Distributed protocols for Ego Betweenness Centrality computation in DOSNs

Barbara Guidi
Department of Computer Science,
University of Pisa, Italy
Email: guidi@di.unipi.it

Marco Conti
IIT-CNR
via G. Moruzzi, 1
56124 Pisa, Italy
Email: m.conti@iit.cnr.it

Andrea Passarella
IIT-CNR
via G. Moruzzi, 1
56124 Pisa, Italy
Email: a.passarella@iit.cnr.it

Laura Ricci
Department of Computer Science,
University of Pisa, Italy
Email: ricci@di.unipi.it

***Abstract*—Online Social Networks (OSNs) tipically exploit a logically centralized infrastructure which has several drawbacks including scalability, privacy, and dependence on a provider. In contrast to centralized OSNs, a Distributed Online Social Network helps to lower the cost of the provider drastically, and allows better control of user privacy. A distributed approach introduces new problems to address, as data availability or information diffusion, which require the definition of methods for the analysis of the social graph. This paper focuses the problem of the evaluation of the centrality of a nodes in a Distributed Online Social Network and proposes a distributed approach for the computation of the Ego Betweenness Centrality, which is an ego-centric method to approximate the Betweenness Centrality. We propose a set of algorithms to compute the betweenness centrality in static and dynamic graphs, which can be directed or undirected. We propose both a broadcast and a gossip protocol to compute the Ego Betweenness Centrality. A set of experimental results proving the effectiveness of our approach are presented.**

***Keywords*—Ego Betweenness Centrality, P2P, DOSN, Social Network Analysis.**

I. Introduction

Online Social Networks (OSNs) have become a common ground where people are generating and consuming huge amount of information. The currently popular OSNs are centralized which means they are based on centralized servers storing all the information. In this way, users have less control over their own data and their data is scattered over the internet in different OSN providers which usually do not support data interoperability. Centralized social networking services present several problems that include both technical and social issues that emerge as a consequence of the centralized management of the services.

A current trend for developing Online Social Network services is towards the decentralization of the OSN infrastructure. A Distributed Online Social Network (DOSN) [1] is an Online Social Network implemented on a distributed information management platform, such as a network of trusted servers, P2P systems or opportunistic networks.

Decentralizing the existing functionalities of Online Social Networks requires finding ways for providing robustness against churn, distributing storage of data, propagating updates, defining an overlay topology and a protocol enabling searching and addressing, etc. The problem of dynamism is related to social and infrastructure dynamism. The social dynamism is present in social relationships due both to the relations updates between users and to the varying number of users of the DOSN. The infrastructure dynamism is related to the underlying overlay network. Nodes may join/leave the underlying network, so that the corresponding user has a state that indicates its availability (online/offline). Nodes should be connected according to their social connections in order to cluster friends in the overlay network. This should facilitate operations as information diffusion or data storage and each update of profile from one user must be diffused to all its social connections. In term of node availability, information diffusion, but also network analysis, a centrality metric is needed. The centrality of a node in a network is a measure of the structural importance of the node. A central node, typically, has a stronger capability of connecting to other network nodes. There are several ways to measure centrality. The three most widely used centrality indexes are the degree centrality, closeness centrality and betweenness centrality [2]. The Betweenness Centrality index is the most complex and best measure suited for describing network communication based on shortest paths and predicting the congestion of a network. The computation of the Betweenness Centrality is a high time consuming task and, considering a distributed network, such as a P2P network, where nodes have a local knowledge of the network, it is not easily applicable. A more computationally efficient approach is compute the betweenness on the ego network as opposed to the global network topology. An ego network is a network consisting of a single actor (ego) together with the actors it is connected to ($alters$) and all the links among those alters [3]. A betweenness centrality based on an ego-centric approach, known as Ego Betweenness Centrality can be considers a good solution to evaluate centrality in a DOSN.

