 2500 (5692), 3000 (6209), 7000 (14175), 8000 (21304), 9000 (35892). All datasets define interactome networks that obey both the informal and formal definition of sparse networks that we provided in Section II, C and, consequently, they are sparse networks.

B. The execution times

Prior to commenting on the new computation scheme's performance, we shall present the execution times that we obtained after running the three algorithms on the above described datasets. The results are summarized in *Table I* and *Table II*, which are shown on the next page. The results are given in milliseconds.

The first table contains execution times resulted after all the three flavours of the novel betweenness computation technique were run on all seven datasets. We recall that the shortest path algorithms that we used belong to Ramalingam-Reps, Pettie and Wagner. The best performer proved to be the flavour of the optimization technique that is based on the Ramalingam-Reps shortest path algorithm. Nevertheless, all three flavours perform very well in terms of execution times. Practically, we managed to improve our own computation technique that is based on the Dijkstra-generic algorithm which, at its turn, improved the Brandes' algorithm performance. Compared to the previous version of the betweenness computation technique [19], it can be noticed a performance gain of approximately 70% in the case of the largest dataset used. Considering the fact that no interactome database is ever processed exhaustively, the performances of this novel sequential computation technique make us conclude that it can be used to analyze the data recorded in any medium to large biological dataset. This is an important achievement, as any existing sequential betweenness computation algorithm is only suitable for analyzing small to medium-sized datasets. The graph in *Figure 4* is generated using the already presented experimental data, and it further enhances the idea of geometric speedup that the new betweenness computation technique provides.

The graph shows in a suggestive manner how the new betweenness optimization technique considered in all three flavours (the blue, pink and yellow lines) behaves in practice. The Brandes based computation performs well only for relatively small datasets. The previous optimization technique, which is based on the Dijkstra shortest path algorithm (the

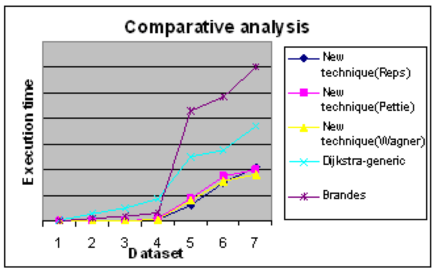


Fig. 4. The geometric speedup of the new computation technique

TABLE I
COMPARATIVE VIEW OF THE EXECUTION TIMES (1)

Test no.	Reps-based	Pettie-based	Wagner-based
1	1410	2010	1890
2	11770	13030	12290
3	22580	28060	25930
4	137250	183740	173960
5	1235520	1734970	1604260
6	2974100	3473260	3101420
7	4223710	3984260	3663380

cyan line) [19] brings an important speedup for the betweenness computation and the growth of the execution time becomes sensibly less abrupt. Nevertheless, the new technique proves its superior efficiency in all its three flavours (the blue, pink and yellow lines) and the execution time growth pattern suggests that it can be successfully used to process fairly large biological datasets. Up to now, a similar speedup on interactome networks has been achieved only by using parallel computation methods.

The already gathered experimental information can be summed up into the following conclusions:

- The new betweenness computation scheme improves on our previous computation technique for interactome networks [19] which, at its turn, improved on the speed of the Brandes-based betweenness computation (the purple line in the graph) [13]
- We tested and confirmed our initial assumption that a shortest path algorithm for sparse networks is likely to speedup betweenness computation for interactome networks
- Although the Brandes algorithm remains a milestone for every researcher interested in betweenness computation, it is outperformed by the new computation technique in the case of large sparse biological networks

C. Conclusions and future developments

This paper proposed a sensibly improved technique for analyzing the interactome networks by computing the betweenness centrality for all proteins in a certain interactome network. The initial assumption that generated the novel computational model was related to the fact that interactome networks are sparse and, hence, a betweenness computation algorithm specially designed for sparse networks should perform better

TABLE II
COMPARATIVE VIEW OF THE EXECUTION TIMES (2)

Test no.	Dijkstra-generic	Brandes
1	74793	25782
2	597386	213721
3	1007235	394070
4	1763815	679694
5	5023491	8627049
6	5526159	9723802
7	7431206	12024317

on protein networks than a generic betweenness algorithm. The initial hypothesis was confirmed and the results of the testing procedure were presented and commented in the previous section. The performance gain is significant, as the computation time reduces by 70% in the case of a 9000-protein network, and even more for smaller datasets. The future of biological networks analysis using sequential betweenness computation algorithms looks very promising, as sensibly larger portions of the whole interactome can be analyzed faster than ever before. The future work will concentrate towards finding better, more efficient sequential algorithms as well as towards the discovery and implementation of some parallel algorithms for computing the betweenness centrality measure and, thus, analyzing the available proteomic data.

ACKNOWLEDGMENT

This work is supported by the Irish Research Council for Science, Engineering and Technology, under the Embark Initiative program.

REFERENCES

[1] R. Dunn et al., *The use of node-clustering to investigate biological function in protein interaction networks*. BMC Bioinformatics, 2004.
[2] D. Bader et al., *Approximating betweenness centrality*. Georgia Institute of Technology, 2007.
[3] D. Meunier and H. Paugam-Moisy, *Cluster detection algorithm in neural networks*. Institute for cognitive science, BRON, France, 2006.
[4] J. Yoon, A. Blumer and K. Lee, *An algorithm for modularity analysis of directed and weighted biological networks based on edge-betweenness centrality*. Bioinformatics, 2006.
[5] M.E.J. Newman, *Shortest paths, weighted networks, and centrality*. Physical review, volume 64, 2001.
[6] M. Girvan and M.E.J. Newman, *Community structure in social and biological networks*. State University of New Jersey, 2002.
[7] P. Holme et al., *Subnetwork hierarchies of biochemical pathways*. Bioinformatics, 2003.
[8] D. Ucar et al., *Improving functional fodularity in protein-protein interactions graphs using hub-induced subgraphs*. Ohio State University, 2007.
[9] K. Lehmann and M. Kaufmann, *Decentralized algorithms for evaluating centrality in complex networks*. IEEE, 2002.
[10] J. Griebsch et al., *A fast algorithm for the iterative calculation of betweenness centrality*. Technical University of Munchen, 2004.
[11] G.H. Traver et al., *How complete are current yeast and human protein-interaction networks?*. Genome biology, 2006.
[12] R. Bunescu et al., *Consolidating the set of known human protein-protein interactions in preparation for large-scale mapping of the human interactome*. Genome biology, 2005.
[13] U. Brandes, *A faster algorithm for betweenness centrality*. University of Konstanz, 2001.
[14] B. Preiss, *Data structures and algorithms with object-oriented design patterns in C++*. John Wiley and sons, 1998.
[15] EMBL-EBI, *The IntAct protein interactions database*. URL: http://www.ebi.ac.uk/intact/site/index.jsf, 2009.

[16] C. Demetrescu et al., *The Leonardo Library*. URL: http://www.leonardo-vm.org/, 2003.
[17] University of California, *The DIP protein interactions database*. URL: http://dip.doe-mbi.ucla.edu/, 2009.
[18] Johns Hopkins University, *The HPRD protein interactions database*. URL: http://www.hprd.org/, 2009.
[19] R. Bocu and S. Tabirca, *Betweenness Centrality Computation - A New Way for Analyzing the Biological Systems*. Proceedings of the BSB 2009 conference, Leipzig, Germany, 2009.
[20] L.C. Freeman, *A set of measures of centrality based on betweenness*. Sociometry, Vol. 40, 35-41, 1977.