In this paper we consider the problem of computing the Ego Betweenness Centrality in Distributed Online Social Networks, and we propose two distributed protocols for the computation of Ego Betweenness Centrality. Furthermore, we evaluate the Ego Betweenness Centrality on undirected graphs by using the method proposed in [4], and we study the Ego Betweenness Centrality for directed graphs. The rest of the paper is structured as follows. We discuss the related work in Section II, the overall scenario in Section III, and the concept of the Ego Betweenness Computation in undirected and directed in graph in the sections IV and V. We provide two distributed protocols for the EBC computation in Section VI. In Section VII we present the evaluation of our proposal. We report some conclusions in Section VIII.

II. Related Work

Centrality is a fundamental concept in social network analysis to study different properties of the networks. Freeman [2] provides three basic concepts for centrality: degree, closeness and betweenness. The betweenness centrality is one of the most used centrality metrics. A more computationally efficient approach is to compute the betweenness on the ego networks. In Social Networks, an ego network is defined as a network of a single actor together with the actors it is directly connected to. The concept of Betweenness Centrality of an ego network was first introduced by [2]. [4] presents an efficient centralized and simple algorithm for calculating the betweenness score. The simulation study indicates that the local ego betweenness is highly correlated with the betweenness of the actor in the complete network. A similar analysis is shown in [5]. Classic betweenness centrality algorithms are inefficient in a distributed environment, and in most cases they are inapplicable because they require a complete knowledge of the network. [6] and [7] present egocentric betweenness centrality as the basis for a distributed routing protocol in a delay tolerant network. [8] proposes a framework for the decentralized calculation of different centrality indices which can be applied to many complex networks, such as wireless multi-hop ad-hoc communication networks, router networks, and P2P networks.

A. Centrality Indexes

The centrality indexes compute the importance (as centrality) of a node into a network, such as Distributed Online Social Networks. In the context of social network analysis several centrality indexes have been proposed.

The Degree Centrality index provides information about the number of links of each node, and it is computed as defined by the Equation (1), where $\delta(v)$ is the degree of the node v.

$$C_D(v) = \frac{\delta(v)}{|V| - 1} \tag{1}$$

From the point of view of a DOSN, the Degree Centrality can be seen as the number of social relationships of a node.

The Closeness Centrality [9] provides information about the diffusion of information, from a node to all the other nodes in the network. Closeness Centrality is computed by the Equation (2), where $d(v, t)$ id the minimum distance between the node v and the node t.

$$C_C(v) = \frac{\sum_{t \in V} d(v, t)}{|V| - 1} \tag{2}$$

Since in a DOSN the information is principally directed to 2-hops nodes, this measure is not very relevant for DOSNs. This index can be used when the information have to be diffused among all nodes in the network.

The Betweenness Centrality (BC) is one of the most famous centrality indexes. It has been introduced in [10] and [2], and it is used to evaluate the influence of a node on the flow of information in a social network context. The BC of a node v evaluates the number of shortest paths $\sigma_{s,t}(v)$ between a couple of nodes s and t that pass through v (Equation (3)). A node with a high betweenness centrality value is more likely to be located on the shortest paths between multiple node pairs in the network, and thus more information must travel through that node.

$$C(v) = \sum_{v \neq s,t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \tag{3}$$

Since the BC requires the shortest paths computation for each couple of nodes, it requires a computation cost equals to $O(nm)$, where n is the number of nodes in the graph, and m is the number of edges of the graph. Furthermore, the principal models for the BC computation consider a static graph. If the size of the graph varies, then the BC value has to be computed again. In a DOSN, this index is very important and it can be used to know which nodes are central in term of the network topology. It follows that it may well be useful to calculate the betweenness centrality in a distributed way.

III. Scenario

In a general DOSN peers are connected according to their social links, which can represent social relationships or other social aspects depending on the DOSN model. In order to create a distributed network, an interconnecting network has to be created, known as *overlay network*. The overlay network is a layer on top of the physical network connections, which creates transparent services to handle the virtual topology of the network. Overlay networks offer services for the communication and connection handling between the peers in the network, including peers that are not directly connected to the peer. In particular we talk about a *friend-to-friend* (or F2F) computer network, which is a type of P2P network in which users only make direct connections with people they know. In this kind of environment, the overlay network is called *social overlay*: peers are connected to known peers, and an edge between a pair of nodes indicates that a tie exists between two adjacent nodes. Social overlays are expected to improve:

- privacy, non-friends do not see the information disseminated,
- locality, due to network homophily,
- cooperation, due to friendship.

Each peer p knows its social relationships and it maintains a view which contains the descriptors of all the nodes which are directly connected to p.

IV. Ego Betweenness Centrality Computation in undirected graphs

The Ego Betweenness Centrality is the value of the Betweenness Centrality computed using only the nodes and the links in the ego-network of a node. The computation of the Ego Betweenness Centrality for an undirected graph has been proposed in [4]. Given A_n the adjacency matrix of the ego network of n, $EN(n)$, the EBC is the sum of the reciprocal values $A_n^2(i, j)$ such that $A_n(i, j) = 0$, as defined in (4).

$$EBC(n) = \sum_{A_n(i,j)=0, j>i} \frac{1}{A_n^2(i, j)} \tag{4}$$

The correlation between BC and EBC has not a theoretical link [4]. However, it has been verified on random networks and real networks composed by a little amount of nodes. and we have computed the correlation through experimental results.

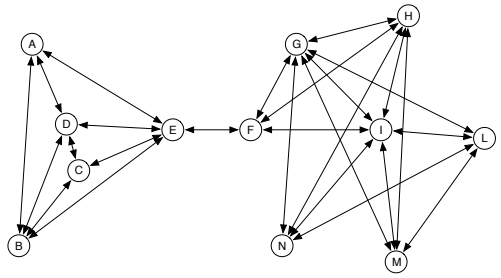

Fig. 1. An example of network graph

Figure 1 shows a simple graph, where the nodes E and F are very important because they have a high BC and EBC values because of the link between them permits the information diffusion within the two independent communities composed respectively by {A,B,C,D,E} e {F,G,H,I,L,M,N}.

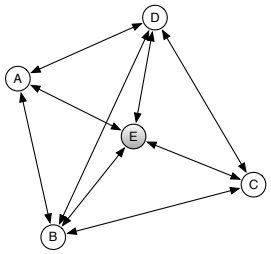

Fig. 2. Ego Network of the node E

Figures 2 shows the ego network of the node E. The alter D is able to communicate directly with {A,B,C}. The same for the node B. Nodes A and C are able to communicate through E (path A-E-C), or through B (path A-B-C), or through D (path A-D-C). Consider the adjacency matrix of the node E:

$$A_E = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} E & A & B & C & D \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array} \tag{5}$$

and its square:

$$A_E^2 = \begin{array}{c} \\ E \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} E & A & B & C & D \end{array} \\ \begin{pmatrix} 4 & 2 & 3 & 2 & 3 \\ 2 & 3 & 2 & 3 & 2 \\ 3 & 2 & 4 & 2 & 3 \\ 2 & 3 & 2 & 3 & 2 \\ 3 & 2 & 3 & 2 & 4 \end{pmatrix} \end{array} \tag{6}$$

The matrix $A_E^2(i,j)$ provides the number of geodesics of length 2 between the nodes i and j. The adjacency matrix is symmetric (due to the undirected graph) and the computation considers only the nodes i e j such that $A_E(i,j) = 0$ which are above the diagonal. The node E has an ego betweenness centrality $EBC(E) = \frac{1}{3}$, since it contributes to one path out of three possible paths between A and C.

The BC computation requires the solution of the shortest paths problem between a node and all the other nodes in the network. This problem can be computed in $O(nm)$. [11] shows an approximation method to compute the BC, and the computation complexity is $O(\sqrt{n}m)$. The EBC computation requires a computation complexity equal to $O(n^3)$ for a square matrix of nxn dimension, where n is the number of nodes contained into the ego network.

V. Ego Betweenness Centrality Computation in a directed graph

In some scenario, the communications are not bidirectional and the resulting graph is a directed graph. For example, Twitter social graph is a directed graph, while Facebook social graph is an undirected graph. The EBC is proposed and evaluated for undirected graph in [4]. We have analysed the computation of the BC on a directed graph as proposed in [12], and we propose a method to compute the EBC on a directed graph with an ego-centric approach. The proposed protocols can be used with both type of graphs. The Betweenness Centrality, as proposed in [2], may be normalised by dividing through the number of pairs of vertices not including v, which for directed graphs is $(n-1)(n-2)$ [12], and for undirected graphs is $(n-1)(n-2)/2$. The computation of the EBC on a directed graph requires some modifications to the basic algorithm. The adjacency matrix shown in (5), is not symmetric, and the computation does not consider only the nodes i e j such that $A_E(i,j) = 0$ which are above the diagonal, but also the the nodes i e j such that $A_E(i,j) = 0$ which are under the diagonal. Furthermore, in $A^2[j,k]$ we have the number of oriented path starting in j and terminating in k. Nodes j and k are alters of the ego node i ($A[i,k] = 1$ and $A[i,j] = 1$). If $A[j,i] = 1$ and $A[j,k] = 0$, then exists a minimal oriented path from j to k passing through the ego node i. We define the computation of the EBC on directed graph by adding the following step: *given the oriented adjacency matrix A for the ego node with ID equals to k, an oriented path between i and j passing through the ego node has to exists and this implies that $A[i][k] = 1$.*

VI. Distributed protocols for EBC computation

We propose two protocols to compute the EBC on a generic ego network social overlay. The proposed protocols provide a simple mechanism to compute the EBC by building the adjacency matrix of each ego node. The two protocols differ from each other in the update phase of the adjacency matrix. The first protocol is *Ego-BC broadcast* (EBC Broadcast), and each node maintains the adjacency matrix up-to-date by doing communications with all the nodes in its ego network. The second protocol called *Ego-BC Gossip* (EBC Gossip) maintains the adjacency matrix up-to-date through specific gossip techniques [13]. Both protocols can be used both with a directed and with an undirected graph.
For each node v, we define $N(v)$ as the set of nodes into its the ego network, and E_v the set of edges between the node v and the set of nodes into the ego network.

A. EBC Broadcast Protocol

In a DOSN, peers can be mobile devices or PC. By considering a mobile device, it is connected to the network

with a stable or unstable Internet connection. This kind of protocol can be used when a peer uses an unstable connection or in case of opportunistic communications. Each node, in a single step, sends to its neighbouring nodes information about its EBC. The protocol is structured according to the following steps:

- *Connection*, when a node v and a node n join to the overlay, they exchange their neighbourhood $N(n)$ and $N(v)$. Furthermore, the nodes v e n inform their neighbours $N(v)$ e $N(n)$ about the updates into their ego network. This phase is described by the algorithm 1. In this phase, each ego node v exchanges $O(|N(v|)$ messages, which are the only essential messages to notify the exchanges into the ego network.

Algorithm 1 The node v is connecting to $alter$

```
function CONNECT(alter)
    send MessageUpdate(N(v),v) to alter;
    send MessageUpdate(alter,v) to N(v);
    N(v)=N(v) U alter;
end function
```

- *Disconnection*, when a node v voluntary leaves the system, it sends a notification message ($disconnect$) to its neighbours in $N(v)$.

Each node v has a handler for receiving messages, which specifies the steps executed by v for each message. When an ego node v receives a $disconnect$ message from the node n, it updates its local data structures and communicates to neighbouring nodes the removal of the node n from its ego network through the $MessageDelete$ message. Each ego node v exchanges only $O(|N(v)|)$ messages, where $|N(v)|$ is the number of neighbouring nodes. When the node v receives messages containing updates about the ego network of its neighbours, it updates its local data structures.

When a node v login into the system, it executes one or more $connect$ procedure calls taking into account the active nodes of its ego network. Each instance of the protocol communicates only the occurred changes in its ego network, instantly or at a frequency that is considered appropriate.

B. EBC Gossip Protocol

The reference model of this protocol is SIR (Susceptible, Infected, Removed) [13]. The protocol is a *notify-pull* protocol.

The protocol exploits two types of messages:

- *UpdateRequest*. It is a request message sent from the node a to the node b to ask information about neighbouring nodes $N(b)$ of node b.
- *UpdateReply*. It is the response message sent from node b to the node a containing information about neighbouring nodes $N(b)$ of the node b

In the *pull* phase, nodes send update requests to neighboring nodes of which they do not yet know the ego network. A node can finish the pull phase when it has received all the information about the ego networks of its neighbors.

Let K_v the set of the alters of which the ego node v wants to know the ego network, the algorithm VI-B defines the pull phase, which is periodically executed (every Δ_1 time unit) for each node, until $K_v = \emptyset$.

The handler *OnUpdateRequest* manages the update requests received from the other nodes in the network. When the node v receives a update request (*UpdateRequest*) from a node n, it replies with a $UpdateMessage$ message informing all nodes about its ego network.

The handler *OnUpdateReply* manages the $UpdateReply$ messages. The function *update* executes the update of the local data structures as described into the received message.

Given a node v, it is able to know all the information about the neighbouring nodes $N(v)$ ego networks, after a certain number of cycles.

Algorithm 2 Pull

```
while K_v ≠ ∅ do
    select random alter n ∈ K_v;
    Send UpdateRequest to n;
    K_v = K_v − {n};
    wait Δ_1;
end while
```

The *notify* phase, described by the algorithm VI-B permits to communicate all the changes occurring into the ego network of the node v to all the nodes n which have the node v as an alter node. Let E_o^v the set of nodes that need to know the notification of the update o relative to the ego network of the node v. The notification phase is periodically executed, each Δ_2 time units, and it terminates when all the updates are notified. The selection of the node n to which have to be sent the updates can be random or based on the number of updates to notify. Furthermore, it is possible to reduce the number of exchanged messages with the aggregation of the all notifications o (the set $\{o \mid n \in E_o^v\}$) into a single message.

Algorithm 3 Notify

```
function NOTIFY
    while (∃ o : E_o^v ≠ ∅) do
        select n ∈ E_o^v;
        Send UpdateMessage({o | n ∈ E_o^v}) to n;
        for {o | n ∈ E_o^v} do
            E_o^v = E_o^v − {n};
        end for
        wait Δ_2;
    end while
end function
```

The two phases of the protocol can be separated and executed with different frequency depending of the dynamics of the network.

Let us now show how algorithms and are exploited to implement the basic operations of a peer participating to a social overlay:

- *Join the network*. When a node joins the network for the first time, it executes one or more *Pull* phases (Algorithm VI-B).

- *Add a new link*. When a new link between two nodes a and b is created, the two nodes update the list of nodes for which is unknown the neighbouring nodes $K_a = K_a \cup b$ e $K_b = K_b \cup a$. So that, it is executed a *Pull* phase (Algorithm VI-B) to request the update. Furthermore, the nodes a and b add a new entry into their lists of updates to report $E^a_{o_b}$ e $E^b_{o_a}$ (they notify the presence of a new link to their neighbouring nodes (Algorithm VI-B)).
- *Remove a link*. When a node v removes a link to its alter $a \in N(v)$, it notifies the removal to its neighbouring nodes by inserting into its updates list the notification o_a relative to the removal of the node a, $E^v_{o_a}$. (Algorithm VI-B).
- *Updating of link properties*. When a node v updates its links properties p, it notifies the changes to its neighbouring nodes by inserting into the updates list E^v the entry relative to p (Algorithm VI-B).

The Pull phase for a generic node v, explained by the algorithm VI-B, terminates after $O(|N(v)|)\Delta_1$ gossip cycles and it requires a request message (*UpdateRequest*) and a response message (*UpdateMessage*) for each cycle. The Notify phase requires a single message for each gossip cycle Δ_2 and the termination depends by the temporal characteristics of the system.

VII. Experimental Results

The proposed protocols have been implemented using the P2P Peersim simulator [14], which is written in java. The used dataset is obtained by a Facebook Regional Network[1] and it is composed by:

- A *Social Graph*: an undirected graph which defines the whole network structure. An edge is a relatioships between two Facebook users.
- Four *Interaction Graphs*: directed graphs which define the interaction between users within different time windows: *last month*, *last 6 months*, *last year*, *2004-2008*. The Interaction graph contains an edge for each interaction (Post or Photo Comment) between two users happened in the considered time window.

The Social Graph has been used to evaluate the EBC computation on undirected graphs and the Interaction Graph to evaluate the EBC on directed graphs. If a user j has done an interaction, i.e. a comment, to a user i, a link from j to i is created in the Interaction Graph.

Table I shows some characteristics of the Social Graph computed in [15].

# Nodes	3,097,165
# Edges	23,667,394
Average Degree	15.283
Average Clustering Coefficient	0.098
Assortativity	0.048

TABLE I. Social Graph

[1] online available http://current.cs.ucsb.edu/facebook/

In a dynamic environment, nodes can be online and/or offline. We have evaluated the computation of the EBC for two different scenarios:

- *static network*. The computation analyzes all the social relationships, independently from the online/offline status of each nodes. Each node is considered online and the static EBC (*StaticEBC*) evaluates the social importance of each node according to its social relationships, independently from the network configuration.
- *dynamic network*. The computation analyzes the importance of a node with respect to its online/offline status. The dynamic EBC (*DynamicEBC*) does not consider the offline nodes. To simulate node churn, we have used a well-known churn model [16].

We have evaluated the correlation between the BC and the EBC on the social graph and on the interaction graphs by using the Pearson correlation. We have evaluated our protocols and the correlation between EBC and BC on directed/undirected graphs by extracting different networks randomly chosen from the data set and by varying the number of nodes.

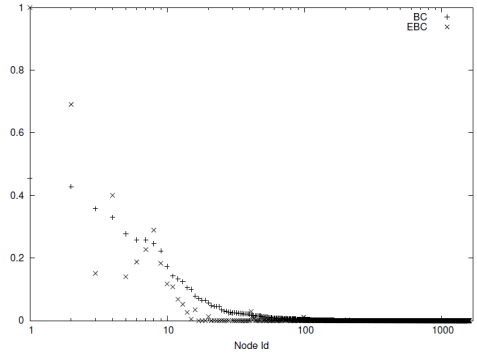

Fig. 3. Normalized values for BC and EBC for a random network extracted from the data set and composed by 1595 nodes

Figure 3 shows the correlation between BC and EBC for an undirected graph containing 1595 nodes, and figure 4 shows the correlation for a directed graph containing 5000 nodes. There is a strong correlation between the two metrics, and they are the same for nodes with centrality equals to 0.

We have evaluated the average number of messages sent for each protocol by varying the number of nodes in the network on 10 iterations (Table II). The number of messages sent by the broadcast protocol are on average about three times the number of messages sent by the gossip protocol. The broadcast protocol has been introduced to provide a fast message delivery between nodes, but this implies a larger number of messages.

#Nodes	EBC Gossip Protocol	EBC Broadcast Protocol
1000	29646	64580
2000	62581	160006
3000	121220	358254
4000	100824	249187
5000	241339	776880

TABLE II. Number of Messages of the both protocols by varying the number of nodes

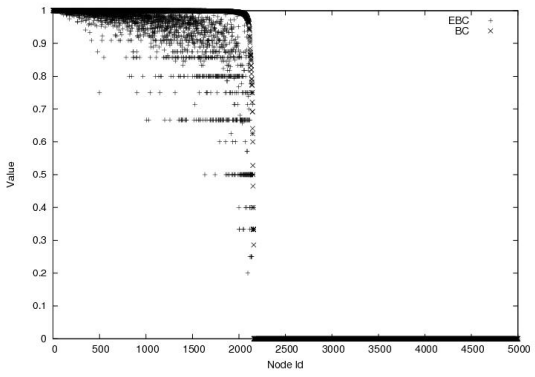

Fig. 4. Normalized values for BC and EBC for a random network extracted from the data set and composed by 5000 nodes

The gossip protocol provides message delivery with a small number of messages, and it can be used to prevent the network congestion or in case of devices with low capabilities.

A. EBC in a dynamic environment

In a distributed environment, peers can be online or offline. The status of peers changes the underlying graph on which the EBC is calculated. Each time the ego network of a peer is changed, the EBC has to be computed again. We have evaluated how often this computation is required by considering a network of 1000 peers and by executing ten experiments. The simulation is organized on 150 time instants, where a time instant $t \in [0, 149]$ corresponds to a minute (in according to the churn model [16]). We have evaluated the average number of time instant after that a peer has to recalculate the EBC.

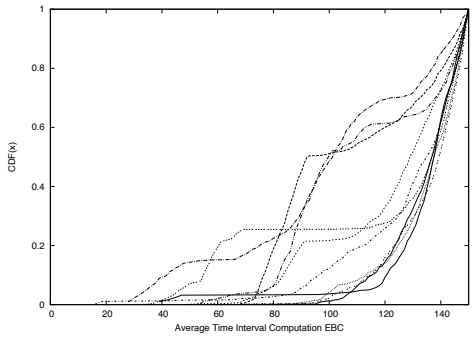

Fig. 5. Evaluation of how often the EBC computation is required in a network of 1000 nodes and 150 time instants.

Figure 5 shows, for each experiment, on the x-axis the time instants x, while on the y-axis the cumulative function of the number of nodes which have to recalculate the EBC at a time instant less or equals to x. The results show that about 50% of nodes recalculate the EBC after an average number of time instants less than 90 and 140. Furthermore, the majority of nodes performs the calculation of the EBC after at least 60 cycles of simulation.

VIII. Conclusion and Future works

In this work, we have evaluated a distributed computation of the EBC on undirected graphs, and we have studied the computation of the EBC on directed graph. Furthermore, we have provided two distributed protocols which can be used on directed and undirected graphs. The experimental results show the strong correlation between BC and EBC on directed/undirected graphs extracted from the Facebook regional data set. Furthermore, we have evaluated by using a well-known churn model how many times a node has to recalculate its ego betweenness. In ongoing and future work, we anticipate that our focus will be on optimizing the EBC recalculation in dynamic environment and the study of the EBC computation in a specific social overlay by using a weighted version of the social graph.

References

[1] A. Datta, S. Buchegger, L. Vu, T. Strufe, and K. Rzadca, "Decentralized online social networks." in *Handbook of Social Network Technologies*, B. Furht, Ed. Springer, 2010, pp. 349–378.

[2] Freeman, Linton C., "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, 1977.

[3] L. C. Freeman, "Centered graphs and the structure of ego networks." *Mathematical Social Sciences*, vol. 3, no. 3, pp. 291–304, 1982.

[4] M. G. Everett and S. P. Borgatti, "Ego network betweenness." vol. 27, no. 1, 2005, pp. 31–38.

[5] P. V. Marsden, "Egocentric and sociocentric measures of network centrality." *Social Networks*, vol. 24, no. 4, pp. 407–422, 2002.

[6] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '07. ACM, 2007, pp. 32–40.

[7] E. M. Daly and M. Haahr, "Social network analysis for information flow in disconnected delay-tolerant manets," *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, pp. 606–621, May 2009.

[8] K. Lehmann and M. Kaufmann, *Decentralized Algorithms for Evaluating Centrality in Complex Networks*, ser. WSI: Wilhelm-Schickard-Institut für Informatik. WSI, 2003.

[9] Gert Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966.

[10] J. M. Anthonisse, *The rush in a directed graph*. SMC, 1971.

[11] S. Y. Chan, I. X. Leung, and P. Liò, "Fast centrality approximation in modular networks," in *Proceedings of the 1st ACM international workshop on Complex Networks in Information and Knowledge Management*, ser. CNIKM '09. ACM, 2009, pp. 31–38.

[12] D. R. White and S. P. Borgatti, "Betweenness centrality measures for directed graphs," *Social Networks*, vol. 16, no. 4, pp. 335–346, 1994.

[13] Márk Jelasity, "Gossip," in *Self-organising Software*, ser. Natural Computing Series, G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, Eds. Springer-Verlag, 2011, vol. 12, pp. 139–162.

[14] A. Montresor and M. Jelasity, "Peersim: A scalable p2p simulator." in *Peer-to-Peer Computing*, H. Schulzrinne, K. Aberer, and A. Datta, Eds. IEEE, 2009, pp. 99–100.

[15] V. Arnaboldi, M. Conti, A. Passarella, and F. Pezzoni, "Analysis of ego network structure in online social networks," in *SocialCom/PASSAT*. IEEE Computer Society, 2012, pp. 31–40.

[16] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured p2p networks." in *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*. IEEE Computer Society, 2006, pp. 32–41.